Low-Rank Exploiting Optimization Methods for Inverse Problems and Machine Learning

By

Kai Fung Kan
Doctor of Philosophy

Mathematics

Lars Ruthotto, Ph.D.
Advisor

James Nagy, Ph.D.
Committee Member

Yuanzhe Xi, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Low-Rank Exploiting Optimization Methods for Inverse Problems and Machine Learning

By

Kai Fung Kan
B.Sc., The Chinese University of Hong Kong, Hong Kong, 2016
M.Phil., The Chinese University of Hong Kong, Hong Kong, 2018

Advisor: Lars Ruthotto, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2023

Abstract

Low-Rank Exploiting Optimization Methods for Inverse Problems and Machine Learning

By Kai Fung Kan

Due to rapid technological development, datasets of enormous size have emerged in various domains, including inverse problems and machine learning. Many important applications in these domains, e.g., PDE parameter estimation, data classification and regression, are formulated as optimization problems. These problems are often of large-scale and can be computationally intractable to solve. Fortunately, it has been empirically observed that large datasets can be accurately estimated by low-rank approximation. Specifically, they can be approximately expressed using a relatively compact representation whose computation is less demanding. Therefore, an effective way to circumvent the computational obstacle is to exploit the low-rank approximation. In addition, low-rank approximation can serve as a regularization technique to filter out irrelevant features (e.g., noise) from the data since it can capture essential features while discarding less pertinent ones.

This dissertation presents three applications of low-rank exploiting optimization methods for inverse problems and optimization. The first application is a projected Newton-Krylov method which efficiently exploits the low-rank approximation to the Hessian matrix to compute the projection for bound-constrained optimization problems. The second application is a modified Newton-Krylov method geared toward log-sum-exp minimization for a linear model. It is scalable to large problem sizes thanks to its utilization of the low-rank approximation to the Hessian. In the third application, we apply hybrid regularization, which synergistically combines iterative and Tikhonov regularization, to effectively and automatically avoid the double descent phenomenon in machine learning.

Low-Rank Exploiting Optimization Methods for Inverse Problems and Machine Learning

By

Kai Fung Kan
B.Sc., The Chinese University of Hong Kong, Hong Kong, 2016
M.Phil., The Chinese University of Hong Kong, Hong Kong, 2018

Advisor: Lars Ruthotto, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2023

Acknowledgments

I would like to express my deepest gratitude to my advisors Prof. Lars Ruthotto and Prof. James Nagy for their constant support and caring throughout my Ph.D. studies. In addition to their excellent academic and mathematical training, they have also distilled in me the value of integrity, and as role models, demonstrated to me how to be a better person. I feel extremely fortunate to be their student, and I am grateful to them beyond words.

I would like to thank my M.Phil. advisor Prof. Raymond Chan and collaborators Prof. Robert Plemmons and Dr. Alfred Ma. They have taught me many valuable lessons which have well-equipped me for my Ph.D. studies.

My thanks also go to my thesis committee member Prof. Yuanzhe Xi, who has given me valuable suggestions for my thesis and advice for my career path.

I would also like to thank my academic brothers and sisters: Samy Wu Fung, Yunyi (Larry) Hu, Chang Meng, Derek Onken, Elizabeth Newman, Ying Wai (Daniel) Fan, Xingjian Li, Deepanshu Verma, Nicole Yang, Malena Sabaté Landman, Abbey Julian, Malvern Madondo, Ariana Brown, and Haley Rosso. I have learned a lot from discussing research with them.

A huge shoutout to my graduate school friends Jack Barlow, Gary Vestal, James Robb, Davide Evangelista, Marcelo Sales, Irving Martínez, Chris Keyes, Haozhe Yu, Jennifer Wang, Jayanth Guhan, Alessandro Barone, Guangqiu Liang, Shifan Zhao, Ayush Basu, Ru Huang, Shilpi Mandal, Maxwell Auerbach, Olivia Greathouse, Benjamin Yellin, Vishwanath Seshagiri, Alex Dunbar, Sreejani Chaudhury, Anton Molnar, Francesco Brarda, Katherine Keegan, Elle Buser, Emma Hart, Ylli Andoni, Sean Longbrake, and Griffin Johnston. My graduate school life has always been joyful thanks to all of them.

Lastly but most importantly, I would like to thank my parents Chun Keung Kan and Sau Ying Wong and my brother Kai Yuen Kan for their unconditional love and selfless devotion throughout my life. I would like to thank my fiancée Shuk Ching Ho, with whom I have had a five-year long distance relationship, for her love, trust and extreme patience all these years. I look forward to starting our new lives together after my graduation. I also thank my future family-in-law for always being supportive.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The unprecedentedly rapid advancement of technology in recent decades has given rise to large-scale data generation and collection. Consequently, data of massive size have emerged in a wide spectrum of domains, including inverse problems and machine learning. This is often referred to as the era of big data. Many important applications in these domains, e.g., image processing [132, 20, 18, 80, 81, 138], PDE parameter estimation [13, 23, 42, 32, 152, 151], image classification [109, 63, 19, 153, 119, 120], and numerous others, can be performed by solving optimization problems. However, solving optimization problems can be computationally prohibitive due to the enormous data size. Fortunately, it has been commonly observed that big data have intrinsically low-rank structure [159, 142]. To be precise, a data matrix $\mathbf{A} \in \mathbf{R}^{m\times n}$ can be estimated effectively by a low-rank approximation $\mathbf{A} \approx \mathbf{BC}$, where $\mathbf{B} \in \mathbb{R}^{m\times r}$, $\mathbf{C} \in \mathbb{R}^{r\times n}$, and $r < \min\{m, n\}$. Since the low-rank approximation stores only $r(m+n)$ entries compared to the original mn entries, it has a reduced complexity compared to the original matrix. This motivates the use of low-rank approximation to overcome the computational burden. Moreover, since low-rank approximation seeks to capture essential features and discards less relevant information (e.g., noise), it can be applied as a regularization scheme during optimization.

1.1 Related Work and Contributions

In this dissertation, we introduce three applications of low-rank exploiting optimization methods. These applications can be grouped into two categories. The first category is Newton-type methods that exploit the low-rank approximation to the Hessian matrix. The second category is hybrid regularization methods that exploit the low-rank structures of data matrices.

1.1.1 A Projected Newton-Krylov Method

The first application is PNKH-B, a projected Newton-Krylov method designed for large-scale bound-constrained optimization. Generally, in each iteration of projected Newton-type methods, the Newton step is first computed and then projected onto the constraints to render it feasible. Note that the Newton step (search direction) is induced by the Hessian metric. Existing methods can be classified into two categories by their choice of projection metrics: 1. two-metric schemes with a piecewise linear arc [10, 47, 64, 73, 85, 95, 96, 97, 137], whose projection uses the Euclidean metric which is inconsistent with the Hessian metric used in the search direction, and 2. one-metric schemes with a linear arc [16, 113, 5, 6, 68, 100, 136], whose projection uses the Hessian metric and the line search is linear. Unlike existing methods, PNKH-B is a (generalized) one-metric scheme with a piecewise linear arc. To be precise, PNKH-B projects the Newton step onto the bound constraints with respect to the Hessian metric and performs an adaptive piecewise linear line search. This combination of Hessian projection metric and piecewise linear line search allows PNKH-B to perform a more natural and effective line search and achieve better convergence. However, in each line search, it is required to solve a quadratic projection problem that has no analytic solution and is computationally prohibitive, especially for large-scale problems.

To overcome the computational hurdle, we apply Lanczos tridiagonalization to construct a low-rank approximation for the Hessian metric. This renders the projection problem computationally tractable. More specifically, our main contribution in this work is an interior point method implementation, tailored to effectively exploit the low-rank approximation to the Hessian matrix, for solving the projection problem. The interior point method implementation only scales linearly with respect to the number of variables; hence it only adds negligible computational cost to the algorithm. We also prove the global convergence of PNKH-B to a stationary point under standard assumptions. Three numerical experiments on parameter estimation, machine learning, and image reconstruction show that PNKH-B has significantly better convergence, especially in the first few iterations. This is due to the consistent use of the Hessian metric, adaptive line search, and the low-rank approximation to reduce the otherwise intractable computational cost significantly.

1.1.2 A Modified Newton-Krylov Method

The second approach is a modified Newton-Krylov method geared toward log-sum-exp minimization for a linear model. This kind of problem arises commonly in multinomial logistic regression [153, 119] and geometric programming [140, 154, 157] and is often of large-scale.

Although the log-sum-exp function is smooth and convex, a standard implementation of line search Newton-type schemes can be problematic since the quadratic approximation can be unbounded from below. On the one hand, this problem is less significant in the presence of a Tikhonov regularization [41, 59, 70]. However, the regularization introduces a bias and thus changes the optimal solution. On the other hand, the log-sum-exp minimization problem can be formulated using disciplined convex programming (DCP) packages [61] and solved with several backend solvers [2, 141, 139]. While the problem can be solved reliably even in large-scale settings, it can be computationally intractable. Moreover, DCP packages generally do not support matrix-free settings where the linear model is not known, but routines for performing matrix-vector products are provided.

Standard modified Newton-type methods are an effective class of schemes to solve optimization problems with indefinite or positive semi-definite but rank-deficient Hessian. The idea is to modify the Hessian to render it sufficiently positive definite. In this case, the quadratic approximation is bounded from below, and the convergence issues can be avoided. Typical choices of Hessian modification include amending its eigenvalues [62, 122], adding a multiple of the identity [101, 106, 127], and altering its factorization [54, 114, 117].

In this work, we propose a novel modified Newton-Krylov method tailored for log-sum-exp minimization for a linear model. The main novelty is a Hessian shift defined in the output space of the linear model. This is motivated by the fact that, in machine learning the model input often does not have an intuitive meaning, but the model output is interpretable. Although unlike standard modified Newton-type methods, our Hessian modification can be rank deficient, we show that the shift renders the quadratic approximation to be bounded from below and the update directions are in the output space of the linear model. Consequently, the proposed scheme provably converges to a global minimizer under standard assumptions. The proposed scheme applies a Krylov subspace method to construct a low-rank approximation to the Hessian matrix and compute the update direction, thus it only requires matrix-vector products with the linear model and is scalable to large problem sizes. Two numerical experiments motivated by image classification and geometric programming show that the proposed method is competitive in terms of accuracy, time-to-solution and robustness with standard Newton-Krylov methods, disciplined convex programming approaches, and natural gradient descent.

1.1.3 Hybrid Regularization for Avoiding the Double Descent Phenomenon

In the third application, we apply hybrid regularization methods [27, 25, 26, 24] to effectively and automatically avoid the double descent phenomenon [7, 1, 104, 8, 72, 108] arising in the training of random feature models (RFMs) [131, 79].

Hybrid regularization methods are a class of efficient regularization methods to tackle ill-posed linear inverse problems. In particular, hybrid regularization methods combine the two most common and successful regularization schemes: iterative regularization [104, 156, 27] and Tikhonov regularization [41, 69, 60]. Iterative regularization applies, e.g., Krylov subspace methods [135, 56, 125, 126], to construct a low-rank approximation for the data matrix so as to project the problem onto a reduced dimensional subspace in each iteration. This not only generates practical regularization properties [24], but also is used to overcome computational bottlenecks for large-scale problems. In Tikhonov regularization, a regularization term, which incorporates prior knowledge of the true solution, is added to the objective function in order to promote certain properties. However, the parameter selection for iterative and Tikhonov regularization is not trivial and often requires solving a high-dimensional parameter searching or a grid search for which a dedicated validation dataset is needed. Hybrid regularization methods combine these two regularization schemes such that their respective strengths are fully utilized and synergized, and their respective weaknesses are circumvented.

The distinguishing characteristic of the double descent phenomenon is a surge in the generalization gap when the number of features in the RFM equals the number of training samples. Our numerical experiments on image classification demonstrate that this is due to the ill-posedness of the training problem. Hence, this motivates us to apply hybrid regularization methods to tackle the ill-posed training problem effectively. Our main contribution in this work is the first use case of hybrid regularization in machine learning. In this application, generalized cross-validation (GCV) is used to adaptively and automatically select the parameter for Tikhonov regularization and determine the stopping iteration. This avoids the necessity of parameter tuning and a dedicated validation dataset. In our numerical experiments, hybrid regularization methods successfully avoid the double descent phenomenon, are robust with respect to different stopping criteria, and yield RFMs whose generalization is comparable to optimally tuned classical regularization methods while having comparable computational costs.

1.2 Overview of Dissertation

This dissertation is organized as follows. In Chapter 2, we present the mathematical background relevant to this dissertation. The review covers low-rank approximation methods, line search Newton-type methods, and supervised classification problems. In Chapter 3, we motivate the use of a one-metric scheme for solving bound-constrained optimization problems. Based on this, we propose PNKH-B, which is a projected Newton-Krylov method and can be seen as a generalized one-metric scheme. We introduce an interior point method implementation that effectively exploits the low-rank approximation of the Hessian and solves the projection problem in a tractable way even for large-scale problems. We prove that PNKH-B globally converges to a stationary point. We compare PNKH-B with two-metric schemes in three numerical experiments. In Chapter 4, we describe the convergence issues with standard line search Newton-type methods for minimizing the log-sum-exp function for a linear model. To overcome the issues, we introduce a novel modified Newton scheme that adds a shift to the Hessian in the output space of the linear model. A global convergence guarantee is provided for the scheme. We compare the proposed scheme with common solvers for log-sum-exp minimization in two numerical experiments. We shift our focus in Chapter 5 and apply a hybrid regularization scheme to avoid the double descent phenomenon arising in training random feature models. Hybrid regularization combines iterative and Tikhonov regularization in a synergistic way. We show that the hybrid regularization scheme has comparable performance with optimally tuned gradient flow and Tikhonov regularization. We provide concluding remarks and motivate future work that can be extended from this dissertation in Chapter 6.

Chapter 2

Preliminaries

This chapter gives an overview of the mathematical background relevant to this dissertation. We review low-rank approximation methods in Section 2.1. We then describe line search Newton-type methods in Section 2.2. We conclude the chapter by presenting supervised classification problems in Section 2.3.

2.1 Low-Rank Approximation Methods

Low-rank approximation methods aim to represent a given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ in a more compact form while maintaining a limited information loss. A low-rank approximation can generally be represented as $\mathbf{A} \approx \mathbf{BC}$, where $\mathbf{B} \in \mathbb{R}^{m \times r}$, $\mathbf{C} \in \mathbb{R}^{r \times n}$ and $r < \min\{m, n\}$. We see that the low-rank approximation stores only $r(m+n)$ entries compared to the original mn entries; thus it has a reduced complexity in terms of computation and storage.

Low-rank approximation methods are applied extensively in areas like optimization, machine learning, and inverse problems. For instance, low-rank approximation methods can be applied as a dimensionality reduction technique to circumvent computational bottlenecks for large-scale optimization problems which would otherwise be intractable to solve [12, 83, 105]. By constructing an accurate estimation with a more compact representation, low-rank approximation methods can capture the essential features while discarding irrelevant information (e.g., noise) from the data [159, 80, 81, 138]. Moreover, low-rank approximation methods can serve as a regularization scheme for ill-posed inverse problems to filter out components in the forward model that are sensitive to small perturbations in data [24, 112, 130].

In the following, we review two common types of low-rank approximation methods pertinent to this dissertation: truncated singular value decomposition and Krylov subspace methods.

2.1.1 Truncated Singular Value Decomposition

Perhaps one of the most common low-rank approximation techniques is truncated singular value decomposition (TSVD). To begin our discussion, we first introduce the singular value decomposition (SVD) of $\mathbf{A} \in \mathbb{R}^{m\times n}$ given by

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top, \tag{2.1}$$

where

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_m] \in \mathbb{R}^{m\times m}, \quad \mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n] \in \mathbb{R}^{n\times n},$$

and

$$\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_r, 0, ..., 0) \in \mathbb{R}^{m\times n},$$
$$\text{with } \ r = \mathrm{rank}(\mathbf{A}) \ \text{ and } \ \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0.$$

Here diag represents a diagonal matrix, and σ_i's are called the singular values of $\mathbf{A}$. The vectors $\mathbf{u}_i$'s and $\mathbf{v}_i$'s are called the left and right singular vectors of $\mathbf{A}$, respectively. The SVD (2.1) of a matrix with real entries always exists [56, Theorem 2.4.1].

By re-writing (2.1), we obtain a representation of $\mathbf{A}$ as a sum of rank-1 matrices

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top. \tag{2.2}$$

Since the σ_i's are non-decreasing, an intuitive way to construct a rank-k approximation with $k < r$ is the TSVD defined as

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top. \tag{2.3}$$

Indeed, this intuition is verified by the Eckhart-Young-Mirsky theorem [40, 110, 58] stated in the following.

Theorem 1 (The Eckhart-Young-Mirsky Theorem)**.** *Let the SVD of* $\mathbf{A}$ *be given by* (2.1) *and* $k < r = rank(\mathbf{A})$*, then*

$$\min_{rank(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}, \tag{2.4}$$

and

$$\min_{rank(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_F = \|\mathbf{A} - \mathbf{A}_k\|_F = \left(\sum_{i=k+1}^{p} \sigma_i^2 \right)^{\frac{1}{2}}. \tag{2.5}$$

Here $\|\cdot\|_F$ *denotes the Frobenius norm, and* $\mathbf{A}_k$ *is the TSVD defined in* (2.2)*.*

Essentially, the Eckhart-Young-Mirsky theorem says that the best rank-k approximation to $\mathbf{A}$ in terms of both l_2-norm and Frobenius norm is the TSVD. It shows that the TSVD is a very powerful low-rank approximation method because it maintains minimum information loss with a given rank in terms of the two norms.

2.1.2 Krylov Subspace Methods

We discuss Krylov subspace methods, a class of iterative algorithms that constructs a low-rank approximation to a given matrix $\mathbf{A}$. For large-scale problems, Krylov subspace methods are generally preferred over TSVD. This is because they do not require the matrix $\mathbf{A}$ to be built explicitly but only require routines to perform matrix-vector products with $\mathbf{A}$ and $\mathbf{A}^\top$. We review three Krylov subspace methods that are applied in this dissertation.

Lanczos Tridiagonalization Lanczos tridiagonalization [94] seeks to solve

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.6}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a *symmetric* matrix. At the kth iteration, it computes the iterate $\mathbf{x}_k \in \mathcal{K}_k(\mathbf{A}, \mathbf{b})$ under the assumption that the initial vector $\mathbf{x}_0 = \mathbf{0}$, the zero vector. Here $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ is the Krylov subspace defined as

$$\mathcal{K}_k(\mathbf{A}, \mathbf{b}) := \text{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, ..., \mathbf{A}^{k-1}\mathbf{b}\}. \tag{2.7}$$

In particular, at the kth iteration, Lanczos tridiagonalization first generates

$$\mathbf{A} \approx \mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^\top, \tag{2.8}$$

where the columns of $\mathbf{V}_k \in \mathbb{R}^{n\times k}$ form an orthonomal basis for the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$, and $\mathbf{T}_k \in \mathbb{R}^{k\times k}$ is a tridiagonal matrix. The kth iterate is then computed by

$$\mathbf{x}_k = \mathbf{V}_k\mathbf{T}_k^{-1}\mathbf{V}_k^\top\mathbf{b}. \tag{2.9}$$

We note that Lanczos tridiagonalization is particularly useful for situations where multiple instances of (2.6) with the same $\mathbf{A}$ and different $\mathbf{b}$'s are to be solved. This is because the solutions can be obtained by (2.9) by re-using (2.8), which does not require performing the algorithm again. An implementation of Lanczos tridiagonalization is given in Algorithm 1.

Algorithm 1: Lanczos Tridiagonalization

```
 1: Initialize $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}$, $\eta_1 = \|\mathbf{r}_0\|_2$, $\mathbf{v}_0 = \mathbf{0}$, and $\mathbf{v}_1 = \mathbf{r}_0/\eta$.
 2: for $j = 1, 2, \ldots, k$ do
 3:     $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \eta_j \mathbf{v}_{j-1}$
 4:     $\gamma_j = \mathbf{w}_j^\top \mathbf{v}_j$
 5:     $\mathbf{w}_j = \mathbf{w}_j - \gamma_j \mathbf{v}_j$
 6:     $\eta_{j+1} = \|\mathbf{w}_j\|_2$
 7:     if $\eta_{j+1} = 0$ then
 8:         set $k = j$ and break
 9:     end if
10:     $\mathbf{v}_{j+1} = \mathbf{w}_j/\eta_{j+1}$
11: end for
12: Output: $\mathbf{T}_k = \text{tridiag}(\eta_i, \gamma_i, \eta_i)$, $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$
13: Output: $\mathbf{x}_k = \mathbf{V}_k \mathbf{T}_k^{-1} \mathbf{V}_k^\top \mathbf{b}$
```

Conjugate Gradient Method The conjugate gradient (CG) method [74] is for solving (2.6) with a *symmetric positive definite* (SPD) matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. An outline of CG is given in Algorithm 2. A hallmark property for CG is that it generates update $\mathbf{p}_i$'s such that they are conjugate to each other with respect to $\mathbf{A}$, that is $\mathbf{p}_i^\top \mathbf{A}\mathbf{p}_j = 0$ for all $i \neq j$. CG can be derived by reformulating the iteration of Lanczos tridiagonalization [56, Chapter 11.3.4]. Therefore, it returns the same iterate $\mathbf{x}_k$ as Lanczos tridiagonalization in exact arithmetic and is regarded as a Krylov subspace method. Compared to Lanczos tridiagonalization, CG is more memory efficient as it only stores iterates in the previous iteration. However, when one needs to solve (2.6) with a different right-hand side, CG must be performed again to obtain the new solution.

With an initial vector $\mathbf{x}_0 = \mathbf{0}$, the CG iterate $\mathbf{x}_k$ satisfies

$$\mathbf{x}_k = \underset{\mathbf{x}\in\mathcal{K}_k(\mathbf{A},\mathbf{b})}{\arg\min} \|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{A}}, \quad \text{where} \quad \mathbf{A}\mathbf{x}^* = \mathbf{b}.$$

In other words, the CG iterate minimizes the error in the $\mathbf{A}$-norm over the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$. Since when $k = n$ the Krylov subspace is $\mathbb{R}^n$, CG converges in at most n iterations in exact arithmetic. In fact, it can converge even faster than

Algorithm 2: Conjugate Gradient Method

```
1: Initialize $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}$, $\mathbf{p}_0 = \mathbf{r}_0$.
2: for $j = 0, 1, 2, \ldots, k-1$ do
3:     $\tau_j = (\mathbf{r}_j^\top \mathbf{r}_j)/(\mathbf{p}_j^\top \mathbf{A}\mathbf{p}_j)$
4:     $\mathbf{x}_{j+1} = \mathbf{x}_j + \tau_j \mathbf{p}_j$
5:     $\mathbf{r}_{j+1} = \mathbf{r}_j - \tau_j \mathbf{A}\mathbf{p}_j$
6:     $\nu_j = (\mathbf{r}_{j+1}^\top \mathbf{r}_{j+1})/(\mathbf{r}_j^\top \mathbf{r}_j)$
7:     $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \nu_j \mathbf{p}_j$
8: end for
9: Output: $\mathbf{x}_k$
```

that, as it is guaranteed to converge in p iterations [122, Theorem 5.4], where p is the number of distinct eigenvalues of $\mathbf{A}$. This renders CG a remarkable tool for solving SPD linear systems.

Paige-Saunders Bidiagonalization Paige-Saunders bidiagonalization [55, 125] targets to transform a generally rectangular shaped $\mathbf{A} \in \mathbb{R}^{m\times n}$, where m does not necessarily equal to n, into a lower bidiagonal form. In particular, at the kth iteration it generates $\mathbf{V}_k \in \mathbb{R}^{n\times k}$ and $\mathbf{U}_k \in \mathbb{R}^{m\times(k+1)}$, whose columns form an orthonormal basis for $\mathcal{K}_k(\mathbf{A}^\top\mathbf{A}, \mathbf{A}^\top\mathbf{b})$ and $\mathcal{K}_{k+1}(\mathbf{A}\mathbf{A}^\top, \mathbf{b})$, respectively, and a lower bidiagonal $\mathbf{B}_k \in \mathbb{R}^{(k+1)\times k}$ such that the following relationships hold

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_k\mathbf{B}_k, \tag{2.10}$$

$$\mathbf{A}^\top\mathbf{U}_k = \mathbf{V}_k\mathbf{B}_k^\top + \alpha_{k+1}\mathbf{v}_{k+1}\mathbf{e}_{k+1}^\top. \tag{2.11}$$

Here $\mathbf{e}_{k+1} \in \mathbb{R}^{k+1}$ is the (k+1)th standard basis vector, α_{k+1} and $\mathbf{v}_{k+1}$ will be the $(k+1)$th diagonal entry of $\mathbf{B}_{k+1}$ and the $(k+1)$th column of $\mathbf{V}_{k+1}$, respectively. The algorithm is presented in Algorithm 3. Applications of Paige-Saunders bidiagonalization include that it is used as the backbone of the LSQR [125, 126] algorithm for solving least-squares problems.

Algorithm 3: Paige-Saunders Bidiagonalization

```
1: Initialize $\mathbf{u}_1 = \mathbf{b}/\|\mathbf{b}\|_2$, $\mathbf{v}_0 = \mathbf{0}$, $\beta_0 = 0$
2: for $j = 1, 2, \ldots, k$ do
3:   $\mathbf{r}_j = \mathbf{A}^\top \mathbf{u}_j - \beta_{j-1}\mathbf{v}_{j-1}$
4:   $\alpha_j = \|\mathbf{r}_j\|_2$
5:   $\mathbf{v}_j = \mathbf{r}_j/\alpha_j$
6:   $\mathbf{p}_j = \mathbf{A}\mathbf{v}_j - \alpha_j \mathbf{u}_j$
7:   $\beta_j = \|\mathbf{p}_j\|_2$
8:   if $\beta_j = 0$ then
9:     set $k = j - 1$ and break
10:  end if
11:  $\mathbf{u}_{j+1} = \mathbf{p}_j/\beta_j$
12: end for
13: Output: $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$, $\mathbf{U}_k = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{k+1}]$
14: Output: $\mathbf{B}_k = \text{lower\_bidiag}(\beta_i, \alpha_i)$
```

2.2 Line Search Newton-Type Methods

Line search Newton-type methods are an effective class of solvers for optimization problems. They have been applied extensively thanks to their ability to incorporate curvature information and their superior local convergence rate. In this section, we review line search Newton-type methods related to this dissertation.

Newton's Method We begin our discussion by deriving the standard line search Newton's method. The method is usually obtained by an iterative minimization of a local quadratic approximation. Here we use an alternative derivation, which argues that the Newton update is the steepest descent direction with respect to the metric induced by the Hessian. This perspective inspires the design of our projected Newton-Krylov method in Chapter 3 and is also presented in [14, Chapter 9.4].

We consider an optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}), \tag{2.12}$$

where f is twice differentiable. We define an iterative scheme that at the ith iteration reads

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mu_i \Delta\mathbf{x}_i, \tag{2.13}$$

where μ_i is an appropriately chosen step size, e.g., by backtracking Armijo line search, and $\Delta\mathbf{x}_i$ is the search direction given by

$$\Delta\mathbf{x}_i = \|\nabla f(\mathbf{x}_i)\|_{\mathbf{M}_i,*}\mathbf{d}_i, \tag{2.14}$$

$$\text{with} \quad \mathbf{d}_i = \arg\min_{\mathbf{d}} \nabla f(\mathbf{x}_i)^\top \mathbf{d} \quad \text{s.t.} \quad \|\mathbf{d}\|_{\mathbf{M}_i} = 1. \tag{2.15}$$

Here $\mathbf{M}_i$ is a symmetric positive definite matrix, $\|\cdot\|_{\mathbf{M}_i,*}$ is the dual norm to $\|\cdot\|_{\mathbf{M}_i}$, and $\mathbf{d}_i$ is the steepest descent direction over the unit ball induced by $\|\cdot\|_{\mathbf{M}_i}$. Because $\mathbf{d}_i$ has unit length, it is re-scaled using (2.14). Since the dual norm and $\mathbf{d}_i$ have closed form representations $\|\mathbf{y}\|_{\mathbf{M}_i,*} = \|\mathbf{y}\|_{\mathbf{M}_i^{-1}}$ and $\mathbf{d}_i = \mathbf{M}_i^{-1}\nabla f(\mathbf{x}_i)/\|\nabla f(\mathbf{x}_i)\|_{\mathbf{M}_i^{-1}}$, respectively, the search direction is given by

$$\Delta\mathbf{x}_i = \mathbf{M}_i^{-1}\nabla f(\mathbf{x}_i). \tag{2.16}$$

We see from (2.15) that the search direction is induced by the metric defined by $\mathbf{M}_i$. For instance, the choice of the identity matrix corresponds to the Euclidean metric and gives the gradient descent direction $\Delta\mathbf{x}_i = -\nabla f(\mathbf{x}_i)$. Although the gradient descent method is very simple, it only uses first-order (gradient) information and has linear convergence under standard assumptions [14, Chapter 9.3]. Newton's method adaptively incorporates local second-order (curvature) information in each iteration and has superior convergence. In particular, it uses $\mathbf{M}_i = \nabla^2 f(\mathbf{x}_i)$, which induces the Hessian metric and generates the Newton search direction $\Delta\mathbf{x}_i = -(\nabla^2 f(\mathbf{x}_i))^{-1}\nabla f(\mathbf{x}_i)$. An outline of the standard Newton's method is given in Algorithm 4. The standard Newton's method with step size $\mu_i = 1$ attains local quadratic convergence to a local minimum $\mathbf{x}^*$ under the following assumptions [84, Theorem 2.3.3]:

Algorithm 4: A Standard Line Search Newton's Method

```
1: initialize $\mathbf{x}_0$
2: for $i = 1, 2, \ldots$ do
3:    evaluate $f(\mathbf{x}_i)$, $\nabla f(\mathbf{x}_i)$, and $\nabla^2 f(\mathbf{x}_i)$
4:    compute the search direction $\Delta\mathbf{x}_i = -(\nabla^2 f(\mathbf{x}_i))^{-1}\nabla f(\mathbf{x}_i)$
5:    set $\mathbf{x}_{i+1} = \mathbf{x}_i + \mu_i\Delta\mathbf{x}_i$, where $\mu_i$ is determined by a line search scheme
6:    if stopping criteria are satisfied then
7:       break
8:    end if
9: end for
10: Output: approximate solution $\mathbf{x}_{i+1}$
```

1. f is twice differentiable and $\nabla^2 f$ is Lipschitz continuous,
2. $\nabla f(\mathbf{x}^*) = 0$, and
3. $\nabla^2 f(\mathbf{x}^*)$ is positive definite.

Newton-Krylov Methods We note that the standard Newton search direction solves the Newton equation

$$\nabla^2 f(\mathbf{x}_i)\Delta\mathbf{x}_i = -\nabla f(\mathbf{x}_i). \tag{2.17}$$

For large problem sizes, it is not economical to solve (2.17) exactly or even build the Hessian matrix. In addition, an accurate solution to (2.17) might not be necessary since the local approximation model (2.15) used to derive the Newton iterations might not estimate f well, especially when the current iterate $\mathbf{x}_i$ is far away from the optimal solution. It is therefore practical to solve the Newton equation approximately. Newton-Krylov methods apply Krylov subspace methods to (2.17) and stop the iterations at some appropriate solution to the system. Commonly, the iterations are terminated when the norm of the residue $\mathbf{r}_i = \nabla^2 f(\mathbf{x}_i)\Delta\mathbf{x}_i + \nabla f(\mathbf{x}_i)$ is small enough or when a negative curvature is encountered (for the case of indefinite Hessian). Since a Krylov subspace method is used, it does not require access to entries of the Hessian

Algorithm 5: A General Newton-Krylov Method

```
1: initialize $\mathbf{x}_0$
2: for $i = 1, 2, \ldots$ do
3:    evaluate $f(\mathbf{x}_i)$, $\nabla f(\mathbf{x}_i)$, and build functions to compute $\mathbf{y} \mapsto \nabla^2 f(\mathbf{x}_i)\mathbf{y}$
4:    approximately solve the Newton equation (2.17) using Krylov subspace
      methods to obtain the search direction $\Delta\mathbf{x}_i$
5:    set $\mathbf{x}_{i+1} = \mathbf{x}_i + \mu_i \Delta\mathbf{x}_i$, where $\mu_i$ is determined by a line search scheme
6:    if stopping criteria are satisfied then
7:       break
8:    end if
9: end for
10: Output: approximate solution $\mathbf{x}_{i+1}$
```

but routines to perform matrix-vector products with the Hessian and is thus applicable to large-scale problems. An outline for a general Newton-Krylov method is given in Algorithm 5. Examples of the scheme include the Newton-CG method, which applies conjugate gradient (CG) method to (2.17). Other Krylov subspace methods are also applied, e.g., Lanczos tridiagonalization [83, 117] and Arnoldi method [15, 120]. Newton-Krylov methods with step size $\mu_i = 1$ attains local superlinear convergence to a local minimum $\mathbf{x}^*$ under the following assumptions [84, Theorem 2.5.2]:

1. f is twice differentiable and $\nabla^2 f$ is Lipschitz continuous,
2. $\nabla f(\mathbf{x}^*) = 0$,
3. $\nabla^2 f(\mathbf{x}^*)$ is positive definite, and
4. the iterations of the Krylov-subspace method are terminated when the norm of the residue $\|\mathbf{r}_i\|_2 \leq \eta_i \|\nabla f(\mathbf{x}_i)\|_2$, where $0 \leq \eta_i \leq \eta < 1$ and $\lim_{i\to\infty} \eta_i = 0$.

Modified Newton's Methods In some applications, the Hessian matrix is not positive definite or is rank-deficient. In the former case, the search direction might not be a descent direction, and in the latter case, the Newton equation (2.17) can be inconsistent. In modified Newton's method, this is overcome by adding a shift

Algorithm 6: A Modified Newton's Method

```
1: initialize $\mathbf{x}_0$
2: for $i = 1, 2, \ldots$ do
3:     evaluate $f(\mathbf{x}_i)$, $\nabla f(\mathbf{x}_i)$, and $\nabla^2 f(\mathbf{x}_i)$
4:     compute $\tau_i$ such that $\mathbf{B}_i = \nabla^2 f(\mathbf{x}_i) + \tau_i \mathbf{I}$ is sufficiently positive definite
5:     solve $\mathbf{B}_i \Delta\mathbf{x}_i = \nabla f(\mathbf{x}_i)$
6:     set $\mathbf{x}_{i+1} = \mathbf{x}_i + \mu_i \Delta\mathbf{x}_i$, where $\mu_i$ is determined by a line search scheme
7:     if stopping criteria are satisfied then
8:         break
9:     end if
10: end for
11: Output: approximate solution $\mathbf{x}_{i+1}$
```

to the Hessian matrix so that it is sufficiently positive definite. The choice of the shift is pivotal to determining the search direction of the scheme. An idea pertinent to our numerical experiments and perhaps the simplest is to add a multiple of the identity so that the smallest eigenvalue is bounded away from zero. An example of the line search modified Newton's method is presented in Algorithm 6. The global convergence of Algorithm 6 to a stationary point is guaranteed under the following assumptions [122, Theorem 3.8]:

1. f is twice continuously differentiable,
2. the level set $\{\mathbf{x} : f(\mathbf{x}) \le f(\mathbf{x}_0)\}$ is compact, and
3. $\text{cond}(\mathbf{B}_i) \le c$ for some $c > 0$ and all for all $i = 0, 1, 2, ...$

2.3 Machine Learning

We conclude this chapter by describing supervised classification problems that will be used in our numerical experiments. We begin by introducing the problem and the random feature model. We then formulate the linear regression and multinomial logistic regression problems. We finally describe two image classification datasets.

Supervised Classification Problems Let n_f be the number of features, n_c be the number of classes, and Δ_{n_c} be the n_c-dimensional unit simplex. We denote the set of data (not necessarily with finite cardinality) by $\mathcal{D} \subset \mathbb{R}^{n_f} \times \Delta_{n_c}$, where each element $(\mathbf{y}, \mathbf{c}) \in \mathcal{D}$ is a pair of input feature and target output label.

We partition $\mathcal{D}$ into a training set $\mathcal{D}_{\text{train}}$, a validation set $\mathcal{D}_{\text{val}}$, and a test set $\mathcal{D}_{\text{test}}$. The goal of supervised classification is to obtain a classifier by utilizing the information from the training and validation sets so that it can *generalize well*, i.e., it accurately predicts the label for the test set. Specifically, the training set is directly used in the optimization. The validation set is used to gauge the generalization of the classifier and tune parameters through cross-validation, and the test set is not used in training but only in performance evaluation.

Random Feature Models The input features $\mathbf{y}$ are often embedded into another space which can better describe the input-output relationship. This mapping $\mathbf{y} \mapsto \mathbf{a}(\mathbf{y})$ is known as feature extraction. Here we review a feature extraction technique called the random feature model (RFM) [131], also known as extreme learning machines [79]. RFM propagates the features into a higher dimensional space $\mathbb{R}^m$ by applying a random nonlinear transformation

$$\mathbf{a}_{\text{RFM}}(\mathbf{y}) = \sigma(\mathbf{Z}\mathbf{y} + \mathbf{b}), \tag{2.18}$$

where σ is an element-wise nonlinear activation function, $\mathbf{Z} \in \mathbb{R}^{m \times n_f}$ and $\mathbf{b} \in \mathbb{R}^m$ are randomly generated. For an appropriately chosen m, improved generalizability is empirically obtained [104, 82].

Problem setup We target to obtain a classifier $\mathbf{s}(\cdot; \mathbf{W}) : \mathbb{R}^m \to \mathbb{R}^{n_c}$ such that $\mathbf{s}(\mathbf{a}(\mathbf{y}); \mathbf{W}) \approx \mathbf{c}$, where $\mathbf{W}$ is the model parameters and $\mathbf{a}$ can be any feature extractors, e.g., RFM or hidden layers of a neural network [119, 120]. To this end, we consider the sample average approximation (SAA) [89, 118, 87] of a training problem formulated as

$$\min_{\mathbf{W}} \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} L(\mathbf{s}(\mathbf{a}(\mathbf{y});\mathbf{W}),\mathbf{c}). \tag{2.19}$$

Here, L is a loss function measuring the discrepancy between the classifier and the target output, and $\mathcal{T}$ is a finite subset of the training set $\mathcal{D}_{\text{train}}$. We consider two major types of training problems. The first type is the linear regression problem which uses a linear classifier $\mathbf{s}_{\text{LR}}(\mathbf{a}(\mathbf{y});\mathbf{W}) = \mathbf{W}\mathbf{a}(\mathbf{y})$ and squared loss $L_{\text{LR}}(\mathbf{s},\mathbf{c}) = \frac{1}{2}\|\mathbf{s}-\mathbf{c}\|_2^2$. It is expressed as

$$\min_{\mathbf{W}} \sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} \frac{1}{2|\mathcal{T}|}\|\mathbf{W}\mathbf{a}(\mathbf{y})-\mathbf{c}\|_2^2. \tag{2.20}$$

The second type is the multinomial logistic regression (MLR). It uses the softmax function $\mathbf{s}_{\text{MLR}}(\cdot;\mathbf{W}) : \mathbb{R}^m \to \Delta_{n_c}$ as its classifier, which is given by

$$\mathbf{s}_{\text{MLR}}(\mathbf{a}(\mathbf{y});\mathbf{W}) = \frac{\exp(\mathbf{W}\mathbf{a}(\mathbf{y}))}{\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}(\mathbf{y}))}. \tag{2.21}$$

Here $\mathbf{1}_{n_c} \in \mathbb{R}^{n_c}$ is the vector of all ones, Δ_{n_c} is the n_c-dimensional unit simplex, and the exponential function is applied element-wise. It uses the cross-entropy loss function $L_{\text{MLR}}(\mathbf{s},\mathbf{c}) = -\mathbf{c}^\top \log \mathbf{s}$ and is formulated as

$$\begin{aligned}
&\min_{\mathbf{W}} -\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} \mathbf{c}^\top \log\left(\frac{\exp(\mathbf{W}\mathbf{a}(\mathbf{y}))}{\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}(\mathbf{y}))}\right) \\
=&\min_{\mathbf{W}} -\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} \left[(\mathbf{c}^\top\mathbf{1}_{n_c})\log\left(\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}(\mathbf{y}))\right) - \mathbf{c}^\top\mathbf{W}\mathbf{a}(\mathbf{y})\right] \\
=&\min_{\mathbf{W}} -\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} \left[\log\left(\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}(\mathbf{y}))\right) - \mathbf{c}^\top\mathbf{W}\mathbf{a}(\mathbf{y})\right].
\end{aligned} \tag{2.22}$$

Here we use the fact that $\mathbf{c}^\top\mathbf{1}_{n_c} = 1$ since $\mathbf{c} \in \Delta_{n_c}$. A derivation of the gradient and Hessian for the MLR problem is provided in Appendix A.

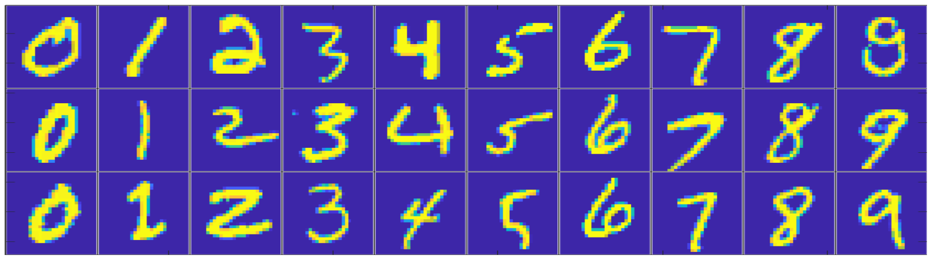


Figure 2.1: Example images from the MNIST dataset

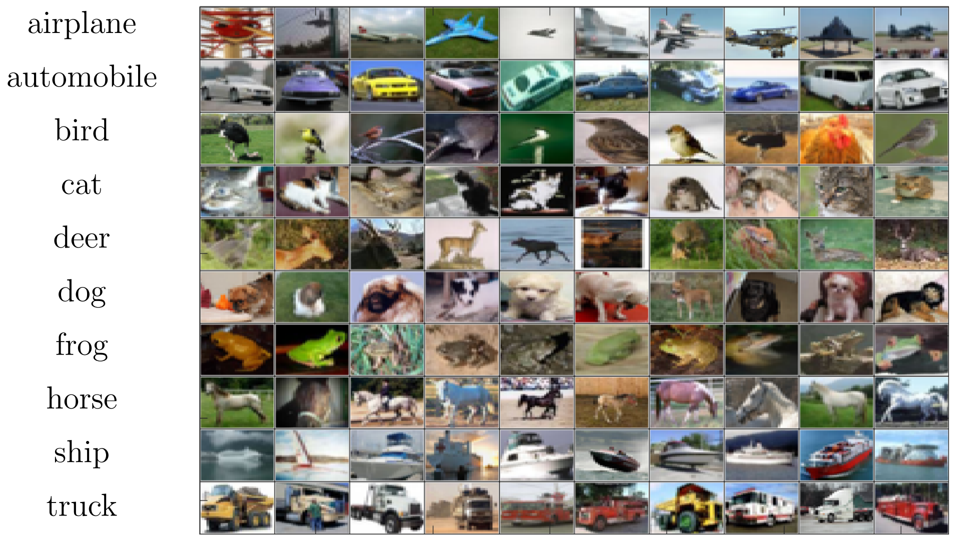


Figure 2.2: Example images from the CIFAR-10 dataset

Datasets In this dissertation, we perform numerical experiments on supervised classification using two datasets. The first is the MNIST dataset which consists of $60,000$ 28×28 hand-written images for digits from 0 to 9. The second one is the CIFAR-10 dataset which consists of $60,000$ 32×32 color images equally distributed for the following ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Example images for the two datasets are shown in Figure 2.1 and Figure 2.2, respectively.

Chapter 3

A Projected Newton-Krylov Method for Large-Scale Bound-Constrained Optimization

The work in this chapter is based on [83] and was done in collaboration with Samy Wu Fung and Lars Ruthotto. This chapter presents PNKH-B, a projected Newton-Krylov method to efficiently solve large-scale bound-constrained optimization problems. In particular, PNKH-B is designed to handle situations in which function and gradient evaluations are expensive, and the (approximate) Hessian is only available through matrix-vector products. In each iteration, PNKH-B constructs a low-rank approximation to the (approximate) Hessian to induce the search direction and projection. The key contribution in this work is a projection metric defined using the low-rank approximation and an interior point implementation which effectively exploits the low-rank structure of the (approximate) Hessian to solve the projection problem.

This chapter is organized as follows. Firstly, we describe the general problem setup. Secondly, we review related projected Newton methods to motivate our approach. Thirdly, we give an outline for our method PNKH-B. Fourthly, we present the derivation and implementation of the interior point method. Fifthly, we provide a global convergence proof of our method. Lastly, we conclude the chapter with three numerical experiments.

3.1 Problem Description

In this work, our goal is to approximately solve large-scale bound-constrained optimization problems formulated as

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \tag{3.1}$$

Here, $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, the inequalities are applied component-wise, and the vectors $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n \cup \{\pm\infty\}$ with $\mathbf{l} \leq \mathbf{u}$ define the bound constraints. To be precise, we focus on solving large-scale problems in which evaluating the objective function f, and its gradient $\nabla_{\mathbf{x}} f$ are computationally expensive, and its (approximate) Hessian is only available through matrix-vector products. This is common in problems like PDE parameter estimation [36, 46, 64, 107, 134, 149, 152], image processing [19, 76, 77, 78, 145, 146, 150], neural networks [21, 153, 67, 65, 66, 133], etc. PNKH-B targets to approximately solve these problems with as few objective function and gradient evaluations and Hessian-vector products as possible, due to their costly computations.

3.2 Related Work and Motivation

Projected inexact Newton and quasi-Newton methods are among the most effective and popular solvers for problems of type (3.1). Generally, their kth iteration reads

$$\mathbf{x}_{k+1} = \Pi_{\|\cdot\|_{\mathbf{P}_k}} (\mathbf{y}_{k+1}), \quad \text{with} \quad \mathbf{y}_{k+1} = \mathbf{x}_k - \mu_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k), \tag{3.2}$$

where $\mathbf{H}_k$ is a low-rank approximation to the (approximate) Hessian, μ_k is a step size, and the projection matrix $\mathbf{P}_k$ induces $\Pi_{\|\cdot\|_{\mathbf{P}_k}}$, the projection operator onto the bound constraints. More specifically,

$$\Pi_{\|\cdot\|_{\mathbf{P}_k}}(\mathbf{y}) = \arg\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{z}-\mathbf{y}\|^2_{\mathbf{P}_k}, \quad \text{subject to} \quad \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}. \tag{3.3}$$

We slightly abuse notation and denote the pseudoinverse of $\mathbf{H}_k$ by $\mathbf{H}_k^{-1}$ in order to be consistent with the conventional notation used in Newton's method. The (semi-)norm induced by the matrix $\mathbf{P}_k$ is defined by $\|\mathbf{v}\|_{\mathbf{P}_k} := \sqrt{\mathbf{v}^\top \mathbf{P}_k \mathbf{v}}$ for all $\mathbf{v} \in \mathbb{R}^n$. We also refer to pseudometric as metric for simplicity of exposition in the following discussion.

One property that can be used to group existing schemes is the metric used to determine the search direction and the projection. We refer to schemes that use the same metric for both steps as one-metric schemes and schemes that use different metrics for the two steps as two-metric schemes. Another distinguishing feature is the order in which projections and line searches are performed. Schemes that project each line search iterate in general lead to a piecewise linear arc, while applying the projection only once before the line search yields a linear arc. In the following, we review existing methods according to these choices and provide an example to highlight their differences.

An extensively studied two-metric scheme [10, 47] uses a search direction induced by an approximated Hessian norm and a projection with respect to the Euclidean metric. That is, its expression in each iteration is given by (3.2) and (3.3) with $\mathbf{P}_k = \mathbf{I}$, the identity matrix. In this setting, the projection (3.3) admits a simple closed-form solution

$$\Pi_{\|\cdot\|_{\mathbf{I}}}(\mathbf{y}) = \Pi_{\|\cdot\|_2}(\mathbf{y}) = \max\{\min\{\mathbf{y}, \mathbf{u}\}, \mathbf{l}\}. \tag{3.4}$$

Also, its line search induces a piecewise linear arc, see Figure 3.1. Despite the low computational cost of the projection problem, its convergence is not guaranteed in general, see Example 1, [10] and [84, Chapter 5.5.1]. The global convergence of this approach for convex problems with linear constraints was proven in [10, 47] when a variable partitioning scheme is used. It partitions the components of the kth iterate into an active set in which the components are at or close to the boundary of the feasible set and an inactive set in which the components are in the interior of the feasible set. A search direction induced by the Euclidean norm is used for the active components, and a search direction induced by $\|\cdot\|_{\mathbf{H}_k}$ is used for the inactive components. They prove local superlinear convergence under certain conditions. Since then, variable partitioning has become a recurring theme for two-metric schemes [64, 73, 85, 95, 96, 97, 137]. Although the projection of the convergent two-metric scheme (3.4) can be computed immediately, it requires appropriate scaling for the Euclidean norm induced search direction before combining the two search directions. Moreover, when many constraints are active, two-metric schemes essentially become projected gradient methods. A specific drawback in Newton-Krylov schemes for large-scale problems is that the partitioning of the variables complicates the design of effective preconditioners. Given a preconditioner $\mathbf{M}_k$ for the Hessian $\nabla^2 f(\mathbf{x}_k)$ and $\mathbf{Q}_k$ the projection operator onto the inactive set at the kth step, the most natural choice is to precondition $\mathbf{Q}_k^\top \nabla^2 f(\mathbf{x}_k)\mathbf{Q}_k$ by $\mathbf{Q}_k^\top \mathbf{M}_k^{-1}\mathbf{Q}_k$. Since it is intractable to compute $(\mathbf{Q}_k^\top \mathbf{M}_k^{-1}\mathbf{Q}_k)^{-1}$, one might use the approximation $\mathbf{Q}_k^\top \mathbf{M}_k \mathbf{Q}_k$. However, note that $(\mathbf{Q}_k^\top \mathbf{M}_k^{-1}\mathbf{Q}_k)^{-1} \neq \mathbf{Q}_k^\top \mathbf{M}_k \mathbf{Q}_k$ in general.

Another well-studied approach is one-metric methods with a linear arc. Generally, it is performed as follows. At the kth iteration, it (approximately) solves (3.2) with $\mu_k = 1$ and $\mathbf{P}_k = \mathbf{H}_k$ to obtain a projection. Then it performs a line search along the straight line connecting the current iterate $\mathbf{x}_k$ and the projection; this linear line search is done in order to limit the number of solving costly projections. This scheme is studied with different approximations of the Hessian, solvers for the projection (3.2) with $\mu_k = 1$ or backtracking schemes to determine the next iterate $\mathbf{x}_{k+1}$. For instance, the widely-applied L-BFGS-B [16, 113] uses a limited-memory BFGS matrix for $\mathbf{H}_k$ and approximately solves the projection (3.2) with $\mu_k = 1$ without any constraints, then it truncates the path toward the solution in order to satisfy the constraints. Finally, it backtracks along the straight line to obtain $\mathbf{x}_{k+1}$. Other variants of this one-metric method with linear arc include [5, 6, 68, 100, 136]. Although the consistent choice of metric could generate a better update direction than the two-metric scheme, it results in a suboptimal iterate which does not lie in the boundary whenever a step size of 1 is not used.

Inspired by projected variable metric methods, our PNKH-B takes the form (3.2) and (3.3). It constructs $\mathbf{H}_k$, a low-rank approximation to the (approximate) Hessian, using Lanczos tridiagonalization to compute a basis to the Krylov subspace defined by the (approximation) Hessian and gradient at the kth step. It uses $\mathbf{P}_k = \tilde{\mathbf{H}}_k$, where $\tilde{\mathbf{H}}_k$ is a symmetric positive definite matrix that equals to $\mathbf{H}_k$ on the Krylov subspace. This renders the projection problem well-defined. To be more specific, the kth iteration of PNKH-B reads

$$\mathbf{x}_{k+1} = \Pi_{\|\cdot\|_{\tilde{\mathbf{H}}_k}}\left(\mathbf{y}_{k+1}\right), \quad \text{with} \quad \mathbf{y}_{k+1} = \mathbf{x}_k - \mu_k \mathbf{H}_k^{-1}\nabla f(\mathbf{x}_k), \tag{3.5}$$

$$\text{and} \quad \Pi_{\|\cdot\|_{\tilde{\mathbf{H}}_k}}(\mathbf{y}) = \arg\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{z}-\mathbf{y}\|^2_{\tilde{\mathbf{H}}_k}, \quad \text{subject to} \quad \mathbf{l} \le \mathbf{z} \le \mathbf{u}. \tag{3.6}$$

PNKH-B is a generalized one-metric (or generalized variable metric) method because the variable metric used to determine the search direction and the projection are equivalent on the subspace spanned by $\mathbf{H}_k$. In each line search, that is, for each attempted value of μ_k, it is required to solve the quadratic projection problem Equation (3.6), which has no closed form solution. However, thanks to the low-rank structure of $\mathbf{H}_k$ and an effective implementation of an interior point method, the computational overhead for the projection is negligible; see Section 3.3.2. Due to the consistent choice of metrics for the search direction and projection and an adaptive piecewise linear line search, PNKH-B can generate a better iterate when compared to the existing approaches; see Figure 3.1.

Example 1. *We illustrate the differences between one-metric and two-metric schemes with linear and piecewise linear arcs, respectively, using a two-dimensional quadratic program*

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{b}^\top \mathbf{x} \quad \textit{subject to} \quad \mathbf{l} \le \mathbf{x} \le \mathbf{u}. \tag{3.7}$$

The first iteration before projection of the one-metric method with piecewise linear arc reads

$$\mathbf{y}_1(\mu) = \mathbf{x}_0 - \mu\mathbf{H}^{-1}\nabla f(\mathbf{x}_0) = (1-\mu)\mathbf{x}_0 - \mu\mathbf{H}^{-1}\mathbf{b},$$

where μ is a step size determined by a backtracking line search scheme. The projection with the Hessian metric is given by

$$\Pi_{\|\cdot\|_{\mathbf{H}}}(\mathbf{y}_1(\mu)) = \operatorname*{arg\,min}_{\mathbf{l}\le\mathbf{z}\le\mathbf{u}} \frac{1}{2}\mathbf{z}^\top\mathbf{H}\mathbf{z} - (1-\mu)\mathbf{z}^\top\mathbf{H}\mathbf{x}_0 + \mu\mathbf{b}^\top\mathbf{z}. \tag{3.8}$$

When $\mu = 1$, i.e., the first step of the backtracking line search, the projection problem is equivalent to the original optimization problem. So the backtracking line search stops at the first step, and the one-metric method with the piecewise linear line search converges in one iteration. This is because the Hessian metric projection is consistent with the steepest descent direction $\mathbf{H}^{-1}\nabla f$ induced by the Hessian metric. If for a non-quadratic objective function, the initial step size is not accepted, then the piecewise linear and linear arc lead to different iterates; see Figure 3.1. Solving the projection problem with the Euclidean metric leads to a suboptimal projection at which the scheme stagnates in the absence of any of the remedies outlined above.

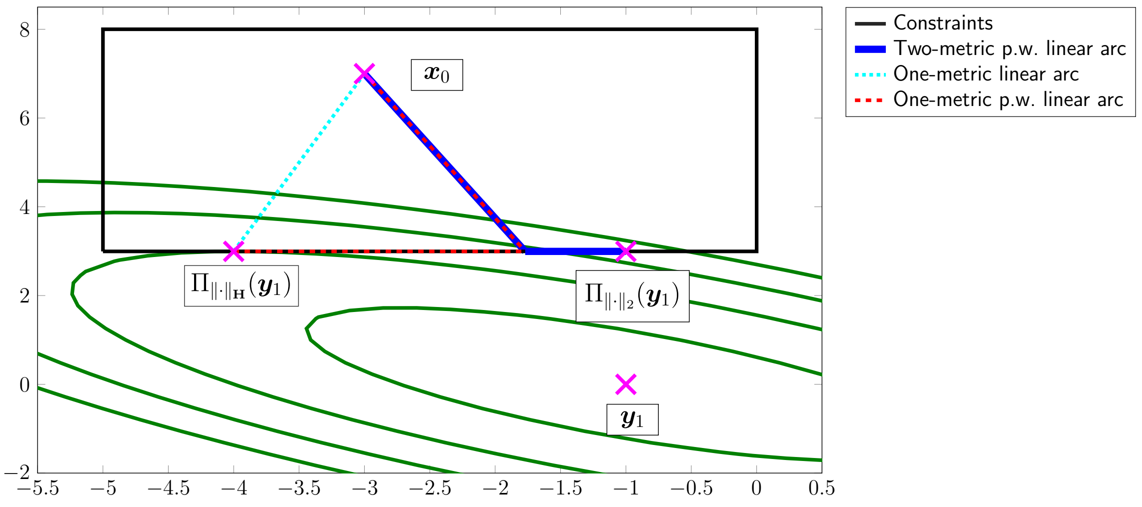


Figure 3.1: Figure description: An illustration of the first iterations of different methods on the quadratic optimization problem (3.7), where $\mathbf{H} = [1, 1; 1, 2]$, $\mathbf{b} = [1; 1]$, $\mathbf{l} = [-5; 3]$, $\mathbf{u} = [0; 8]$, $\mathbf{x}_0 = [-3; 7]$, $\mathbf{y}_1 = \mathbf{x}_0 - \mathbf{H}^{-1}\nabla f(\mathbf{x}_0) = -\mathbf{H}^{-1}\mathbf{b} = [-1; 0]$ is the updated variable before projection, $\Pi_{\|\cdot\|_{\mathbf{H}}}(\mathbf{y}_1) = [-4; 3]$ is the projection with the Hessian metric and is the optimal solution, and $\Pi_{\|\cdot\|_2}(\mathbf{y}_1) = [-1; 3]$ is the projection with the Euclidean metric. The linear/piecewise linear line search arcs for one/two-metric methods are shown. The one-metric piecewise linear arc, which is used in our proposed PNKH-B, is the best one as it searches along the boundary and gives the optimal solution. The one-metric linear arc is less natural, does not search along the boundary, and gives suboptimal iterate whenever step size 1 is not used. Finally, the two-metric piecewise linear arc searches for the opposite direction of the one-metric piecewise linear arc. It gives a suboptimal iterate $\Pi_{\|\cdot\|_2}(\mathbf{y}_1)$ and it will be stuck at $\Pi_{\|\cdot\|_2}(\mathbf{y}_1)$ even when the exact Hessian is used, i.e., it generates $\Pi_{\|\cdot\|_2}(\mathbf{y}_k) = \Pi_{\|\cdot\|_2}(\mathbf{y}_1)$ for all $k \geq 2$.

3.3 PNKH-B

We describe the proposed PNKH-B in this section. In Section 3.3.1, we present an outline of the algorithm. At each iteration, each backtracking line search requires computing a projection, which is a quadratic program. In Section 3.3.2, we present the derivation and implementation of an interior point method to solve the quadratic program effectively. In Section 3.3.3, we present two variants of our PNKH-B, which incorporate the current estimates of the active set.

3.3.1 Outline of PNKH-B

PNKH-B is a projected Newton-Krylov method with a low-rank approximated Hessian metric geared toward large-scale bound-constrained optimization problems. It is a generalized one-metric scheme whose iteration is given by (3.5) and (3.6).

The global convergence of PNKH-B is guaranteed under standard assumptions; see Section 3.4. We set $\mathbf{H}_k$ as a low-rank approximation of the (approximate) Hessian at $\mathbf{x}_k$ generated by Lanczos tridiagonalization [94]. Specifically, the Krylov subspace is defined by the (approximate) Hessian and gradient at $\mathbf{x}_k$. Therefore, the search direction obtained from the rank-l approximation is equal to that obtained using l steps of the conjugate gradient method (CG) up to roundoff errors. However, storing the low-rank Hessian approximation allows its re-use to compute the projection. The low-rank approximation is given by $\mathbf{H}_k = \mathbf{V}_k \mathbf{T}_k \mathbf{V}_k^\top$, where $\mathbf{V}_k \in \mathbb{R}^{n \times l}$ has orthonormal columns, $\mathbf{T}_k \in \mathbb{R}^{l \times l}$ is tridiagonal and l is the rank of the low-rank approximation. We slightly abuse notation and denote the pseudoinverse $\mathbf{V}_k \mathbf{T}_k^{-1} \mathbf{V}_k^\top$ by $\mathbf{H}_k^{-1}$ in order to be consistent with the conventional notation used in Newton's method. The positive definite matrix $\tilde{\mathbf{H}}_k$ used to define the projection norm is obtained by applying a shift in the orthogonal complement of the Krylov subspace to $\mathbf{H}_k$. Lanczos tridiagonalization is suitable for large-scale problems because it does not require the explicit (approximate) Hessian $\mathbf{G}_k$, but only the function $g_k : \mathbf{y} \mapsto \mathbf{G}_k\mathbf{y}$. Using the low-rank approximation, we effectively compute the pseudoinverse $\mathbf{H}_k^{-1}$ and the projection $\Pi_{\|\cdot\|_{\tilde{\mathbf{H}}_k}}(\cdot)$, which has to be done once for each line search. The projection problem is solved using an interior point method, which exploits the low-rank approximation effectively and scales only linearly with the number of variables. The interior-point method will be discussed in Section 3.3.2. The outline of our PNKH-B is summarized in Algorithm 7.

Although not shown here, our method can be straightforwardly extended to other low-rank representations, e.g., arising in L-BFGS [16, 33, 53]. However, we focus on inexact Newton methods because we aim to solve large-scale problems using as few evaluations as possible, and quasi-Newton methods generally require more iterations to convergence than inexact Newton methods [64, Chapter 6].

3.3.2 Interior Point Method

In this section, we present the derivation and effective implementation of the interior point method tailored to exploit the low-rank structure in (3.6).

Derivation We use a standard primal-dual interior point method to solve the projection problem (3.6), which we derive following the outline in [122, Chapter 16.6]. To obtain $\mathbf{x}_{k+1}$, we re-formulate (3.6) as

$$\min_{\mathbf{z},\mathbf{w}} \frac{1}{2}\mathbf{z}^\top \tilde{\mathbf{H}}_k \mathbf{z} - \mathbf{z}^\top \tilde{\mathbf{H}}_k \mathbf{y}_{k+1} \quad \text{subject to} \quad \mathbf{K}\mathbf{z} - \mathbf{b} = \mathbf{w} \text{ and } \mathbf{w} \geq \mathbf{0},$$

where $\tilde{\mathbf{H}}_k = \mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^\top + c\mathbf{U}_k\mathbf{U}_k^\top$ is the low-rank approximation of the (approximate) Hessian plus a small shift $c > 0$ in the orthogonal complement spanned by

Algorithm 7: Outline of PNKH-B for solving (3.1)

```
1: Inputs: Initial guess $\mathbf{x}_0 \in C$, tolerance xtol and gtol, line search parameter
   $\alpha \in (0, 1)$, and the rank of the low-rank approximation $l$
2: for $k = 0, 1, 2, \ldots$ do
3:     select estimate of active set, $\mathcal{A}_k \in \{\emptyset, \mathcal{A}_k^{\text{bound}}, \mathcal{A}_k^{\text{aug}}\}$, and build projection
       matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$.
4:     compute $f(\mathbf{x}_k)$, $\nabla f(\mathbf{x}_k)$ and (approximate) Hessian $\mathbf{G}_k \approx \nabla^2 f(\mathbf{x}_k)$
5:     compute the Lanczos tridiagonalization $\mathbf{F}_k = \mathbf{V}_k \mathbf{T}_k \mathbf{V}_k^\top \approx \mathbf{Q}_k \mathbf{G}_k \mathbf{Q}_k^\top$ with
       initial vector $-\mathbf{Q}_k \nabla f(\mathbf{x}_k)$ (using matrix-free implementation)
6:     compute the Hessian approximation $\mathbf{H}_k$ in (3.16) and the search direction
       $-\mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k)$
7:     set $\mu = 1$
8:     for $i = 0, 1, 2, \ldots$ do
9:         solve the projection $\mathbf{x}_t = \Pi_{\|\cdot\|_{\tilde{\mathbf{H}}_k}}(\mathbf{x}_k - \mu \mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k))$ (see Section 3.3.2)
10:        if $f(\mathbf{x}_t) < f(\mathbf{x}_k) + \alpha \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_t - \mathbf{x}_k)$ then
11:            set $\mathbf{x}_{k+1} = \mathbf{x}_t$ and break
12:        else
13:            set $\mu = \mu/2$
14:        end if
15:    end for
16:    if $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2 / \|\mathbf{x}_k\|_2 <$ xtol or norm of projected gradient $<$ gtol then
17:        break
18:    end if
19:    if $\mu = \mu_{\text{old}}$ then
20:        set $\mu = \min(1.5 * \mu, 1)$
21:    end if
22: end for
23: Output: approximate solution $\mathbf{x}_{k+1} \in C$.
```

$\mathbf{U}_k \in \mathbb{R}^{n \times (n-l)}$, $\mathbf{w} \in \mathbb{R}^{2n}$ is a slack vector and

$$\mathbf{K} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{l} \\ -\mathbf{u} \end{bmatrix}.$$

From this, we obtain the KKT conditions

$$\tilde{\mathbf{H}}_k\mathbf{z} - \tilde{\mathbf{H}}_k\mathbf{y}_{k+1} - \mathbf{K}^\top\boldsymbol{\lambda} = \mathbf{0}, \tag{3.9a}$$

$$\mathbf{K}\mathbf{z} - \mathbf{b} - \mathbf{w} = \mathbf{0}, \tag{3.9b}$$

$$w_i\lambda_i = 0, \text{ for } i = 1, ..., 2n, \tag{3.9c}$$

$$\mathbf{w} \geq \mathbf{0}\ ,\ \boldsymbol{\lambda} \geq \mathbf{0}, \tag{3.9d}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{2n}$ is a vector of Lagrange multipliers. Since the problem (3.6) is convex and the interior of the feasible set is non-empty, Slater's condition is satisfied, and hence the KKT conditions are necessary and sufficient. As usual in interior point methods, we consider the perturbed KKT conditions

$$F(\mathbf{z}, \mathbf{w}, \boldsymbol{\lambda}; \sigma, \xi) = \begin{bmatrix} \tilde{\mathbf{H}}_k\mathbf{z} - \tilde{\mathbf{H}}_k\mathbf{y}_{k+1} - \mathbf{K}^\top\boldsymbol{\lambda} \\ \mathbf{K}\mathbf{z} - \mathbf{b} - \mathbf{w} \\ \mathbf{W}\boldsymbol{\Lambda}\mathbf{e} - \sigma\xi\mathbf{e} \end{bmatrix} = \mathbf{0}, \tag{3.10}$$

where $\mathbf{W} = \operatorname{diag}(\mathbf{w})$, $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda})$, $\mathbf{e} \in \mathbb{R}^{2n}$ is a vector of ones, $\sigma \in [0, 1]$ and $\xi = \mathbf{w}^\top\boldsymbol{\lambda}/(2n)$ is the duality measure. The solutions of (3.10) define the central path and tend to the solution of (3.9) [122, Section 16.6]. We then apply Newton's method to find the root of the system (3.10). At the jth iteration of Newton's method, the step is obtained by solving

$$\begin{bmatrix} \tilde{\mathbf{H}}_k & 0 & -\mathbf{K}^\top \\ \mathbf{K} & -\mathbf{I} & 0 \\ 0 & \boldsymbol{\Lambda}_j & \mathbf{W}_j \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z}_j \\ \Delta\mathbf{w}_j \\ \Delta\boldsymbol{\lambda}_j \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_j \\ -\mathbf{v}_j \\ -\mathbf{W}_j\boldsymbol{\Lambda}_j\mathbf{e} + \sigma\xi_j\mathbf{e} \end{bmatrix}, \tag{3.11}$$

which is obtained by differentiating F in (3.10) and setting

$$\begin{aligned} \mathbf{r}_j &= \tilde{\mathbf{H}}_k \mathbf{z}_j - \tilde{\mathbf{H}}_k \mathbf{y}_{k+1} - \mathbf{K}^\top \boldsymbol{\lambda}_j, \\ \mathbf{v}_j &= \mathbf{K}\mathbf{z}_j - \mathbf{b} - \mathbf{w}_j. \end{aligned}$$

Here, $\mathbf{r}_j$ and $\mathbf{v}_j$ are the dual and primal residuals, respectively. After computing $\Delta\mathbf{z}_j$, $\Delta\mathbf{w}_j$ and $\Delta\boldsymbol{\lambda}_j$, the update of the interior point method is

$$(\mathbf{z}_{j+1}, \mathbf{w}_{j+1}, \boldsymbol{\lambda}_{j+1}) = (\mathbf{z}_j, \mathbf{w}_j, \boldsymbol{\lambda}_j) + \beta_j(\Delta\mathbf{z}_j, \Delta\mathbf{w}_j, \Delta\boldsymbol{\lambda}_j),$$

where $\beta_j = \min(\beta_j^{\text{pri}}, \beta_j^{\text{dual}})$ and

$$\begin{aligned} \beta_j^{\text{pri}} &= \max\{\beta \in (0,1] : \mathbf{w}_j + \beta\Delta\mathbf{w}_j \geq (1-\tau)\mathbf{w}_j\}, \\ \beta_j^{\text{dual}} &= \max\{\beta \in (0,1] : \boldsymbol{\lambda}_j + \beta\Delta\boldsymbol{\lambda}_j \geq (1-\tau)\boldsymbol{\lambda}_j\}. \end{aligned}$$

The parameter $\tau \in (0,1]$ controls the distance to the boundary of the feasible set. While there are other schemes to determine the step size (see, e.g., [31]), this simple choice has been effective in our experiments.

Efficient Implementation The most crucial step of the interior point method is the computation of the solution of the step in (3.11). Our implementation exploits the low-rank structure of $\mathbf{H}_k$ to directly solve the linear system with $\mathcal{O}(nl^2)$ floating point operations, where l is the rank of the low-rank approximation; see Algorithm 8 for an overview.

To compute the update, we first multiply the third equation of (3.11) by $\boldsymbol{\Lambda}_j^{-1}$ and add it to the second equation of (3.11) and obtain

$$\begin{bmatrix} \tilde{\mathbf{H}}_k & -\mathbf{K}^\top \\ \mathbf{K} & \boldsymbol{\Lambda}_j^{-1}\mathbf{W}_j \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z}_j \\ \Delta\boldsymbol{\lambda}_j \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_j \\ -\mathbf{v}_j - \mathbf{w}_j + \sigma\xi_j\boldsymbol{\Lambda}_j^{-1}\mathbf{e} \end{bmatrix}. \tag{3.12}$$

Multiplying the second equation of (3.12) by $\mathbf{K}^\top\mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j$ and adding it to the first equation, we obtain

$$(\tilde{\mathbf{H}}_k + \mathbf{K}^\top\mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j\mathbf{K})\Delta\mathbf{z}_j = -\mathbf{r}_j + \mathbf{K}^\top\mathbf{p}_j, \tag{3.13}$$

where $\mathbf{p}_j = \mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j(-\mathbf{v}_j - \mathbf{w}_j + \sigma\xi_j\boldsymbol{\Lambda}_j^{-1}\mathbf{e})$. Recall that $\tilde{\mathbf{H}}_k = \mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^\top + c\mathbf{U}_k\mathbf{U}_k^\top$ is the low-rank Hessian approximation plus a shift in the orthogonal complement of the Krylov subspace, where $c > 0$. This shift is only applied to the projection matrix but not the matrix of the search direction $\mathbf{H}_k^{-1}\nabla f(\mathbf{x}_k)$. This is because the inverse of the shift c^{-1} is large and will dominate the search direction. The right-hand side of (3.13) can be computed explicitly in $\mathcal{O}(n)$ operations. Since $\mathbf{U}_k\mathbf{U}_k^\top = \mathbf{I} - \mathbf{V}_k\mathbf{V}_k^\top$, we can compute $\tilde{\mathbf{H}}_k$ as

$$\tilde{\mathbf{H}}_k = \mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^\top + c\mathbf{U}_k\mathbf{U}_k^\top = \mathbf{V}_k(\mathbf{T}_k - c\mathbf{I}_l)\mathbf{V}_k^\top + c\mathbf{I}$$

without explicitly computing $\mathbf{U}_k$. Defining $\mathbf{E}_j = c\mathbf{I} + \mathbf{K}^\top\mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j\mathbf{K}$ and noticing that $\mathbf{E}_j$ is diagonal and invertible, we can use the Woodbury matrix identity to invert the left-hand side of (3.13). Specifically

$$\begin{aligned}(\tilde{\mathbf{H}}_k + \mathbf{E}_j)^{-1} &= (\mathbf{V}_k(\mathbf{T}_k - c\mathbf{I}_l)\mathbf{V}_k^\top + \mathbf{E}_j)^{-1} \\ &= \mathbf{E}_j^{-1} - \underbrace{\mathbf{E}_j^{-1}\mathbf{V}_k}_{\in\mathbb{R}^{n\times l}}\underbrace{((\mathbf{T}_k - c\mathbf{I}_l)^{-1} + \mathbf{V}_k^\top\mathbf{E}_j^{-1}\mathbf{V}_k)^{-1}}_{\in\mathbb{R}^{l\times l}}\underbrace{\mathbf{V}_k^\top\mathbf{E}_j^{-1}}_{\in\mathbb{R}^{l\times n}},\end{aligned} \tag{3.14}$$

where l is the rank of the low-rank approximation. From (3.14), we see that it requires

Algorithm 8: Interior Point Method for the Projection Problem (3.6)

```
1: Inputs: low-rank approximation $\mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^\top \approx \nabla^2 f(\mathbf{x}_k)$, point to be projected
   $\mathbf{y}_{k+1}$, initial guess $\mathbf{z}_0 \in \mathbb{R}^n$, $\boldsymbol{\lambda}_0, \mathbf{w}_0 \in \mathbb{R}^{2n}$, $\tau \in (0,1]$, tol $> 0$
2: for $j = 0, 1, 2, \dots$ do
3:    compute $\mathbf{r}_j = \tilde{\mathbf{H}}_k\mathbf{z}_j - \tilde{\mathbf{H}}_k\mathbf{y}_{k+1} - \mathbf{K}^\top\boldsymbol{\lambda}_j$
4:    compute $\mathbf{v}_j = \mathbf{K}\mathbf{z}_j - \mathbf{b} - \mathbf{w}_j$
5:    compute $\mathbf{p}_j = \mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j(-\mathbf{v}_j - \mathbf{w}_j + \boldsymbol{\Lambda}_j^{-1}\sigma\xi_j\mathbf{e})$
6:    compute $\mathbf{E}_j = c\mathbf{I} + \mathbf{K}^\top\mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j\mathbf{K}$
7:    compute
      $\Delta\mathbf{z}_j = \big(\mathbf{E}_j^{-1} - \mathbf{E}_j^{-1}\mathbf{V}_k((\mathbf{T}_k - c\mathbf{I}_l)^{-1} + \mathbf{V}_k^\top\mathbf{E}_j^{-1}\mathbf{V}_k)^{-1}\mathbf{V}_k^\top\mathbf{E}_j^{-1}\big)(-\mathbf{r}_j + \mathbf{K}^\top\mathbf{p}_j)$
8:    compute $\Delta\boldsymbol{\lambda}_j = \mathbf{p}_j - \mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j\mathbf{K}\Delta\mathbf{z}_j$
9:    compute $\Delta\mathbf{w}_j = \mathbf{K}\Delta\mathbf{z}_j + \mathbf{v}_j$
10:   compute $\beta_j = \min(\beta_\tau^{\mathrm{pri}}, \beta_\tau^{\mathrm{dual}})$, where
      $\beta_\tau^{\mathrm{pri}} = \max\{\beta \in (0,1] : \mathbf{w}_j + \beta\Delta\mathbf{w}_j \geq (1-\tau)\mathbf{w}_j\}$ and
      $\beta_\tau^{\mathrm{dual}} = \max\{\beta \in (0,1] : \boldsymbol{\lambda}_j + \beta\Delta\boldsymbol{\lambda}_j \geq (1-\tau)\boldsymbol{\lambda}_j\}$
11:   update the variables $(\mathbf{z}_{j+1}, \mathbf{w}_{j+1}, \boldsymbol{\lambda}_{j+1}) = (\mathbf{z}_j, \mathbf{w}_j, \boldsymbol{\lambda}_j) + \beta_j(\Delta\mathbf{z}_j, \Delta\mathbf{w}_j, \Delta\boldsymbol{\lambda}_j)$
12:   if $\|\mathbf{r}_j\|_2 <$ tol and $\|\mathbf{v}_j\|_2 <$ tol then
13:      break
14:   end if
15: end for
16: Output: $\mathbf{z}_{j+1}$ approximate projection of $\mathbf{y}_{k+1}$ onto $C$
```

$\mathcal{O}(nl^2)$ flops to compute the solution $\Delta\mathbf{z}_j$ of (3.13). After obtaining $\Delta\mathbf{z}_j$, we substitute $\Delta\mathbf{z}_j$ into (3.11) and (3.12) and obtain

$$\Delta\boldsymbol{\lambda}_j = \mathbf{p}_j - \mathbf{W}_j^{-1}\boldsymbol{\Lambda}_j\mathbf{K}\Delta\mathbf{z}_j, \quad \text{and} \quad \Delta\mathbf{w}_j = \mathbf{K}\Delta\mathbf{z}_j + \mathbf{v}_j,$$

whose computation require $\mathcal{O}(n)$ flops.

Overall, exploiting the structure of $\tilde{\mathbf{H}}_k$, the interior point method requires $\mathcal{O}(nl^2)$ flops per iteration, where n is the number of variables.

3.3.3 Incorporating Estimates of the Active Set

We introduce two variants of PNKH-B that seek to accelerate the convergence by using estimates of the active set. The intuitive idea is to ignore coordinate dimensions associated with constraints that are currently active during the construction of the low-rank approximation $\mathbf{H}_k$. To this end, we partition $\mathbf{x}_k$ into active and inactive components. To update the inactive coordinates, we exploit curvature information, and to update the active coordinates, we use a scaled projected gradient descent step. Our procedure and estimation of the active coordinates are essentially the same as in the two-metric schemes [10, 64, 137], which crucially rely on this step to ensure convergence. Being a generalized one-metric scheme, the convergence theory of PNKH-B applies both with and without partitioning. However, in practice it can be advantageous to use estimates of the active set because with variable partitioning, the active/inactive set information is incorporated into the low-rank approximation. Thus the search direction can capture more feasible directions.

At the kth iteration, let $\mathcal{A}_k \subset \{1, 2, \ldots, n\}$ contain the indices of the components that are estimated to be active and let $m = |\mathcal{A}_k|$. We denote with $\mathbf{Q}_k \in \mathbb{R}^{m\times n}$ and $\mathbf{R}_k \in \mathbb{R}^{(n-m)\times n}$ the projection operators onto the inactive and active set, respectively. For example, $\mathbf{R}_k$ can be constructed by selecting the rows of an identity matrix associated with $\mathcal{A}_k$. We shall discuss two common choices for constructing $\mathcal{A}_k$ below. Given $\mathbf{Q}_k$ and $\mathbf{R}_k$, the intermediate step in (3.5) is

$$\mathbf{y}_{k+1} = \mathbf{x}_k - \mu_k \left(\mathbf{Q}_k^\top \mathbf{F}_k^{-1} \mathbf{Q}_k \nabla f(\mathbf{x}_k) + \nu_k^{-1} \mathbf{R}_k^\top \mathbf{R}_k \nabla f(\mathbf{x}_k)\right). \tag{3.15}$$

Here, $\mathbf{F}_k$ is a rank-l approximation of the projected (approximate) Hessian $\mathbf{Q}_k \mathbf{G}_k \mathbf{Q}_k^\top$ and the constant $\nu_k > 0$ is used to balance the sizes of both steps. In practice, this number is often chosen based on the norm of the step for the inactive components. We set $\nu_k = \|\mathbf{R}_k \nabla f(\mathbf{x}_k)\|_\infty / \|\mathbf{F}_k^{-1} \mathbf{Q}_k \nabla f(\mathbf{x}_k)\|_\infty$ in our experiments. One can verify that this leads to the PNKH-B scheme with the Hessian approximation

$$\mathbf{H}_k = \begin{pmatrix} \mathbf{Q}_k^\top & \mathbf{R}_k^\top \end{pmatrix} \begin{pmatrix} \mathbf{F}_k & 0 \\ 0 & \nu_k \mathbf{I}_m \end{pmatrix} \begin{pmatrix} \mathbf{Q}_k \\ \mathbf{R}_k \end{pmatrix}. \tag{3.16}$$

We use the separability introduced by this construction in the projection, which decouples into using (3.4) on the active components and using the interior point method on the inactive components.

We obtain two variants of PNKH-B that differ only by the strategy to estimate active and inactive variables.

PNKH-B (boundary index) Perhaps the most straightforward estimate of the active set is to choose the components that are within an ϵ margin around the boundary, where $\epsilon > 0$, i.e.,

$$\mathcal{A}_k^{\text{bound}} = \{i \, : \, (\mathbf{x}_k)_i \leq l_i + \epsilon \;\text{ or }\; (\mathbf{x}_k)_i \geq u_i - \epsilon\}. \tag{3.17}$$

This choice has been used successfully in [64].

PNKH-B (augmented index) As an alternative active set estimation scheme, we use the one proposed in [10, 137]. The idea is that in addition to the ϵ margin, we consider the sign of the partial derivative so that curvature information is used for those constraints predicted to become inactive, i.e.,

$$\mathcal{A}_k^{\text{aug}} = \{i \;:\; [(\mathbf{x}_k)_i \leq l_i + \epsilon \wedge \partial_i f(\mathbf{x}_k) > 0] \;\text{ or }\; [(\mathbf{x}_k)_i \geq u_i - \epsilon \wedge \partial_i f(\mathbf{x}_k) < 0]\} \,. \tag{3.18}$$

3.4 Proof of Global Convergence

In this section, we introduce and prove the theorem, which guarantees the global convergence of PNKH-B under mild assumptions. We first state the main theorem.

Theorem 2 (Global Convergence)**.** *Suppose*

1. *f is twice differentiable, and ∇f is Lipschitz continuous.*

2. $\inf_{\mathbf{x}}\{f(\mathbf{x})|\mathbf{x} \in C\}$ *is attained, and* C *is a box.*

3. The norm of the projection in our method is induced by

$$\tilde{\mathbf{H}}_k = \mathbf{H}_k + c\mathbf{U}_k\mathbf{U}_k^\top = \mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^\top + c\mathbf{U}_k\mathbf{U}_k^\top,$$

which is the low-rank approximation of the Hessian using Lanczos tridiagonalization plus a positive shift in the orthogonal complement of the Krylov subspace. Moreover, it is symmetric and uniformly positive definite, i.e., $\tilde{\mathbf{H}}_k \succeq s\mathbf{I}$ *for some* $s > 0$ *and for all* $k \in \mathbb{N}$.

Then the sequence $\{\mathbf{x}_k\}_k$ *generated by PNKH-B converges to a stationary point of* (3.1) *regardless of the choice of the starting point* $\mathbf{x}_0 \in C$.

The assumptions hold for PNKH-B with and without variable partitioning. Hence unlike two-metric methods, the convergence of PNKH-B does not hinge upon active/inactive variable partitioning. Also, the theorem can also be straightforwardly extended to the preconditioned setting by repeating the same process in the proof. Moreover, from the theorem, we obtain that our methods globally converge to the optimal solution for convex problems. We now begin to prove the theorem. The proof follows the approach in [100], which studies proximal Newton-type methods. We first state and prove some lemmas, which will be used to prove the global convergence.

Lemma 1 (Descent Direction). *If* f *is twice differentiable,* $\mathbf{H}_k = \mathbf{V}\mathbf{T}\mathbf{V}^\top$ *is generated by Lanczos tridiagonalization with initial vector* $\nabla f(\mathbf{x}_k)$*, the projection norm is induced by* $\tilde{\mathbf{H}}_k = \mathbf{H}_k + c\mathbf{U}_k\mathbf{U}_k^\top$*, where* $\mathbf{U}_k$ *contains orthonormal basis vectors of the orthogonal complement of the Krylov subspace, and* C *is a box, then for any* $\mu_k > 0$*, the update step* $\mathbf{d}_k := \mathbf{x}_{k+1} - \mathbf{x}_k$ *generated by (3.5) and (3.6) satisfies*

$$\nabla f(\mathbf{x}_k)^\top \mathbf{d}_k \leq -\frac{1}{\mu_k}\mathbf{d}_k^\top \tilde{\mathbf{H}}_k \mathbf{d}_k. \tag{3.19}$$

Hence the update step $\mathbf{d}_k$ *is a descent direction.*

Proof of Lemma 1. By the second projection theorem [4, Chapter 9.3], the iterate $\mathbf{x}_{k+1} = \Pi_{\|\cdot\|_{\tilde{\mathbf{H}}_k}}(\mathbf{y}_{k+1})$ if and only if

$$(\mathbf{y}_{k+1} - \mathbf{x}_{k+1})^\top \tilde{\mathbf{H}}_k(\mathbf{z} - \mathbf{x}_{k+1}) \leq 0 \quad \text{for all } \mathbf{z} \in C. \tag{3.20}$$

Substituting $\mathbf{y}_{k+1} = \mathbf{x}_k - \mu_k \mathbf{V}_k \mathbf{T}_k^{-1} \mathbf{V}_k^\top \nabla f(\mathbf{x}_k)$ and $\mathbf{z} = \mathbf{x}_k$ in (3.20), we obtain

$$(\mathbf{x}_k - \mu_k \mathbf{V}_k \mathbf{T}_k^{-1} \mathbf{V}_k^\top \nabla f(\mathbf{x}_k) - \mathbf{x}_{k+1})^\top \tilde{\mathbf{H}}_k(\mathbf{x}_k - \mathbf{x}_{k+1}) \leq 0. \tag{3.21}$$

Substituting $\mathbf{d}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\tilde{\mathbf{H}}_k = \mathbf{V}_k \mathbf{T}_k \mathbf{V}_k^\top + c\mathbf{U}_k \mathbf{U}_k^\top$ into (3.21) and by the fact that the columns of $\mathbf{V}_k$ are orthogonal to that of $\mathbf{U}_k$, we get

$$\mathbf{d}_k^\top \tilde{\mathbf{H}}_k \mathbf{d}_k + \mu_k \nabla f(\mathbf{x}_k)^\top \left(\mathbf{V}_k \mathbf{V}_k^\top\right) \mathbf{d}_k \leq 0. \tag{3.22}$$

Noting that the Lanczos algorithm is initialized by the normalized gradient vector, the first column of $\mathbf{V}_k$ is given by

$$\mathbf{v}_1 = \frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|_2} = \frac{\nabla f(\mathbf{x}_k)}{\gamma}.$$

Consequently, we obtain

$$\begin{aligned} \nabla f(\mathbf{x}_k)^\top \left(\mathbf{V}_k \mathbf{V}_k^\top\right) \mathbf{d}_k &= \gamma \mathbf{v}_1^\top \left(\mathbf{V}_k \mathbf{V}_k^\top\right) \mathbf{d}_k \\ &= \gamma \begin{bmatrix} 1 & 0 \ldots 0 \end{bmatrix} \mathbf{V}_k^\top \mathbf{d}_k \\ &= \gamma \mathbf{v}_1^\top \mathbf{d}_k \\ &= \nabla f(\mathbf{x}_k)^\top \mathbf{d}_k. \end{aligned} \tag{3.23}$$

Plugging this into (3.22), we finally obtain

$$\mu_k \nabla f(\mathbf{x}_k)^\top \mathbf{d}_k \leq -\mathbf{d}_k^\top \tilde{\mathbf{H}}_k \mathbf{d}_k, \tag{3.24}$$

which is equivalent to (3.19). □

Lemma 2 (Armijo Line Search Condition)**.** *Suppose C is a box, ∇f is Lipschitz continuous with constant $L > 0$, $\tilde{\mathbf{H}}_k$'s are symmetric, and $\tilde{\mathbf{H}}_k \succeq s\mathbf{I}$ for all $k \in \mathbb{N}$ and for some $s > 0$, i.e.*

$$||\mathbf{z}||^2_{\tilde{\mathbf{H}}_k} \geq s||\mathbf{z}||_2^2, \quad \textit{for all } \mathbf{z} \textit{ and for all } k \in \mathbb{N}.$$

For line search parameter $\alpha \in (0, 1)$, if step size μ_k satisfies

$$\mu_k \leq \min\left(1, \frac{2s}{L}(1-\alpha)\right),$$

then the following sufficient descent condition is satisfied

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \alpha \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k). \tag{3.25}$$

Proof of Lemma 2. Since $\mathbf{x}_k, \mathbf{x}_{k+1} \in C$, by the Lipschitz continuity of ∇f, we have

$$\begin{aligned}
f(\mathbf{x}_{k+1}) &\leq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 \\
&= f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{L\mu_k}{2}\left(\frac{1}{\mu_k}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2\right) \\
&\leq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) + s(1-\alpha)\left(\frac{1}{\mu_k}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2\right) \\
&\leq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) + (1-\alpha)\left(\frac{1}{\mu_k}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_{\tilde{\mathbf{H}}_k}^2\right) \\
&\leq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) - (1-\alpha)\nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) \\
&\leq f(\mathbf{x}_k) + \alpha \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k).
\end{aligned}$$

Here, the third step uses $\mu_k \leq \frac{2s}{L}(1-\alpha)$, the fourth step utilizes $\tilde{\mathbf{H}}_k \succeq s\mathbf{I}$, and the fifth step applies (3.19). Note that Lemma 1 implies $\nabla f(\mathbf{x}_k)^\top(\mathbf{x}_{k+1} - \mathbf{x}_k) \leq 0$, therefore (3.25) is a sufficient descent condition. □

Lemma 3. *Suppose the assumptions on f, C and $\tilde{\mathbf{H}}_k$'s are the same as those in Lemma 2. Also, the backtracking Armijo line search scheme in Algorithm 7 is used. Then $\mathbf{x}_*$ is a stationary point of* (3.1) *if and only if $\mathbf{x}_*$ is a fixed point of our method.*

Proof of Lemma 3. The iterate $\mathbf{x}_*$ is a fixed point of our method if and only if

$$\mathbf{x}_* = \Pi_{\|\cdot\|_{\tilde{\mathbf{H}}_*}} \left(\mathbf{x}_* - \mu_* \mathbf{V}_* \mathbf{T}_*^{-1} \mathbf{V}_* \nabla f(\mathbf{x}_*)\right), \tag{3.26}$$

where $\tilde{\mathbf{H}}_*$ is the shifted Hessian approximation, and $\mu_* > 0$ is the step size determined by the line search scheme. By the second projection theorem again, it is equivalent to

$$(\mathbf{x}_* - \mu_* \mathbf{V}_* \mathbf{T}_*^{-1} \mathbf{V}_* \nabla f(\mathbf{x}_*) - \mathbf{x}_*)^\top \tilde{\mathbf{H}}_* (\mathbf{z} - \mathbf{x}_*) \leq 0 \quad \text{for all } \mathbf{z} \in C.$$

Using $\tilde{\mathbf{H}}_* = \mathbf{V}_* \mathbf{T}_*^{-1} \mathbf{V}_* + c\mathbf{U}_* \mathbf{U}_*^\top$, and following the same approach as in (3.23), this is simplified to $\nabla f(\mathbf{x}_*)^\top (\mathbf{z} - \mathbf{x}_*) \geq 0$ for all $\mathbf{z} \in C$, which is true if and only if $\mathbf{x}_*$ is a stationary point of the problem. □

Now, we are ready to prove Theorem 2, the global convergence of our method.

Proof of Theorem 2. The sequence $\{f(\mathbf{x}_k)\}_k$ is decreasing because the update directions are descent directions (Lemma 1), and the backtracking Armijo line search scheme guarantees sufficient descent at each step (Lemma 2). Since f is closed and its infimum in C is attained, the decreasing sequence $\{f(\mathbf{x}_k)\}_k$ converges to a limit.

By the sufficient descent condition (3.25), the convergence of $\{f(\mathbf{x}_k)\}_k$ and $\alpha > 0$,

$$\nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k)$$

converges to zero. By Lemma 1, one has

$$(\mathbf{x}_{k+1} - \mathbf{x}_k)^\top \tilde{\mathbf{H}}_k (\mathbf{x}_{k+1} - \mathbf{x}_k) \leq -\mu_k \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k).$$

Hence $(\mathbf{x}_{k+1} - \mathbf{x}_k)^\top \tilde{\mathbf{H}}_k (\mathbf{x}_{k+1} - \mathbf{x}_k)$ converges to zero. Since $\tilde{\mathbf{H}}_k$'s are uniformly positive definite, $\mathbf{x}_{k+1} - \mathbf{x}_k$ converges to the zero vector.

This implies that the sequence $\{\mathbf{x}_k\}_k$ converges to a fixed point of our method. By Lemma 3, the sequence converges to a stationary point of the problem. $\square$

3.5 Experimental Results

We perform three numerical experiments motivated by different applications with PNKH-B. We compare its performance with two state-of-the-art projected Newton-CG (PNCG) methods, which are two-metric schemes; see Section 3.5.1. In Section 3.5.2, we consider a PDE parameter estimation problem. In Section 3.5.3, we apply our method to an image classification problem. In Section 3.5.4, we experiment with an image reconstruction problem. All these applications require fitting a computational model to data, which is typically noisy. Therefore, and since the computational models are expensive, we seek to use the optimization scheme to obtain a high-quality reconstruction within only a few iterations.

We use a fixed tolerance and the same maximum number of iterations for the CG and Lanczos schemes that vary in each experiment. Using the low-rank approximated Hessian metric during the projection renders PNKH-B competitive with respect to the optimization performance and reconstruction quality to similar state-of-the-art two-metric methods. We set the Hessian shift parameter $c = 10^{-3}$.

3.5.1 Benchmark Methods

We compare PNKH-B to an implementation of the two-metric scheme described in [64] and a variant that includes the augmented indexing scheme from [10, 137]. We refer to the scheme obtained using $\mathcal{A}_k^{\text{bound}}$ as PNCG (boundary index) and the scheme obtained using $\mathcal{A}_k^{\text{aug}}$ as PNCG (augmented index); see Section 3.3.3. Since the Lanczos tridiagonalization has the same iterates as CG up to roundoff errors, the main difference between these schemes and our proposed method is the projection. The PNCG schemes use (3.4) to project all the components and are therefore considered two-metric schemes. In contrast, our PNKH-B scheme uses a metric that, on the Krylov subspace, is consistent with that implied by the low-rank approximation of the Hessian and is therefore a generalized one-metric scheme.

As another benchmark, we use the implementation of a reflective Trust region method provided by MATLAB's `fmincon` [28, 29]. We provide this method with our implementation of gradients and Hessian matrix-vector products. We used the same number of iterations as in the PNKH-B and PNCG approaches and kept the other settings at the default.

3.5.2 Experiment 1: Direct Current Resistivity

We use PNKH-B to solve the PDE parameter estimation problem motivated by the Direct Current Resistivity (DCR) described in [64, 134]; see also [36, 107, 149, 152] for background and different instances of this problem.

Model Description The goal of DCR in geophysical imaging is to estimate the conductivity of the subsurface by means of indirect measurement obtained on the earth's surface. Specifically, it first uses electrical sources on the surface to generate direct currents to create electric potential fields in the subsurface. Measurements of these potential fields are then collected on the surface. The parameter estimation aims

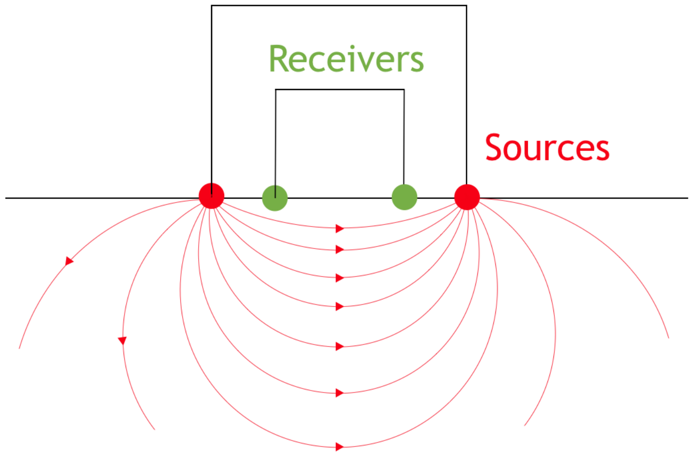


Figure 3.2: An illustration of the DCR experimental setup

at reconstructing a three-dimensional image of the conductivity in the subsurface that is consistent with the measurements. An illustration of the DCR experimental setup is shown in Figure 3.2. For more details and illustrations of the DCR experiment, see, e.g., [36, 64, 107, 134].

Experimental Results To set up the test, we follow the same discretize-then-optimize approach described in [64] that is also used in [134, 151]. Using a uniform mesh with N_m cells and N_n nodes, we obtain the discrete forward problem

$$\mathbf{D} = \mathbf{P}^\top \mathbf{A}(\mathbf{m})^{-1}\mathbf{Q} + \boldsymbol{\epsilon} = \mathbf{P}^\top \mathbf{U} + \boldsymbol{\epsilon}, \tag{3.27}$$

where $\mathbf{A}(\mathbf{m}) \in \mathbb{R}^{N_n \times N_n}$ is a finite-volume discretization of the Poisson operator for the conductivity model $\mathbf{m} \in \mathbb{R}^{N_m}$, $\mathbf{P} \in \mathbb{R}^{N_n \times N_r}$ is the receiver matrix that maps the fields to data, the columns of $\mathbf{Q} \in \mathbb{R}^{N_n \times N_s}$ are discretized sources, the columns of $\mathbf{U} \in \mathbb{R}^{N_n \times N_s}$ are the potential fields, and $\boldsymbol{\epsilon} \in \mathbb{R}^{N_r \times N_s}$ is Gaussian noise. Here N_r and N_s are the number of receivers and sources, respectively. Note that with suitable discretization and boundary conditions, $\mathbf{A}$ is non-singular, which means that $\mathbf{m} \mapsto \mathbf{U}(\mathbf{m})$ is well-defined and differentiable.

Given the measurement data $\mathbf{D}$, sources $\mathbf{Q}$, and receivers $\mathbf{P}$, we estimate the corresponding model parameter $\mathbf{m}$ by solving the optimization problem

$$\min_{\mathbf{m}} \frac{1}{2}\|\mathbf{P}^\top \mathbf{A}(\mathbf{m})^{-1}\mathbf{Q} - \mathbf{D}\|_F^2 + \frac{\gamma}{2}\|\mathbf{L}(\mathbf{m} - \mathbf{m}_{\text{ref}})\|_2^2 \quad \text{subject to} \quad \mathbf{m}_l \le \mathbf{m} \le \mathbf{m}_u.$$

Here, $\gamma > 0$ is a regularization parameter, $\mathbf{m}_{\text{ref}}$ is a given reference model, $\mathbf{L}$ is a regularization operator, $\mathbf{m}_l$ and $\mathbf{m}_u$ are the upper and lower bounds, respectively, which are used to enforce the physical constraints for the model parameters.

As common, we use the Gauss-Newton approximate Hessian $\mathbf{G}$ given by

$$\mathbf{G} = \mathbf{J}(\mathbf{m})^\top \mathbf{J}(\mathbf{m}) + \gamma \mathbf{L}^\top \mathbf{L},$$

where the Jacobian of the residual of (3.27) is

$$\mathbf{J}(\mathbf{m}) = -\mathbf{P}^\top \mathbf{A}(\mathbf{m})^{-1}(\nabla_{\mathbf{m}}(\mathbf{A}(\mathbf{m})\mathbf{U}))^\top. \tag{3.28}$$

Note that the dimensions of $\mathbf{m}$ are typically very large. Moreover, each evaluation of the objective function or product with the Jacobian $\mathbf{J}$ or its transpose or computing the approximate Hessian-vector multiplication $\mathbf{y} \mapsto \mathbf{G}\mathbf{y}$ require inverting the PDE operator $\mathbf{A}$ (i.e., solving the PDE) $\min(N_r, N_s)$ times per source. Hence the computations in each outer (Newton-Krylov method) or inner (line search) iterations when solving the DCR model problem are very expensive, especially when there are a lot of sources.

In this experiment, we solve a 3-dimensional DCR problem on a mesh containing $36 \times 36 \times 12$ cells discretizing the domain $\Omega = (0,1)^3$. The test problem features 25 sources and 1,369 receivers located on the top surface. Following the finite volume discretization presented in [64], we use a cell-centered discretization of the model $\mathbf{m}$ and nodal discretizations of the sources, receivers, and fields. We add 1% noise to the data and enforce smoothness by using a diffusion regularizer with regularization parameter $\alpha = 10^{-3}$. We also use symmetric successive over-relaxation (SSOR) as a preconditioner.

In our setup, we exclude voxels close to the boundary, sources, and receivers from the inversion. As a result, our model $\mathbf{m}$ is discretized over $30 \times 30 \times 10$ cells instead; in particular, $\mathbf{m}$ has size $n = 9900$. The bounds $\mathbf{m}_l$ and $\mathbf{m}_u$ are set as vectors of all -4.6's and -1's, respectively. The upper bound is purposely set as smaller than some pixel values of the ground truth to test the ability of the methods to identify the active variables. The main cost of the parameter estimation is the large number of discrete PDE solves to evaluate the objective function, its gradient, and matrix-vector products with $\mathbf{J}$ and $\mathbf{J}^\top$. Therefore, we limit the number of CG/Lanczos iterations to five in all instances.

The experimental results for the DCR problem are shown in Figures 3.3 to 3.5. In the experiments, we perform 20 Newton steps. We stop the CG method and Lanczos scheme when the norm of the relative residual drops below 10^{-2} or after five iterations. Using only a small number of CG/Lanczos iterations and preconditioning with the regularization operator is important to obtain smooth image reconstructions [64, 134]. The tolerance in the interior point method is set to be 10^{-10}. As can be seen in Figure 3.3(a)-(b), PNCG and PNKH-B outperform the `fmincon` scheme by some margin. In the plot, we only show the first 120 out of 251 Hessian-vector products for `fmincon`. At the last iteration, `fmincon` reached a comparable objective function value as the PNKH-B and PNCG schemes. Moreover, the proposed methods have a significant boost in the initial convergence on the objective function value and the norm of the projected gradient. This is particularly evident in the early iterations as can be seen, e.g., by a one-order reduction of the objective function and projected gradient in the second iteration and the visual quality of the parameter estimate at the third iteration; see Figure 3.4. At this iteration, we see that the results of

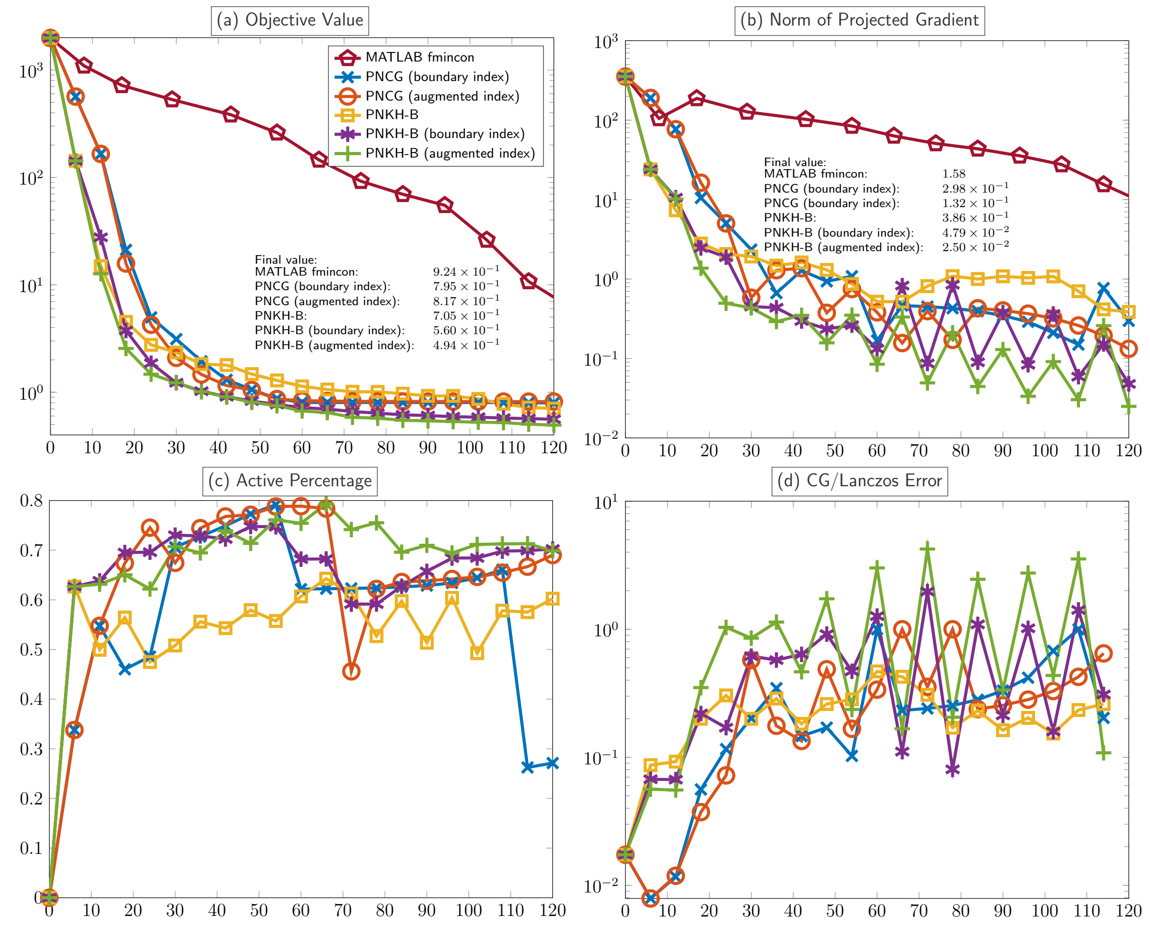


Figure 3.3: Comparison of the convergence of two PNCG methods and three variants of PNKH-B for the direct current resistivity experiment in Section 3.5.2. (a): Relative reduction of objective function. (b): Norm of the projected gradient. (c): Percentage of variables in $\mathcal{A}_k^{\text{aug}}$ defined in (3.18). (d): Relative residual error of CG/Lanczos. The x-axis represents the number of Hessian-vector products.

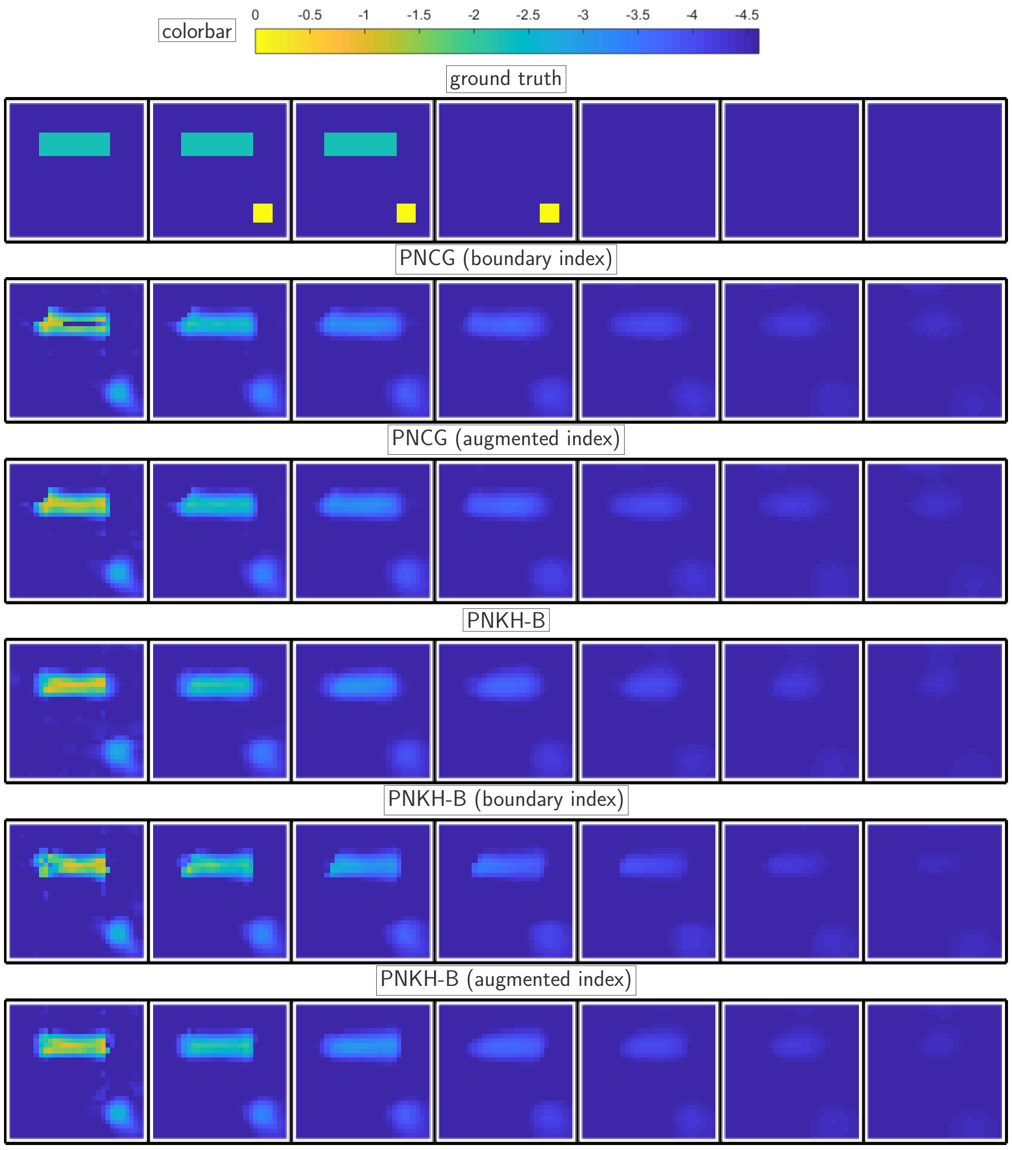


Figure 3.4: Results after the third iteration on DCR generated by the five methods. The upper bound is purposely set to be $\mathbf{m}_u = -1$, which is smaller than some pixel values in the ground truth to test the ability of the methods to identify active variables.

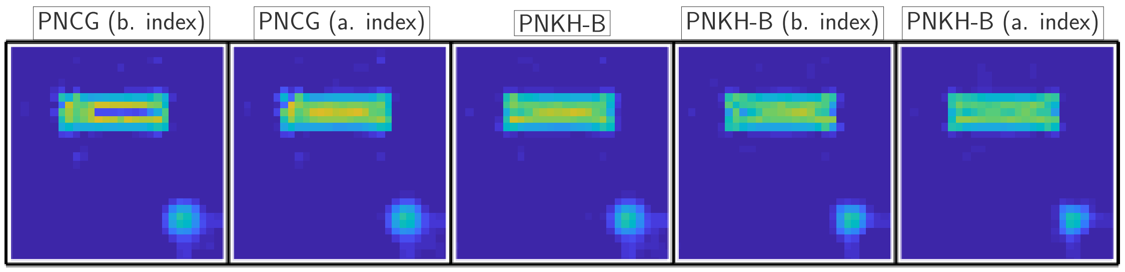


Figure 3.5: First slice of the final results on DCR generated by the five methods. There are noticeable artifacts in the final results of PNCG.

the proposed PNKH-B and PNKH-B (augmented index) are closer to the ground truth and appear smoother. While the results obtained using all methods are similar at the final iteration, we note that PNCG (boundary index) leads to a non-smooth reconstruction; see Figure 3.5. Since PNCG is a two-metric scheme, the loss of the smoothness might be due to suboptimal scaling of the gradient step in (3.15) or the inconsistency of the preconditioner caused by the indexing.

The proposed methods also have slightly smaller objective values after 20 iterations. Table 3.1 shows that all five methods require a comparable runtime. We highlight that the added costs of the interior point method used to compute the projection are only between 0.5% and 2.5% and took on average between 0.04 and 0.2 seconds. While the Lanczos tridiagonalization in PNKH-B takes longer on average than the conjugate gradient method in PNCG, the overall runtime of PNKH-B is reduced. A key reason for the computational savings is the smaller number of backtracking line search iterations, which consequently reduces the number of projections and also PDE solves.

3.5.3 Experiment 2: Image Classification

We compare the performance of PNKH-B and PNCG for a multinomial logistic regression (MLR) arising in the supervised classification of hand-written digits in the MNIST dataset [99]. Some example images of the MNIST dataset are shown in Figure 2.1.

Model Description Let n_c be the number of classes, and Δ_{n_c} be the unit simplex in $\mathbb{R}^{n_c}$. Denote the training data by $\{(\mathbf{b}_j, \mathbf{c}_j)\}_{j=1}^{N} \subset \mathbb{R}^{n_\mathrm{f}} \times \Delta_{n_c}$, where $\mathbf{b}_j$ is the input feature and $\mathbf{c}_j$ is its corresponding class label. First, given training data $\mathbf{b}_j$, we apply a random feature model to propagate the original features into a higher-dimensional space $\mathbb{R}^{m_\mathrm{f}}$, i.e.,

$$\mathbf{d}_j = \tanh(\mathbf{K}\mathbf{b}_j) \in \mathbb{R}^{m_\mathrm{f}},$$

where the entries of $\mathbf{K} \in \mathbb{R}^{m_\mathrm{f} \times n_\mathrm{f}}$ are randomly drawn from a standard normal distribution. We then solve the multinomial logistic regression (MLR) problem

$$\min_{\mathbf{X}_l \le \mathbf{X} \le \mathbf{X}_u} \frac{1}{N} \sum_{j=1}^{N} -\mathbf{c}_j^\top \log\left(h_\mathbf{X}(\mathbf{d}_j)\right)$$

to obtain the classifier

$$h_\mathbf{X}(\mathbf{d}_j) = \frac{\exp\left(\mathbf{X}\mathbf{d}_j\right)}{\mathbf{1}_{n_c}^\top \exp\left(\mathbf{X}\mathbf{d}_j\right)}.$$

For a detailed derivation of the MLR problem, see Section 2.3. Here we use $\mathbf{X}_l, \mathbf{X}_u \in \mathbb{R}^{n_c \times m_\mathrm{f}}$ to model lower and upper bounds on the entries of $\mathbf{X}$, respectively, with the goal to regularize the problem and improve generalization, which means improving the performance on the test data set. Since the MLR problem is a smooth convex optimization problem, we use $\mathbf{G} = \nabla^2 f(\mathbf{X})$.

In our experiment, we use the MNIST dataset [99], which consists of $60,000$ 28×28 grey-scale hand-written images of digits ranging from 0 to 9 that are split into $N = 50,000$ training images and $M = 10,000$ validation images. Here the transformed feature vectors are in a $m_\mathrm{f} = 4,000$-dimensional space.

Experimental Results We use a fixed number of 20 inexact Newton steps. We stop the CG method and Lanczos scheme when the norm of the relative residual drops

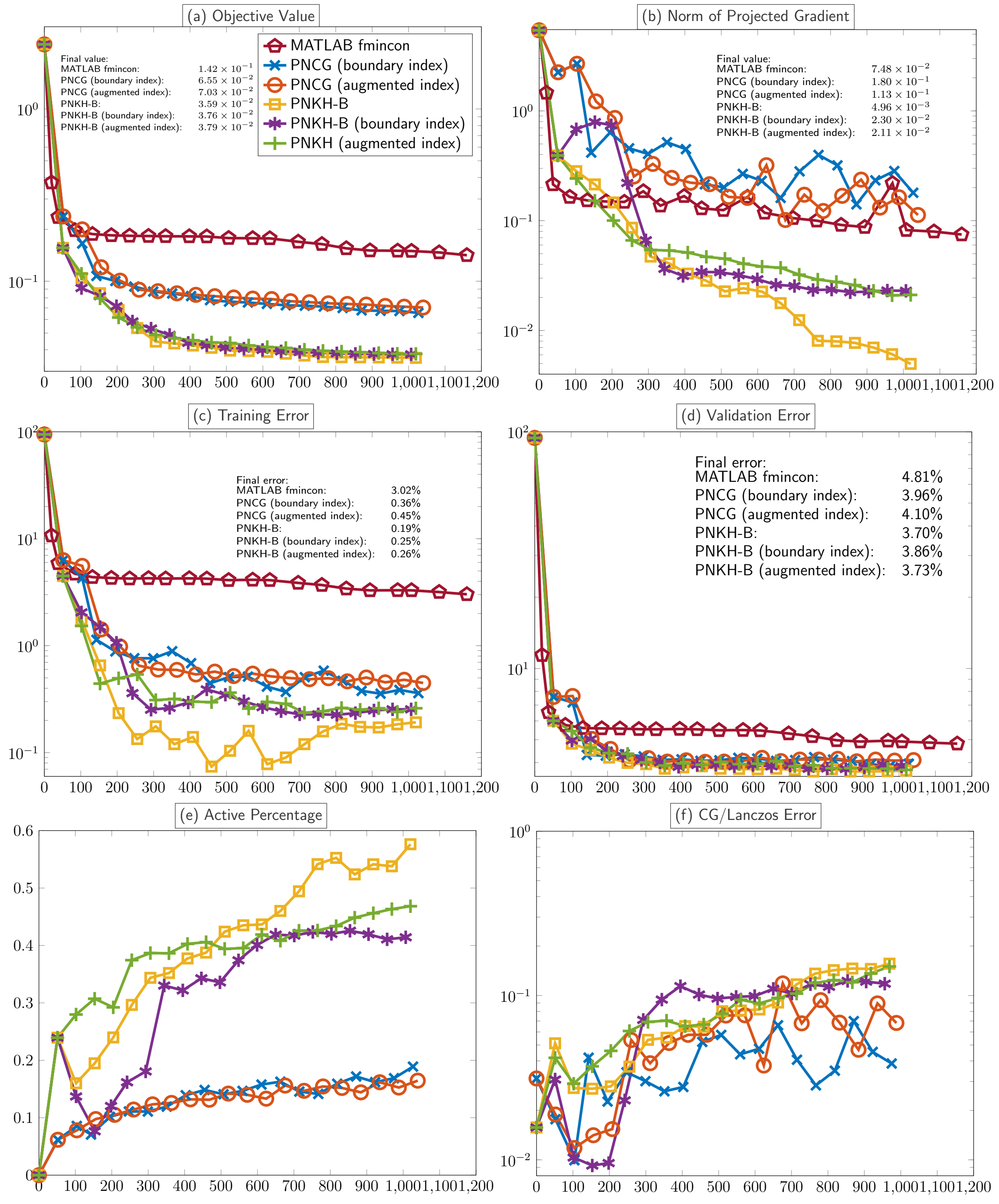


Figure 3.6: Comparison of the convergence of two PNCG methods and three variants of PNKH-B for the image classification problem in Section 3.5.3. (a): Relative reduction of objective function. (b): Norm of the projected gradient. (c): Training errors. (d): Validation errors. (e): Percentage of variables in $\mathcal{A}_k^{\text{aug}}$ defined in (3.18). (f): Relative residual error of CG/Lanczos. The x-axis represents the number of Hessian-vector products.

below 10^{-2} or after 50 iterations. We tuned the bounds on $\mathbf{X}$ so that the trained hypothesis function performs well on the validation data. We set the tolerance in the interior point method to be 10^{-12}. In our case, we choose the entries of $\mathbf{X}_l$ and $\mathbf{X}_u$ to be -0.05 and 0.05, respectively. The performance of the optimization schemes and the accuracy of the hypothesis function can be seen in Figure 3.6. In this example, all PNCG and PNKH-B variants outperform the blackbox MATLAB fmincon method in objective value, training error and testing error. Moreover, in Figure 3.6(a)-(c), the three PNKH-B methods boost the initial convergence and outperform the PNCG methods with respect to the objective function value, norm of the projected gradient, and training error by some margin. The comparison for the validation data is overall comparable; see Figure 3.6(d). Despite the more expensive projection step, the PNKH-B variants require a similar runtime in this experiment; see Table 3.1.

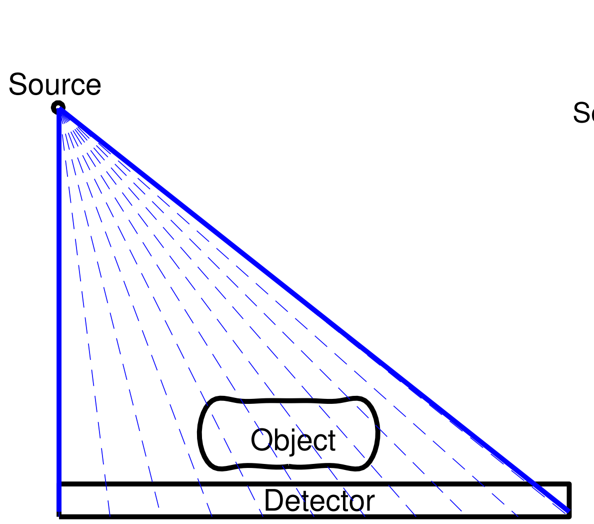
3.5.4 Experiment 3: Spectral Computed Tomography

We consider an image reconstruction problem arising in energy-windowed spectral computed tomography (CT). The goal is to identify the material composition of an object from measurements taken with x-rays at different energy levels and from different projection angles. Our experimental setup follows [76, 77] which also provide an excellent description and derivation of the problem. An illustration of the experimental setup is given in Figure 3.7.

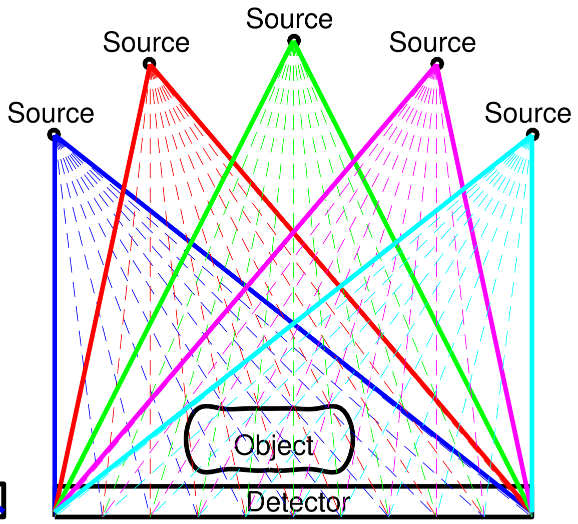
Model Description As a forward model, we consider the discretized energy-windowed spectral CT model

$$\mathbf{y} = (\mathbf{S}^\top \otimes \mathbf{I})\text{exp}\{-(\mathbf{C} \otimes \mathbf{A})\mathbf{w}\} + \boldsymbol{\epsilon}, \tag{3.29}$$

where $\mathbf{I} \in \mathbb{R}^{(N_d \cdot N_p) \times (N_d \cdot N_p)}$ is the identity matrix, $\mathbf{S} \in \mathbb{R}^{N_e \times N_b}$ contains the spectrum energy of each energy window, $\mathbf{C} \in \mathbb{R}^{N_e \times N_m}$ contains the attenuation coefficients of each material at each energy level, $\mathbf{A} \in \mathbb{R}^{(N_d \cdot N_p) \times N_v}$ contains the lengths of the x-ray



(a) X-ray beams from one angle



(b) X-ray beams from multiple angles

Figure 3.7: An illustration of the energy-windowed spectral CT experimental setup

beams through the pixels of the image, $\mathbf{y} \in \mathbb{R}^{N_d \cdot N_p \cdot N_b}$ is the observed data containing the x-ray photons of each energy window, $\mathbf{w} \in \mathbb{R}^{N_v \cdot N_m}$ represents the weights of the materials of each pixel (and is the unknown variable), and $\boldsymbol{\epsilon} \in \mathbb{R}^{N_d \cdot N_p \cdot N_b}$ is the measurement noise. Here, N_p is the number of angles of the x-ray beams, N_b is the number of detectors, and each of them detects a specific energy window, N_m is the number of materials, N_e is the number of energy levels of the emitted x-ray beams, N_d and N_v are related to the number of pixels of the image. In particular, for an image of size $n \times n$, $N_d = n$ and $N_v = n^2$.

The goal of the energy-windowed spectral CT model is to estimate the weights of materials $\mathbf{w}$ given the other variables except the noise in (3.29). Hence we formulate the following optimization problem

$$\min_{0 \le \mathbf{w} \le \mathbf{w}_u} \frac{1}{2} \|\mathbf{y} - (\mathbf{S}^\top \otimes \mathbf{I}) \exp\{-(\mathbf{C} \otimes \mathbf{A})\mathbf{w}\}\|_2^2 + \frac{\gamma_1}{2} \|\mathbf{D}\mathbf{w}_{1:N_v}\|_2^2 + \gamma_2 \sum_{i=N_v+1}^{2N_v} \mathbf{w}_i.$$

Here, the bound constraints are used to enforce physical bounds, where the weights cannot be negative and cannot exceed the upper bound $\mathbf{w}_h$. The second and third terms are the regularization terms also used in [77]. The second term involves the

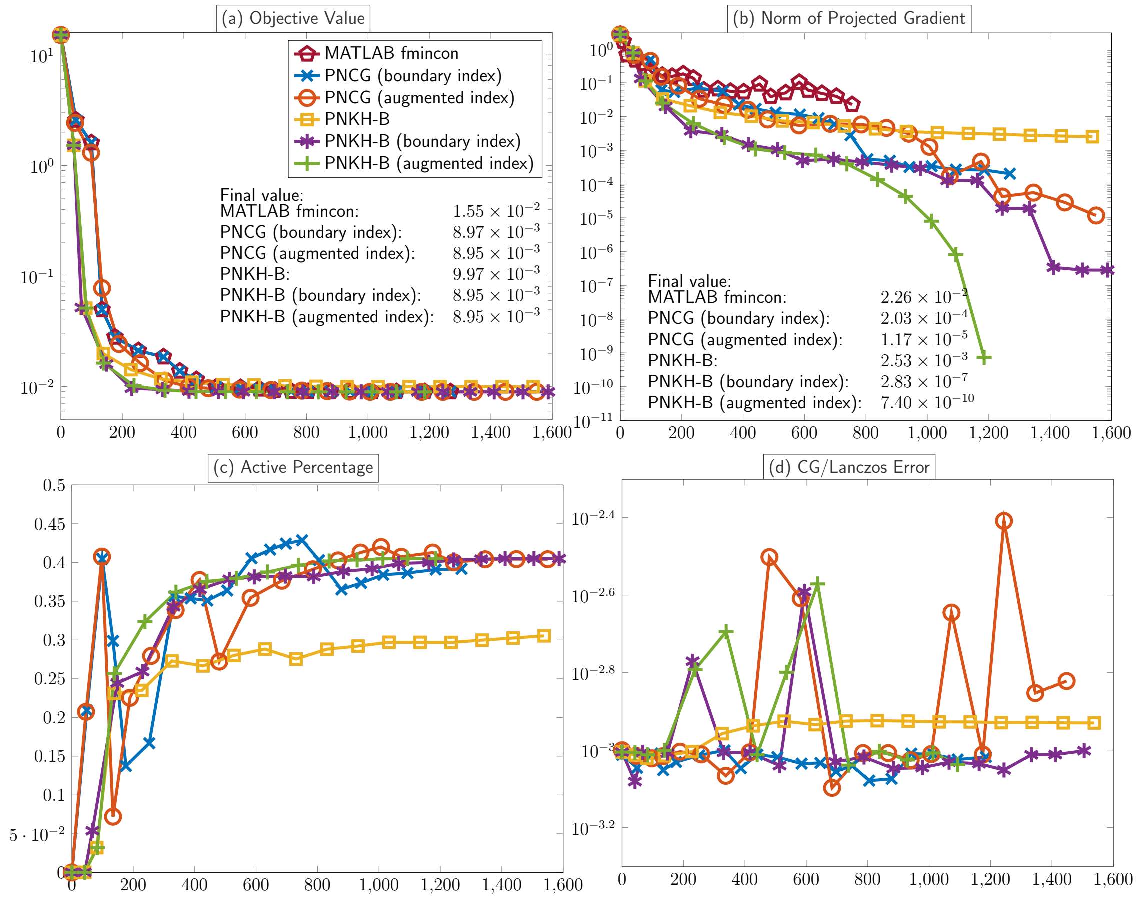


Figure 3.8: Comparison of the convergence of two PNCG methods and three variants of PNKH-B for the energy-windowed spectral CT problem in Section 3.5.4. (a): Relative reduction of objective function. (b): Norm of the projected gradient. (c): Percentage of variables in $\mathcal{A}_k^{\text{aug}}$ defined in (3.18). (d): Relative residual error of CG/Lanczos. The x-axis represents the number of Hessian-vector products.

discrete gradient operator $\mathbf{D}$ and enforces the smoothness of the first material, and the last term promotes sparsity of the second material. As common in nonlinear least-squares problems, we use the Gauss-Newton approximation of the Hessian, i.e.,

$$\mathbf{G} = \mathbf{J}(\mathbf{w})^\top \mathbf{J}(\mathbf{w}) + \gamma_1 \mathbf{D}^\top \mathbf{D},$$

where $\mathbf{D}$ is a discrete differential operator acting on the first N_v entries.

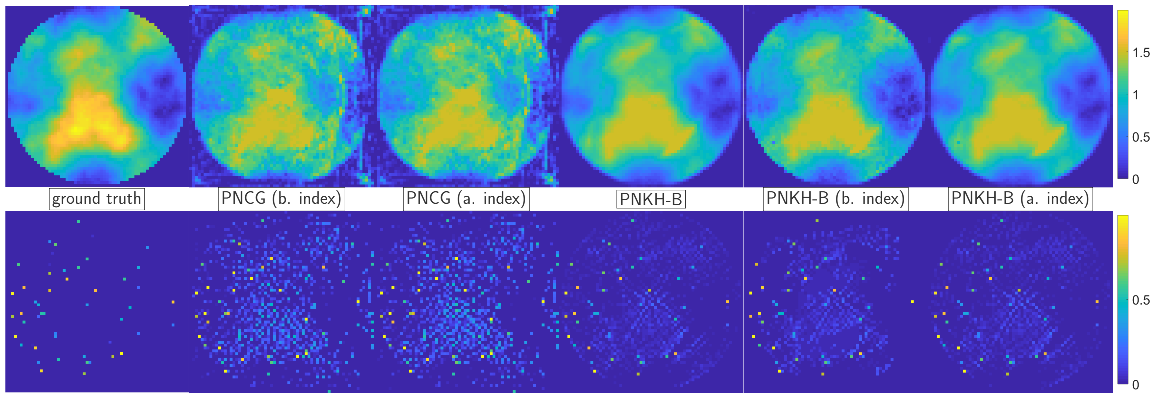


Figure 3.9: Reconstructed images after the second iteration generated by the five methods on CT. The top and bottom images are the estimated composition of the two materials. The upper bound is purposely set to be $\mathbf{w}_u = 1.5$, which is smaller than some pixel values in the ground truth, to test the ability of the methods to identify active variables. The final image quality is comparable for all schemes.

Experimental Results The number of variables in this problem is $n = 8,192$. We set the regularization parameters as $\gamma_1 = 10^9$ and $\gamma_2 = 10^3$. Since the Kronecker products are implemented effectively, the CT model problem is the least intense among the three testing problems in terms of computational cost. Therefore, we use a higher tolerance of 10^{-3} for the relative residual norm and allow for up to 100 iterations in the CG/Lanczos method. In this experiment, we show that our methods can converge to the optimal solution with very small gradient norm by choosing a very small tolerance of 10^{-16} in the interior point method, so that the projection is solved very accurately. Moreover, we purposely choose a tight bound $\mathbf{w}_u = [1.5, 1.5, ..., 1.5]^\top$ to test the ability of the methods to compute a solution with many active entries, specifically some entries in the ground truth are outside of this bound. The experimental results of the CT model problem are shown in Figures 3.8 and 3.9. In this problem instance, the PNKH-B methods outperform the blackbox MATLAB fmincon method objective value and norm of projected gradients. The performances of PNCG methods are similar to fmincon in objective value but outperforms it in norm of projected gradient. The proposed methods converge faster initially, and all schemes achieve comparable results. In the second iteration of Figure 3.8(a), the iterate of the three proposed methods achieve 60 times smaller objective function values than the comparing methods. This also leads to a considerable improvement in the reconstruction quality; see Figure 3.9. In Figure 3.8(b), PNKH-B with boundary index and augmented index give competitive performance in terms of the norm of the projected gradient. Specifically, they achieve gradient norm with magnitudes 10^{-7} and 10^{-11}, respectively. In this example, the norm of the projected gradient decays very slowly in the last iterations of the PNKH-B scheme. From Table 3.1, the runtime of PNKH-B with variable partitioning is roughly double that of PNCG's because of the very low tolerance (10^{-16}) in the interior point method, which we use to reduce the norm of the projected gradient. We note that this increased accuracy does not necessarily improve the quality of the image reconstruction and in our experiments runtimes similar to that of PNCG can be obtained with a less accurate projection. Moreover, the runtime of PNKH-B *without* variable partitioning is roughly three times that of PNCG's because it performs ten backtracking line searches before running into a line search break at iteration 18; the reason is the insufficient decrease in objective function value. Finally, we note that further improvements of the optimality conditions can be obtained by adaptively choosing the rank of $\mathbf{H}_k$, e.g., by choosing the relative residual tolerance in PCG using a forcing sequence that tightens the tolerance as the solution becomes more accurate and increasing or removing the limit on the number of inner iterations.

3.6 Summary

In this chapter, we present PNKH-B, a projected Newton-Krylov method for bound-constrained minimization whose search direction and projection rely on a low-rank approximation of the (approximate) Hessian. Our method can be seen as an extension of Newton-CG methods to bound-constrained problems since we compute the low-

Table 3.1: Comparison of runtime in seconds of our prototype implementation, the number of projections, average number of iterations of the interior point method (IPM) and final step length on the three experiments. The numbers of variables in experiment 1, 2 and 3 are 9900, 40010 and 8192, respectively. The total number of iterations is 20. The tests are run on a Microway system that has four Intel Xeon E5-4627 CPUs with 40 cores and 1 TB of memory and the software plateform is MATLAB R2020b.

		PNCG (b. index)	PNCG (a. index)	PNKH-B	PNKH-B (b. index)	PNKH-B (a. index)
Exp. 1	Runtime	192.4	195.9	195.1	184.0	185.3
	IPM time(avg)			4.8(0.2)	0.9(0.04)	0.9(0.05)
	No. of proj.	31	27	24	20	20
	Avg IPM iter.			79	46	49
	Final step len.	8.0e-1	8.2e-1	1.9e-1	1	1
	# of func. evals	51	47	44	40	40
	# of grad evals	20	20	20	20	20
	# of HessMatvecs	120	120	120	120	120
Exp. 2	Runtime	55.96	56.83	65.73	62.95	67.07
	IPM time(avg)			2.13 (0.07)	0.78 (0.03)	0.80 (0.04)
	No. of proj.	31	32	29	30	30
	Avg IPM iter.			25.67	25.67	23.20
	Final step len.	3.17e-02	1.41e-2	8.45e-2	1	1
	# of func. evals	52	52	50	45	40
	# of grad evals	20	20	20	20	20
	# of HessMatvecs	960	955	963	959	978
Exp. 3	Runtime	31.77	31.37	90.51	52.75	38.53
	IPM time(avg)			25.89(1.04)	14.65(0.70)	13.33(0.89)
	No. of proj.	24	25	34	22	18
	Avg IPM iter.			201	201	187.6
	Final step len.	1	1	1.96e-4	1	1
	# of func. evals	43	44	45	40	29
	# of grad evals	20	20	20	19	14
	# of HessMatvecs	1374	1561	1621	1641	1271

rank approximation of the Hessian using a few steps of Lanczos tridiagonalization. The novelty of our method is the use of the metric induced by this approximation in the projection step. We contribute an interior point method that effectively exploits the low-rank approximation to achieve a complexity that is linear with respect to the number of variables. As compared to two-metric schemes that require partitioning variables into active and inactive sets to ensure convergence, the consistent choice of the metric in PNKH-B leads to a simpler algorithm whose convergence is guaranteed even without variable partitioning. We also propose two variants of the framework, which incorporate the current knowledge of the active/inactive variables; this improved the convergence in some cases. The experimental results on PDE parameter estimation, machine learning, and image reconstruction show that the proposed methods lead to faster initial convergence with moderate runtime overhead compared to the existing state-of-the-art projected Newton-CG methods. Our methods are also competitive in the final objective value, norm of the projected gradient, and reconstruction quality. We provide our prototype MATLAB code at `https://github.com/EmoryMLIP/PNKH-B`.

One direction for future work is to improve the efficiency of the quadratic program solved in the projection stage (e.g., using alternative interior point methods [44, 45, 158] or constrained conjugate gradient methods [115, 155]). In addition, PNKH-B can be generalized to a proximal Newton-Krylov scheme for minimizing large-scale composite functions. We can also extend the idea to trust-region Newton methods, where the region of trust is induced by the Hessian metric.

Chapter 4

A Modified Newton-Krylov Method for Log-Sum-Exp Minimization

The content and results in this chapter are based on a joint work with James Nagy and Lars Ruthotto. In this chapter, we present a simple yet effective modified Newton-Krylov algorithm geared toward the minimization of the log-sum-exp function for a linear model. Problems of this kind arise commonly in geometric programming and machine learning and are typically of large-scale. Since the search direction is computed by a Krylov subspace method, the proposed algorithm only requires matrix-vector products with the linear model and is thus scalable to large problem sizes. Although the log-sum-exp function is smooth and convex, standard line search Newton-type methods can fail to converge because the quadratic approximation is unbounded from below. The key contribution in this work is a novel Hessian modification that shifts the Hessian in the output space of the linear model. This can be motivated by the fact that, in machine learning, model inputs often do not have an intuitive meaning, while model outputs are interpretable. Unlike standard modified

Newton-type schemes, the modified Hessian can be rank-deficient. Still, we prove that the shift renders the quadratic approximation to be bounded from below and that the overall scheme converges to a global minimizer under mild assumptions.

This chapter is organized as follows. We first describe the problem setup and review related work. We then introduce the proposed scheme and provide a proof of global convergence. We finally demonstrate the effectiveness of the proposed scheme with two numerical experiments.

4.1 Problem Description

We consider minimization problems of the form

$$\min_{\mathbf{w}} f(\mathbf{w}) = g(\mathbf{w}) - \mathbf{c}^\top \mathbf{J}\mathbf{w}, \tag{4.1}$$

where

$$g(\mathbf{w}) := \log\left(\mathbf{1}_m^\top \exp(\mathbf{J}\mathbf{w})\right)$$

is the log-sum-exp function for a linear model $\mathbf{J} \in \mathbb{R}^{m\times n}$, $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^m$, and $\mathbf{1}_m \in \mathbb{R}^m$ is a vector of all ones. The problem (4.1) arises commonly in machine learning and optimization. For example, multinomial logistic regression (MLR) in classification problems [153, 119] is formulated as (4.1). In geometric programming [140, 154, 157], a non-convex problem can be convexified through a reformulation to the form (4.1). The log-sum-exp function itself also has extensive applications in machine learning. For instance, it can serve as a smooth approximation to the element-wise maximum function [52, 121], where the smoothness is desirable in model design since gradient-based optimizers are commonly used. Moreover, the log-sum-exp function is closely related to widely used softmax and entropy functions. For instance, the dual to an entropy maximization problem is a log-sum-exp minimization problem [14, Example 5.5], and the gradient of the log-sum-exp function is the softmax function [48].

Despite the smoothness and convexity of the log-sum-exp function, a standard implementation of line search Newton-type methods can be problematic. To realize this, note that the gradient and Hessian of the log-sum-exp function are given by

$$\nabla f(\mathbf{w}) = \mathbf{J}^\top(\mathbf{p} - \mathbf{c}), \quad \text{and} \quad \nabla^2 f(\mathbf{w}) = \mathbf{J}^\top \mathbf{H}\mathbf{J},$$
$$\text{with} \quad \mathbf{p} = \frac{\exp(\mathbf{J}\mathbf{w})}{\mathbf{1}_m^\top \exp(\mathbf{J}\mathbf{w})}, \quad \text{and} \quad \mathbf{H} = \operatorname{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^\top.$$

The Hessian is positive semi-definite but rank-deficient because the null space of $\mathbf{H}$ contains $\mathbf{1}_m$. Even more problematic is that when $\mathbf{p}$ is close to a standard basis vector (commonly in MLR), the Hessian is close to the zero matrix even when the gradient is non-zero. In Newton's method, this means that the local quadratic approximation can be unbounded from below. To be precise, it is unbounded from below if and only if the gradient is not in the column space of the Hessian [4, Exercise 2.19].

4.2 Related Work

Disciplined convex programming (DCP) packages (e.g., CVX [61]) can reliably solve the log-sum-exp minimization problem through a reformulation. For instance, CVX first formulates the problem using exponential cones [3, Section 5.2.6] and applies backend solvers to solve it either directly (e.g., MOSEK [2]) or through successive polynomial approximation (e.g., SPDT3 [141] and SeDuMi [139]). However, it can be computationally demanding as the number of conic constraints scales with the product of the dimension of the output space and the number of data. For instance, our computational equipment cannot perform the image classification experiments for the whole dataset in Section 4.5.2. Furthermore, the formulation relies on access to the elements of $\mathbf{J}$; i.e., this approach is not applicable in a matrix-free setting where

$\mathbf{J}$ is not built explicitly, and only routines for performing matrix-vector products are provided.

Tikhonov regularization [41, 59, 70] avoids the cost of reformulation and alleviates the convergence issues with Newton-type methods. It adds $\frac{\alpha}{2}\|\mathbf{w}\|_2^2$ with $\alpha > 0$ to the objective function. The regularization shifts the Hessian by $\alpha\mathbf{I}$ and renders it positive definite, where $\mathbf{I}$ is the identity matrix. Nonetheless, the Tikhonov regularization introduces a bias and consequently changes the optimal solution. The regularization parameter α has to be chosen judiciously – a large α renders the problem easier to solve and produces a more regular solution but introduces more bias. In addition, one cannot use effective parameter selection algorithms [57, 27, 17, 144] for linear problems due to the nonlinearity of the log-sum-exp function. On the other hand, first-order methods like gradient descent [14, 122] or AdaGrad [38], which do not use the Hessian matrix can avoid the problem. However, their convergence is inferior to methods that utilize curvature information [39].

Modified Newton-type methods effectively tackle problems with rank-deficient or indefinite Hessians and meanwhile do not introduce any bias. The idea is to add a shift to the Hessian so that at the ith iteration, the scheme solves

$$\min_{\mathbf{w}} \frac{1}{2}(\mathbf{w}-\mathbf{w}_i)^\top(\nabla^2 f(\mathbf{w}_i)+\beta_i\mathbf{M}_i)(\mathbf{w}-\mathbf{w}_i)+\nabla f(\mathbf{w}_i)^\top(\mathbf{w}-\mathbf{w}_i), \tag{4.2}$$

where β_i is a parameter and the shift $\mathbf{M}_i$ renders the Hessian to be sufficiently positive definite. The quadratic approximation is bounded from below since the modified Hessian is positive definite. Hence the convergence issues are avoided. The effect of the Hessian shift is reminiscent of the Tikhonov regularization. Indeed, the scheme is sometimes called a Tikhonov-regularized Newton update [127, Chapter 3.3]. However, the key conceptual difference between (4.2) and Tikhonov regularization is that the former does not introduce any bias to the problem [143], i.e., the optimal solution to the problem is independent of β_i's. There are different ways of defining $\mathbf{M}_i$. For instance, $\mathbf{M}_i$ is spanned by some of the eigenvectors of the Hessian [62, 122], or is a modification to the factorization of the Hessian [54, 114, 117]. However, the computations needed for these approaches are intractable for large-scale problems commonly arising in machine learning. A simple and computationally feasible approach is to set $\mathbf{M}_i$ as the identity matrix [101, 106, 127], which will be used as a comparing method in our numerical experiments. There is some existing work that studies the convergence properties of modified Newton-type methods. In [122], conditions for the global convergence of general modified Newton-type methods are given. In [102, 43, 129], the local quadratic convergence of modified Newton-type methods with an identity matrix shift for convex problems is proven under certain assumptions. A globally convergent modified Newton-type method whose Hessian shift is determined by a line search is proposed in [111]. A globally convergent modified Newton-type scheme for composite functions is also proposed [37]. In [147, 148, 75], modified Newton-type methods for specific applications are presented.

4.3 Proposed Method

We propose a modified Newton-Krylov method geared toward log-sum-exp minimization problems of the form (4.1). At the ith iteration, we first consider the quadratic approximation (4.2) with $\mathbf{M}_i = \mathbf{J}^\top \mathbf{J}$. That is,

$$\begin{aligned} \min_{\mathbf{w}} q_i(\mathbf{w}) &= \frac{1}{2}(\mathbf{w}-\mathbf{w}_i)^\top (\nabla^2 f(\mathbf{w}_i) + \beta_i \mathbf{J}^\top \mathbf{J})(\mathbf{w}-\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top (\mathbf{w}-\mathbf{w}_i) \\ &= \frac{1}{2}(\mathbf{w}-\mathbf{w}_i)^\top \mathbf{J}^\top (\mathbf{H}_i + \beta_i \mathbf{I})\mathbf{J}(\mathbf{w}-\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top (\mathbf{w}-\mathbf{w}_i), \end{aligned} \tag{4.3}$$

whose minimizer is given by $\mathbf{w}_i + \mathbf{x}$, where $\mathbf{x}$ solves the Newton equation

$$\nabla^2 q_i(\mathbf{w}_i)\mathbf{x} = -\nabla q_i(\mathbf{w}_i), \tag{4.4}$$

and $\mathbf{H}_i$ is $\mathbf{H}$ evaluated at $\mathbf{w}_i$. It is important to note that the Hessian shift in (4.3) is different from the existing modified Newton approaches discussed earlier, which seek to obtain a positive definite Hessian and lead to an update in the input space of the linear model (i.e., the $\mathbf{w}$ space). Instead, it generates an update direction in the output space of the linear model (i.e., the row space of $\mathbf{J}$). This is preferable especially in machine learning applications because the model inputs often do not have intuitive meaning while the model outputs are explicable. Although the Hessian of (4.3) can be rank-deficient especially when the linear model is over-parametrized (i.e., $\mathbf{J}$ has more columns than rows), it is positive definite in the output space of the linear model. Consequently, the quadratic approximation is bounded from below, and the overall scheme provably converges to a global minimum; see Section 4.4 for a detailed derivation.

An alternative formulation for (4.3) is

$$\min_{\mathbf{w}} \frac{1}{2}(\mathbf{w}-\mathbf{w}_i)^\top \nabla^2 f(\mathbf{w}_i)(\mathbf{w}-\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top(\mathbf{w}-\mathbf{w}_i) + \frac{\beta_i}{2}\|\mathbf{J}(\mathbf{w}-\mathbf{w}_i)\|_2^2,$$

which can be interpreted as a Newton scheme with a proximal term acting on the row space of $\mathbf{J}$. This formulation shows that β_i controls the step size in a nonlinear line search arc. To be precise, $\beta_i = 0$ and ∞ correspond to a Newton update with step size 1 and 0, respectively, and the update is given nonlinearly for $0 < \beta_i < \infty$. The formulation also shows that our proposed scheme bears similarity to L^2 natural gradient descent (NGD) methods [128, 123], which use the same proximal term. Nonetheless, unlike our approach, L^2 NGD methods do not directly incorporate Hessian information into its search direction and approximate curvature information using only the linear model.

The crucial difference between the proximal term and Tikhonov regularization is that the former does not introduce any bias [127, 143]; i.e., the optimal solution is independent of β_i. Another advantage is that Tikhonov regularization requires parameter tuning, which is commonly done using a grid search for nonlinear problems like (4.1), while in our proposed method, β_i's are automatically selected by a backtracking Armijo line search scheme. The proposed scheme can also be perceived as a proximal point algorithm acting on the second-order approximation [127].

We compute the update direction $\Delta\mathbf{w}_i$ by approximately solving the Newton equation (4.4) using a Krylov subspace method (e.g., conjugate gradient method [122, 14]) and obtain the next iterate $\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta\mathbf{w}_i$. In particular, the Krylov subspace is given by

$$\mathcal{K}_r(\nabla^2 q_i(\mathbf{w}_i), \nabla q_i(\mathbf{w}_i)) = \mathcal{K}_r(\mathbf{J}^\top(\mathbf{H}_i + \beta_i\mathbf{I})\mathbf{J}, \nabla f(\mathbf{w}_i)), \tag{4.5}$$

where r is the dimension of the Krylov subspace. Since the Krylov subspace method only requires routines to perform Hessian-vector multiplications, our method is applicable to large-scale problems commonly arising in machine learning applications where the linear model is only available through matrix-vector products. We note that an update direction has to be re-computed for each attempted value of β_i during line search. In other words, unlike standard Newton-CG schemes, the update direction computation cannot be re-used. However, our experimental results show that the proposed method is still efficient in terms of computational cost thanks to the effectiveness of the modified Hessian. An outline of the implementation of the proposed method is presented in Algorithm 9.

The proposed method is simple because it has a straightforward implementation similar to standard line search Newton-Krylov methods. In the meantime, it is still effective in solving the log-sum-exp minimization problems (4.1) and achieves comparable performance compared to commonly used solvers while avoiding their respective drawbacks as outlined in Section 4.2. Moreover, it is robust in the sense that it successfully computes the solutions to challenging problems while other methods fail to converge; see Section 4.5 for numerical experiments.

Algorithm 9: Outline of the proposed algorithm for solving (4.1)

```
1: Inputs: Linear model $\mathbf{w} \mapsto \mathbf{J}\mathbf{w}$, initial guess $\mathbf{w}_0$, initial $\beta_0$, tolerances wtol and
   gtol, line search parameter $\gamma \in (0, 1)$.
2: for $i = 0, 1, 2, \ldots$ do
3:    compute $f(\mathbf{w}_i)$, $\nabla f(\mathbf{w}_i)$ and build routines for performing $\mathbf{v} \mapsto \nabla^2 f(\mathbf{w}_i)\mathbf{v}$
4:    for $j = 0, 1, 2, \ldots$ do
5:       compute $\Delta\mathbf{w}_i$ by applying Krylov-subspace methods to approximately solve
         $\nabla^2 q_i(\mathbf{w}_i)\mathbf{x} = -\nabla q_i(\mathbf{w}_i)$ with the current $\beta_i$ and Krylov subspace
         $\mathcal{K}_r(\nabla^2 q_i(\mathbf{w}_i), \nabla q_i(\mathbf{w}_i))$
6:       if $f(\mathbf{w}_i + \Delta\mathbf{w}_i) < f(\mathbf{w}_i) + \gamma\nabla f(\mathbf{w}_i)^\top \Delta\mathbf{w}_i$ then
7:          set $\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta\mathbf{w}_i$ and break
8:       else
9:          set $\beta_i = 2\beta_i$
10:      end if
11:   end for
12:   if $\|\mathbf{w}_{i+1} - \mathbf{w}_i\|_2 / \|\mathbf{w}_i\|_2 <$ wtol or $\|\nabla f(\mathbf{w}_{i+1})\|_2 <$ gtol then
13:      break
14:   end if
15:   if $j = 0$ then
16:      set $\beta_{i+1} = 0.5 * \beta_i$
17:   else
18:      set $\beta_{i+1} = \beta_i$
19:   end if
20: end for
21: Output: approximate solution $\mathbf{w}_{i+1}$.
```

4.4 Proof of Global Convergence

In this section, we prove the global convergence of the proposed method. It is noteworthy that existing convergence results cannot be directly applied due to the rank-deficiency of our modified Hessian. For instance, it is assumed in [122, Chapter 6.2] that the modified Hessian is positive definite and has a bounded condition number. Our proof is modified from the approach in [100], which studies proximal Newton-type methods for composite functions.

We first state the main theorem.

Theorem 3. *Assume that f is defined in* (4.1)*, and $\inf_{\mathbf{w}} f(\mathbf{w})$ is attained in $\mathbb{R}$, then the iterative scheme* (4.7) *converges to a global minimum regardless of the choice of* *initial guess* $\mathbf{w}_0$.

We note that Theorem 3 also applies to the case where the Newton equation (4.4) is solved exactly. In the following, we will first discuss some properties of the proposed method. We will then state and prove some lemmas which will aid the proof of Theorem 3.

Recall that the Krylov subspace $\mathcal{K}_r(\nabla^2 q_i(\mathbf{w}_i), \nabla q_i(\mathbf{w}_i))$ is constructed to approximately solve the Newton equation and obtain the update direction $\Delta\mathbf{w}_i$. This is equivalent to building a rank-r approximation $\nabla^2 q_i(\mathbf{w}_i) \approx \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top$ and computing the next iterate by

$$\mathbf{w}_{i+1} = \arg\min_{\mathbf{w}} \frac{1}{2}(\mathbf{w}-\mathbf{w}_i)^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top(\mathbf{w}-\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top(\mathbf{w}-\mathbf{w}_i). \tag{4.6}$$

Here, the columns of $\mathbf{V}_i \in \mathbb{R}^{n\times r}$ form an orthonormal basis for the Krylov subspace and $\mathbf{T}_i \in \mathbb{R}^{r\times r}$. Since $\nabla f(\mathbf{w}_i) \in \text{row}(\mathbf{J}) = \text{col}(\mathbf{J}^\top(\mathbf{H}_i + \beta_i\mathbf{I})\mathbf{J})$ for $\beta_i > 0$ and that the Krylov subspace always contains $\nabla f(\mathbf{w}_i)$, the column space of $\mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top$ always contains $\nabla f(\mathbf{w}_i)$. This means that the quadratic function (4.6) is bounded from below [4, Exercise 2.19] and admits a minimum. The iterate $\mathbf{w}_{i+1}$ is the minimum norm solution to (4.6) given by

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta\mathbf{w}_i, \quad \text{where} \quad \Delta\mathbf{w}_i = -\mathbf{V}_i\mathbf{T}_i^{-1}\mathbf{V}_i^\top\nabla f(\mathbf{w}_i). \tag{4.7}$$

Next, we state and prove some lemmas which will be used to prove the main theorem.

Lemma 4 (Update Direction)**.** *The update* $\Delta\mathbf{w}_i$ *generated by the iterative scheme* (4.7) *satisfies*

$$\Delta \mathbf{w}_i \in \mathrm{row}(\mathbf{J}), \tag{4.8}$$

$$\Delta \mathbf{w}_i^\top \nabla^2 q_i(\mathbf{w}_i) \Delta \mathbf{w}_i = \Delta \mathbf{w}_i^\top \mathbf{V}_i \mathbf{T}_i \mathbf{V}_i^\top \Delta \mathbf{w}_i. \tag{4.9}$$

Here, (4.8) *means that the update direction is in the output space of the linear model.*

Proof of Lemma 4. By construction, the Krylov subspace (4.5) is a subspace of $\mathrm{row}(\mathbf{J})$, and by (4.7) we have $\Delta \mathbf{w}_i \in \mathrm{col}(\mathbf{V}_i)$. Thus we have $\Delta \mathbf{w}_i \in \mathrm{col}(\mathbf{V}_i) \subseteq \mathrm{row}(\mathbf{J})$, which proves (4.8).

Consider the full representation of the Hessian of (4.3) generated by the Krylov subspace method

$$\nabla^2 q_i(\mathbf{w}_i) = \mathbf{J}^\top (\mathbf{H}_i + \beta_i \mathbf{I}) \mathbf{J} = \begin{bmatrix} \mathbf{V}_i & \mathbf{U}_i \end{bmatrix} \begin{bmatrix} \mathbf{T}_i & \mathbf{D}_1 \\ \mathbf{D}_2 & \mathbf{D}_3 \end{bmatrix} \begin{bmatrix} \mathbf{V}_i^\top \\ \mathbf{U}_i^\top \end{bmatrix},$$

where $\mathrm{col}(\mathbf{V}_i) \perp \mathrm{col}(\mathbf{U}_i)$. We have

$$\begin{aligned} \Delta \mathbf{w}_i^\top \nabla^2 q_i(\mathbf{w}_i) \Delta \mathbf{w}_i &= \Delta \mathbf{w}_i^\top \begin{bmatrix} \mathbf{V}_i & \mathbf{U}_i \end{bmatrix} \begin{bmatrix} \mathbf{T}_i & \mathbf{D}_1 \\ \mathbf{D}_2 & \mathbf{D}_3 \end{bmatrix} \begin{bmatrix} \mathbf{V}_i^\top \\ \mathbf{U}_i^\top \end{bmatrix} \Delta \mathbf{w}_i \\ &= \begin{bmatrix} \Delta \mathbf{w}_i^\top \mathbf{V}_i & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{T}_i & \mathbf{D}_1 \\ \mathbf{D}_2 & \mathbf{D}_3 \end{bmatrix} \begin{bmatrix} \mathbf{V}_i^\top \Delta \mathbf{w}_i \\ \mathbf{0} \end{bmatrix}, \quad \text{as } \Delta \mathbf{w}_i \in \mathrm{col}(\mathbf{V}_i), \\ &= \Delta \mathbf{w}_i^\top \mathbf{V}_i \mathbf{T}_i \mathbf{V}_i^\top \Delta \mathbf{w}_i, \end{aligned}$$

which proves (4.9). □

Lemma 5 (Descent Direction)**.** *The update* $\Delta \mathbf{w}_i$ *generated by* (4.7) *satisfies the descent condition*

$$\nabla f(\mathbf{w}_i)^\top \Delta \mathbf{w}_i \le -\Delta \mathbf{w}_i^\top \mathbf{J}^\top (\mathbf{H}_i + \beta_i \mathbf{I}) \mathbf{J} \Delta \mathbf{w}_i. \tag{4.10}$$

Proof of Lemma 5. Since $\mathbf{w}_{i+1}$ is a solution to (4.6), for any $t \in (0,1)$, we have

$$\frac{1}{2}\Delta\mathbf{w}_i^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top \Delta\mathbf{w}_i + \nabla f(\mathbf{w}_i)^\top \Delta\mathbf{w}_i \le \frac{1}{2}(t\Delta\mathbf{w}_i)^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top (t\Delta\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top (t\Delta\mathbf{w}_i).$$

This implies

$$\begin{aligned}\frac{(1-t^2)}{2}\Delta\mathbf{w}_i^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top \Delta\mathbf{w}_i + (1-t)\nabla f(\mathbf{w}_i)^\top \Delta\mathbf{w}_i &\le 0\\ \frac{(1+t)}{2}\Delta\mathbf{w}_i^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top \Delta\mathbf{w}_i + \nabla f(\mathbf{w}_i)^\top \Delta\mathbf{w}_i &\le 0\\ \nabla f(\mathbf{w}_i)^\top \Delta\mathbf{w}_i &\le -\frac{(1+t)}{2}\Delta\mathbf{w}_i^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top \Delta\mathbf{w}_i.\end{aligned}$$

Letting $t \to 1^-$, we obtain

$$\nabla f(\mathbf{w}_i)^\top \Delta\mathbf{w}_i \le -\Delta\mathbf{w}_i^\top \mathbf{V}_i\mathbf{T}_i\mathbf{V}_i^\top \Delta\mathbf{w}_i. \tag{4.11}$$

Combining (4.9) and (4.11), we obtain (4.10). $\square$

In the following lemma, we will make use of the fact that ∇f is Lipschitz continuous. This is because the gradient of the log-sum-exp function is the softmax function, which is Lipschitz continuous [48, 91].

Lemma 6 (Armijo Line Search Condition)**.** *Let $\lambda_{\min}$ be the smallest nonzero eigenvalue of $\mathbf{J}^\top\mathbf{J}$, and L be the Lipschitz constant for ∇f. For line search parameter $\gamma \in (0,1)$ and*

$$\beta_i \ge \frac{L}{2\lambda_{\min}(1-\gamma)}, \tag{4.12}$$

the following Armijo line search condition holds

$$f(\mathbf{w}_{i+1}) \le f(\mathbf{w}_i) + \gamma\nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1} - \mathbf{w}_i). \tag{4.13}$$

Proof of Lemma 6. First, note that

$$\|\mathbf{J}(\mathbf{w}_{i+1}-\mathbf{w}_i)\|^2_{\mathbf{H}_i+\beta_i\mathbf{I}} \geq \beta_i\|\mathbf{J}(\mathbf{w}_{i+1}-\mathbf{w}_i)\|^2_2 \geq \beta_i\lambda_{\min}\|(\mathbf{w}_{i+1}-\mathbf{w}_i)\|^2_2. \tag{4.14}$$

Here, in the second step we used that $(\mathbf{w}_{i+1}-\mathbf{w}_i) \in \mathrm{row}(\mathbf{J}) = \mathrm{row}(\mathbf{J}^\top\mathbf{J})$ (Lemma 4), $\mathrm{row}(\mathbf{J}^\top\mathbf{J})^\perp = \mathrm{null}(\mathbf{J}^\top\mathbf{J})$, and $\lambda_{\min}$ is the smallest nonzero eigenvalue of $\mathbf{J}^\top\mathbf{J}$. Next, we have

$$\begin{aligned}
f(\mathbf{w}_{i+1}) &\leq f(\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1}-\mathbf{w}_i) + \frac{L}{2}\|\mathbf{w}_{i+1}-\mathbf{w}_i\|^2_2,\\
&\leq f(\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1}-\mathbf{w}_i) + \beta_i\lambda_{\min}(1-\gamma)\|\mathbf{w}_{i+1}-\mathbf{w}_i\|^2_2,\\
&\leq f(\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1}-\mathbf{w}_i) + (1-\gamma)\|\mathbf{J}(\mathbf{w}_{i+1}-\mathbf{w}_i)\|^2_{\mathbf{H}_i+\beta_i\mathbf{I}},\\
&\leq f(\mathbf{w}_i) + \nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1}-\mathbf{w}_i) - (1-\gamma)\nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1}-\mathbf{w}_i),\\
&= f(\mathbf{w}_i) + \gamma\nabla f(\mathbf{w}_i)^\top(\mathbf{w}_{i+1}-\mathbf{w}_i).
\end{aligned}$$

Here, we used the Lipschitz continuity of ∇f in the first step, (4.12) in the second step, (4.14) in the third step, and Lemma 5 in the fourth step. □

Lemma 7 (Stationary Point)**.** *The iterative scheme* (4.7) *generates a fixed point* $\mathbf{w}_*$ *if and only if* $\mathbf{w}_*$ *is a stationary point.*

Proof of Lemma 7. "$\Leftarrow$": Substituting $\nabla f(\mathbf{w}_*) = \mathbf{0}$ into (4.7), we obtain $\Delta\mathbf{w}_* = 0$. Hence $\mathbf{w}_*$ is a fixed point.

"$\Rightarrow$": Let $\mathbf{v} = \mathbf{w} - \mathbf{w}_*$ for any $\mathbf{w}$. Since $\mathbf{w}_*$ is a fixed point to (4.6), we have, for any $t \in \mathbb{R}$,

$$\frac{1}{2}(t\mathbf{v})^\top\mathbf{V}_*\mathbf{T}_*\mathbf{V}_*^\top(t\mathbf{v})+\nabla f(\mathbf{w}_*)^\top(t\mathbf{v}) \geq \frac{1}{2}(\mathbf{w}_*-\mathbf{w}_*)^\top\mathbf{V}_*\mathbf{T}_*\mathbf{V}_*^\top(\mathbf{w}_*-\mathbf{w}_*)+\nabla f(\mathbf{w}_*)^\top(\mathbf{w}_*-\mathbf{w}_*).$$

This implies

$$\begin{aligned} \frac{t^2}{2}\mathbf{v}^\top \mathbf{V}_* \mathbf{T}_* \mathbf{V}_*^\top \mathbf{v} + t\nabla f(\mathbf{w}_*)^\top \mathbf{v} &\geq \mathbf{0} \\ t\nabla f(\mathbf{w}_*)^\top \mathbf{v} &\geq -\frac{t^2}{2}\mathbf{v}^\top \mathbf{V}_* \mathbf{T}_* \mathbf{V}_*^\top \mathbf{v}. \end{aligned} \tag{4.15}$$

Consider the directional derivative of g at $\mathbf{w}_*$ for any direction $\mathbf{v}$

$$\begin{aligned} D_\mathbf{v} f(\mathbf{w}_*) &= \lim_{t\to 0} \frac{f(\mathbf{w}_* + t\mathbf{v}) - f(\mathbf{w}_*)}{t} \\ &= \lim_{t\to 0} \frac{t\nabla f(\mathbf{w}_*)^\top \mathbf{v} + O(t^2)}{t} \\ &\geq \lim_{t\to 0} \frac{-\frac{t^2}{2}\mathbf{v}^\top \mathbf{V}_* \mathbf{T}_* \mathbf{V}_*^\top \mathbf{v} + O(t^2)}{t}, \quad \text{by (4.15)}, \\ &= 0. \end{aligned}$$

This implies $\nabla f(\mathbf{w}_*) = \mathbf{0}$, that is, $\mathbf{w}_*$ is a stationary point. $\square$

Now, we are ready to prove the main theorem.

Proof of Theorem 3. The sequence $\{f(\mathbf{w}_i)\}_i$ is decreasing because the update directions are descent directions (Lemma 5) and the Armijo line search scheme guarantees sufficient descent at each step (Lemma 6). By the continuity of f, it is closed [9, Proposition 1.1.2]. Since f is closed and attains its infimum in $\mathbb{R}$, the decreasing sequence $\{f(\mathbf{w}_i)\}_i$ converges to a limit.

By the sufficient descent condition (4.13), the convergence of $\{f(\mathbf{w}_i)\}_i$ and $\alpha > 0$,

$$\nabla f(\mathbf{w}_i)^\top (\mathbf{w}_{i+1} - \mathbf{w}_i)$$

converges to zero. Hence, by (4.10),

$$\Delta\mathbf{w}_i^\top \mathbf{J}^\top (\mathbf{H}_i + \beta_i \mathbf{I}) \mathbf{J} \Delta\mathbf{w}_i$$

converges to zero. Since $(\mathbf{H}_i + \beta_i \mathbf{I})$ is positive definite and $\Delta \mathbf{w}_i \in \text{row}(\mathbf{J})$ (Lemma 4), $\Delta \mathbf{w}_i$ converges to the zero vector.

This implies that $\mathbf{w}_i$ converges to a fixed point of (4.7). By Lemma 7, $\mathbf{w}_i$ converges to a stationary point. By the convexity of f, $\mathbf{w}_i$ converges to a global minimum. $\square$

4.5 Numerical Experiments

We perform two numerical experiments for minimizing the log-sum-exp function for a linear model. We compare the performance of the proposed method with three commonly applied line search iterative methods and three disciplined convex programming (DCP) solvers; see Section 4.5.1. In Section 4.5.2, we consider multinomial logistic regression (MLR) arising in image classification. In Section 4.5.3, we experiment with a log-sum-exp minimization problem. The experimental results show that the proposed method is competitive in terms of accuracy, time-to-solution, and robustness with the comparing methods.

4.5.1 Benchmark Methods

We compare the proposed scheme with three common line search iterative schemes and three DCP solvers for machine learning and geometric programming applications. Firstly, we implement a standard Newton-CG (NCG) algorithm with a backtracking Armijo line search. Secondly, we compare with an L^2 natural gradient descent (NGD) method [123, 128] that approximately solves

$$\min_{\mathbf{w}} \frac{1}{2} \nabla f(\mathbf{w}_i)^\top (\mathbf{w} - \mathbf{w}_i) + \frac{\lambda_i}{2} \|\mathbf{J}(\mathbf{w} - \mathbf{w}_i)\|_2^2,$$

using CG to obtain the next iterate, where λ_i controls the step size and is determined by a backtracking Armijo line search scheme, and the last term is a proximal term acting on the output space of the linear model. This scheme bears similarity to the proposed method as the proximal term has the same effect as the shift in Hessian of the proposed method. However, it does not make use of the Hessian and only approximates curvature information using the linear model. Thirdly, to demonstrate the effectiveness of the Hessian modification in the proposed method, we compare with a standard modified Newton-Krylov (SMNK) scheme, which approximately solves (4.2) with $\mathbf{M}_i = \mathbf{I}$ using Lanczos tridiagonalization, which has the same iterates as CG up to rounding errors but allows computations for the update direction to be re-used during line search. For the proposed method, the Newton equation (4.4) is approximately solved by CG. In each experiment, we use the same maximum number of iterations and tolerance for the CG and Lanczos schemes across different line search iterative methods.

In addition, we apply CVX [61], a DCP package, paired with three different back-end solvers (SPDT3 [141], SeDuMi [139], and MOSEK [2]). The best precision for CVX is used in the experiments; see [61] for detailed information.

Cost Measurement We measure the computational costs for different line search iterative methods in terms of work units. In particular, a work unit represents a matrix-vector product with the linear model or its transpose. This is because these computations are usually the most expensive steps during optimization. For instance, in the MLR experiments of Section 4.5.2, the linear model $\mathbf{J}$ contains the propagated high dimensional features of all the training data. Note that the number of work units in one iteration can differ across different line search iterative methods since a different number of CG/Lanczos iterations or line search can be performed. In addition to work unit, we also compare computational costs for all methods in total runtime.

4.5.2 Experiment 1: Image Classification

Perhaps the most prominent example of log-sum-exp minimization is multinomial logistic regression (MLR) arising in supervised classification. Here, we experiment on an MLR problem for the classification of MNIST [99] and CIFAR-10 [92] image datasets. The MNIST dataset consists of $60,000$ 28×28 hand-written images for digits from 0 to 9. The CIFAR-10 consists of $60,000$ 32×32 color images equally distributed for the following ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Example images for the two datasets are shown in Figure 2.1 and Figure 2.2, respectively.

Problem Description Let n_f be the number of features, n_c be the number of classes, and Δ_{n_c} be the n_c-dimensional unit simplex. Denote a set of data by $\{\mathbf{y}_k, \mathbf{c}_k\}_k \subset \mathbb{R}^{n_f} \times \Delta_{n_c}$, where $\mathbf{y}_k$ and $\mathbf{c}_k$ are the input feature and target output label, respectively. In our experiments, we consider two feature extractors that enhance the features $\mathbf{y}_k$ by propagating it into a higher dimensional space $\mathbb{R}^m$. The first feature extractor is a random feature model (RFM) [79, 131]. It applies a nonlinear transformation given by

$$\mathbf{a}_{\mathrm{RFM}}(\mathbf{y}_k) = \sigma(\mathbf{Z}\mathbf{y}_k + \mathbf{b}),$$

where σ is the element-wise ReLU activation function, $\mathbf{Z} \in \mathbb{R}^{m \times n_f}$ and $\mathbf{b} \in \mathbb{R}^m$ are randomly generated. The second feature extractor is performed by propagating the features through the hidden layers of a pre-trained AlexNet [93]. In particular, the AlexNet was pre-trained on the ImageNet dataset [34], which is similar to the CIFAR-10 dataset, using MATLAB's deep neural networks toolbox. This procedure is also known as transfer learning. These feature extractors can empirically enhance the generalization of the model, i.e., the ability to classify unseen data correctly.

The goal of the supervised classification problem is to train a softmax classifier

$$s(\mathbf{W}, \mathbf{a}(\mathbf{y}_k)) = \frac{\exp(\mathbf{W}\mathbf{a}(\mathbf{y}_k))}{\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}(\mathbf{y}_k))} \tag{4.16}$$

such that $s(\mathbf{W}, \mathbf{a}(\mathbf{y}_k)) \approx \mathbf{c}_k$. Here $\mathbf{W}$ are model parameters, the exp and division are applied element-wise, $\mathbf{1}_{n_c}$ is an n_c-dimensional vector of all ones, and $\mathbf{a} : \mathbb{R}^{n_f} \to \mathbb{R}^m$ is a feature extractor. To this end, we consider the sample average approximation (SAA) [89, 118, 87] of an MLR problem given by

$$\min_{\mathbf{W} \in \mathbb{R}^{n_c \times m}} f(\mathbf{W}) = \frac{1}{n} \sum_{k=1}^{n} \left[\log\left(\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}(\mathbf{y}_k)) \right) - \mathbf{c}_k^\top \mathbf{W}\mathbf{a}(\mathbf{y}_k) \right],$$

where n is the number of training data, the log operation is applied element-wise. For a detailed derivation of the MLR problem, see Section 2.3. The feature extractor is assumed to be fixed since the focus is on the log-sum-exp minimization problem.

Experimental Results In the MLR experiments, the line search iterative solvers stop when the norm of gradient is below 10^{-14} or after 3,000 work units. We stop the CG and Lanczos scheme when the norm of the relative residual drops below 10^{-3} or after 20 iterations.

We first perform a small-scale experiment in which only $n = 100$ training data is used, and a random feature model with dimension $m = 1,000$ is applied. Since under this setup the data can be fit perfectly to achieve a zero training error, the model predictions (4.16) are close to standard basis vectors near an optimum. In this situation, the Hessian is close to a zero matrix, and the robustness of the solvers can be tested. The results are reported in Table 4.1 and Figure 4.1. In Table 4.1, the results for the standard Newton-CG scheme are not shown, as it fails to converge near the end. This is because the Hessian vanishes and consequently the second-order approximation is unbounded from below. The natural gradient descent method has

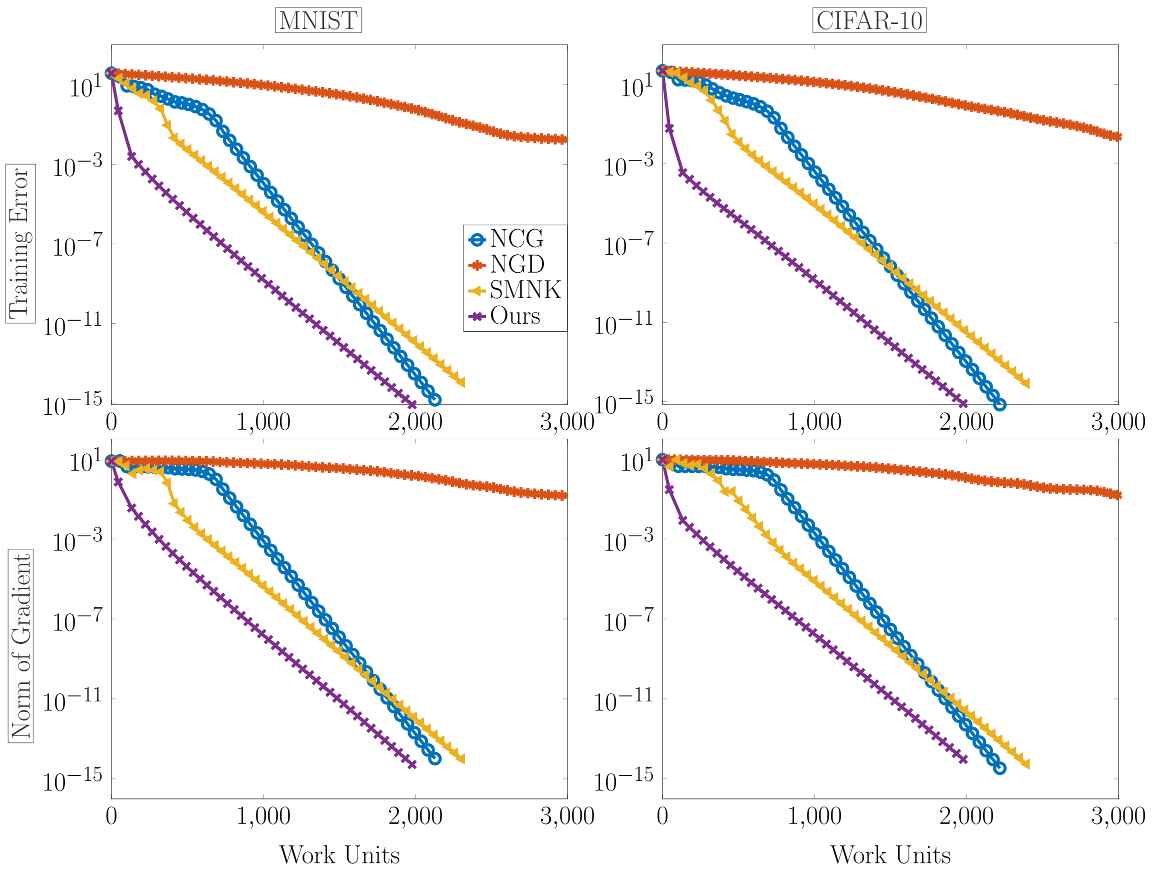


Figure 4.1: Experimental results on small-scale MLR experiments in which the propagated random features have dimension $m = 1,000$ and $n = 100$ training data are used.

the slowest convergence and has yet to converge at the end. Both the standard modified Newton-Krylov method and the proposed scheme achieve the stopping criteria under the specified work units. In particular, the proposed scheme has superior convergence where the objective function value is up to five orders of magnitude smaller than the second-best method. The proposed scheme also has the fastest runtime to obtain solutions. This demonstrates the effectiveness of the proposed method and the efficacy of its modified Hessian over the standard one. SeDuMi, and particularly SDPT3, can achieve very accurate results, but their runtime is about 15 times more than the proposed method. MOSEK fails to obtain a solution.

We then experiment with $n = 50,000$ training data and $10,000$ validation data.

Table 4.1: Results on small-scale MLR experiments in which the propagated random features have dimension $m = 1,000$ and $n = 100$ training data are used. The final objective function value, norm of gradient, and total runtime are reported. Some results are not shown because the corresponding scheme fails to return a solution. The tests are run on an Apple Macbook Pro with a 10-core M1 Max CPU and 32 GBs of memory, and the software platform is MATLAB R2022a.

Dataset		NCG	NGD	SMNK	Ours	SeDuMi	SDPT3	MOSEK
MNIST	f	–	1.69e-02	1.11e-14	8.35e-16	6.65e-15	0.00e+00	–
	$\|\nabla f\|_2$	–	1.42e-01	9.87e-15	5.24e-15	1.15e-14	5.14e-140	–
	Time	–	3.01s	3.14s	2.00s	37.95s	29.65s	–
CIFAR-10	f	–	2.08e-02	8.27e-15	7.97e-16	2.32e-15	0.00e+00	–
	$\|\nabla f\|_2$	–	1.40e-01	5.51e-15	9.47e-15	4.87e-15	2.26e-26	–
	Time	–	3.24s	3.38s	2.08s	35.78s	31.33s	–

For the MNIST dataset, we use an RFM to propagate the features to an $m = 1,000$-dimensional space. For the CIFAR-10 dataset, features with dimension $m = 9,216$ are extracted from the *pool5* layer of a pre-trained AlexNet. Here different feature extractors are used for the two datasets because a better validation accuracy can be achieved. In Figure 4.2, the results for an MLR problem are illustrated. In Figure 4.3, we report the performance for an MLR problem with a Tikhonov regularization term $\frac{\alpha}{2}\|\mathbf{W}\|_{\mathrm{F}}^2$, where $\alpha = 10^{-3}$. Since the CVX solvers have subpar performance in terms of time in the small-scale experiment, we focus on the linear search methods in this test. From the figures, we see that L^2 natural gradient descent method is the slowest. The standard Newton-CG and standard modified Newton-Krylov have good convergence results on one dataset but not the other. In contrast, the proposed scheme is very competitive on both datasets. Specifically, it has good initial convergence where the objective function value is up to an order of magnitude smaller than the second-best scheme in the first few iterations. Moreover, its results are comparable with the other methods in terms of final training error, training accuracy, validation accuracy, and norm of gradient.

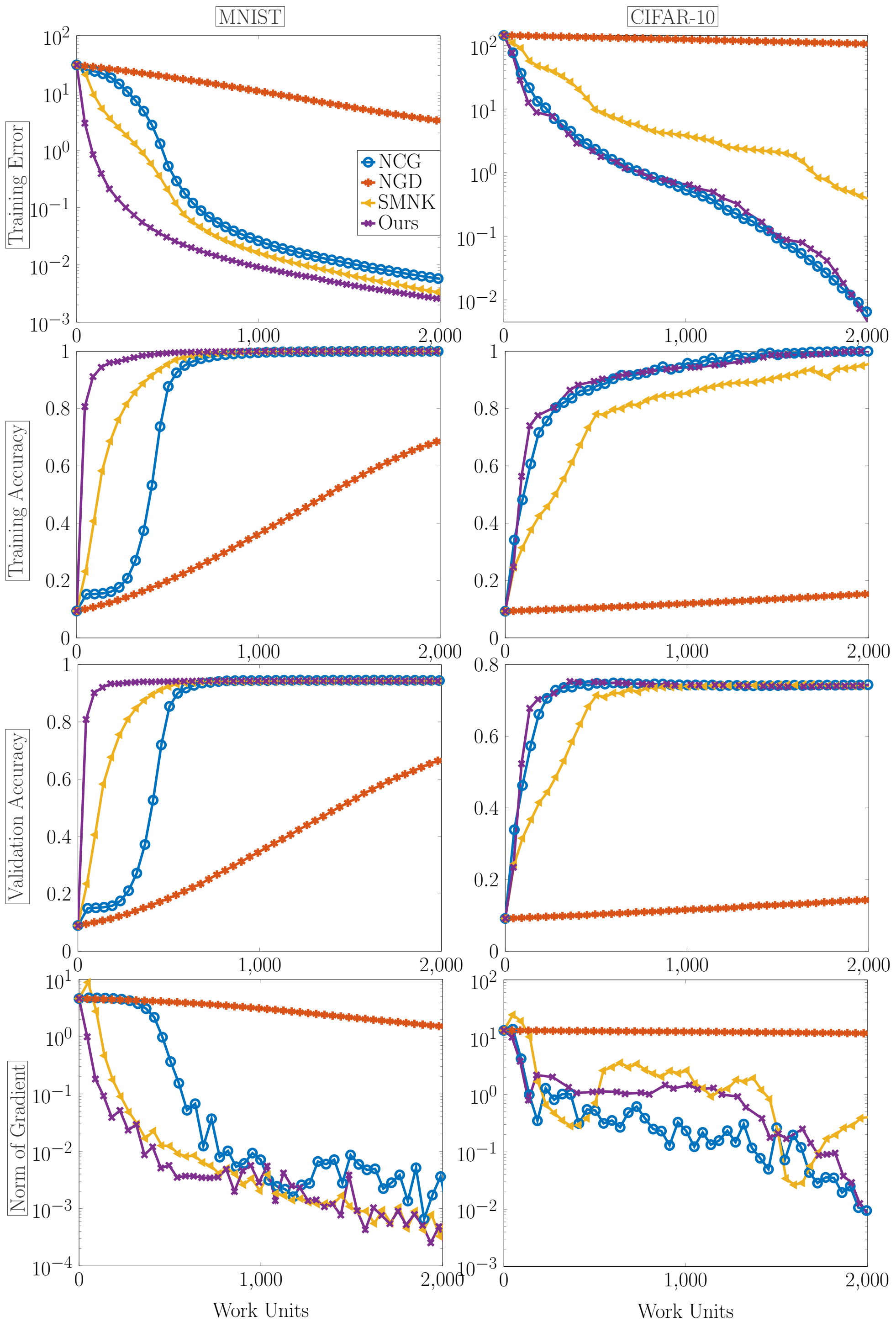


Figure 4.2: Experimental results on MLR without regularization. The x-axes report the number of work units. Here $n = 50,000$ training data and $10,000$ validation data are used.

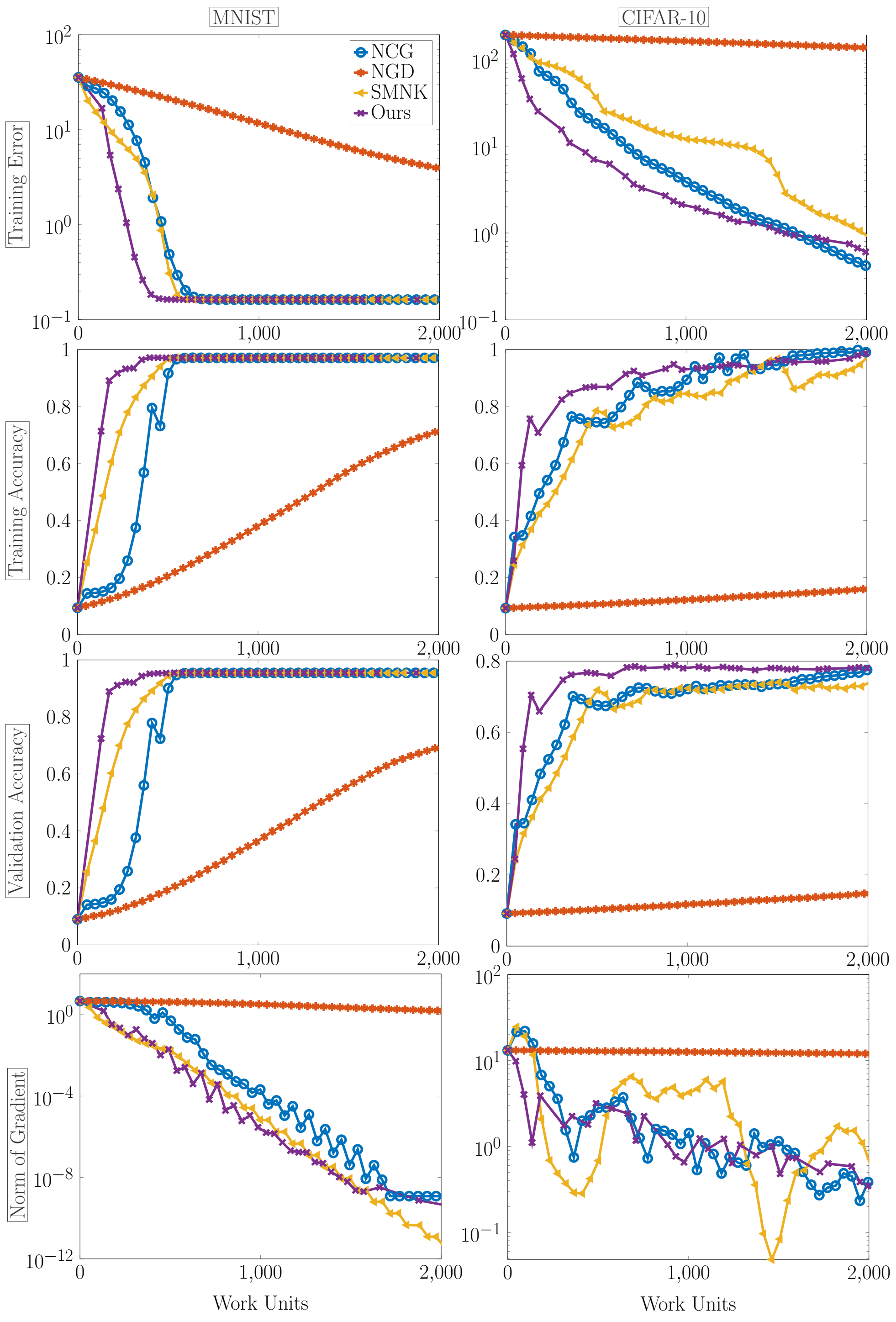


Figure 4.3: Experimental results on MLR with a Tikhonov regularization $\frac{\alpha}{2}\|\mathbf{w}\|_2^2$, with $\alpha = 10^{-3}$. The x-axes report the number of work units. Here $n = 50,000$ training data and $10,000$ validation data are used.

Table 4.2: Results on geometric programming. The final objective function value, norm of gradient, and total runtime are reported. Some results are not shown because the corresponding scheme fails to return a solution. The tests are run on an Apple Macbook Pro with a 10-core M1 Max CPU and 32 GBs of memory, and the software platform is MATLAB R2022a.

η		NCG	NGD	SM	Ours	SeDuMi	SDPT3	MOSEK
1e-6	f	–	2.03e+00	2.65e+00	1.77e+00	1.16e+00	–	–
	$\|\nabla f\|_2$	–	3.67e+00	3.40e+00	3.41e+00	7.13e-01	–	–
	Time	–	0.84s	1.16s	0.71s	1.54s	–	–
1e-4	f	–	1.94e+00	1.16e+00	1.16e+00	1.16e+00	1.16e+00	–
	$\|\nabla f\|_2$	–	1.90e+00	9.38e-13	2.72e-12	1.66e-04	3.94e-07	–
	Time	–	0.71s	0.77s	0.56s	1.47s	8.18s	–
1e-2	f	–	1.20e+00	1.18e+00	1.18e+00	–	1.18e+00	–
	$\|\nabla f\|_2$	–	1.10e-01	3.34e-14	7.31e-13	–	1.03e-10	–
	Time	–	0.65s	0.04s	0.06s	–	18.86s	–

4.5.3 Experiment 2: Geometric Programming

We consider a log-sum-exp minimization problem which commonly arises in geometric programming [140, 154, 157] and is used to test optimization algorithms [86, 124]. In particular, it is formulated as

$$\min_{\mathbf{w}} \eta \log\left(\mathbf{1}_m^\top \exp((\mathbf{J}\mathbf{w} - \mathbf{b})/\eta)\right),$$

where $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{J} \in \mathbb{R}^{m\times n}$, and η controls the smoothness of the problem. In particular, when $\eta \to 0$ the objective function converges to the point-wise maximum function $\max(\mathbf{J}\mathbf{w} - \mathbf{b})$ and its Hessian vanishes.

We follow the experimental setups in [86, 124], which use $m = 100$, $n = 20$, and generate the entries of $\mathbf{J}$ and $\mathbf{b}$ randomly. We perform the experiments with small values of η to test the robustness of the methods. In particular, we test with $\eta = 10^{-6}$, 10^{-4}, and 10^{-2}, respectively. We stop the line search iterative schemes after $25,000$ work units. The CG and Lanczos schemes stop when the relative residual drops below 10^{-3} or after 20 iterations.

The experimental results are shown in Table 4.2 and Figure 4.4. We see that the experiments are very challenging as the standard Newton-CG and all the CVX solvers cannot return a solution in some or all the experiments. In particular, the standard Newton-CG breaks in the first iteration in all the experiments. This is because the quadratic approximation is unbounded from below. Both SeDuMi and SDPT3 fail in one of the experiments. MOSEK fails in all the experiments. When the CVX solvers succeed in returning a solution, they have significantly longer runtime (up to 300 times slower) compared to the line search methods. Similar to the previous experiments, L^2 natural gradient descent method has the slowest convergence and has yet to converge after the specified work units. The standard modified Newton-Krylov and proposed methods are robust in the experiments and can return accurate solutions for $\eta = 10^{-4}$ and 10^{-2}, where the final norm of gradient is at most with magnitude 10^{-12}. This indicates the effectiveness of Hessian modification in handling challenging optimization problems. Moreover, the proposed method converges faster than the comparing standard modified Newton-Krylov method in the early stage. This indicates the effectiveness of the proposed Hessian modification over the standard one.

4.6 Summary

We present a modified Newton-Krylov algorithm tailored for optimizing the log-sum-exp function for a linear model. It has a simple implementation similar to that of a standard Newton-Krylov method and meanwhile is very effective. The novelty of our approach is incorporating a Hessian shift in the output space of the linear model. This does not change the optimal solutions, and renders the quadratic approximation to be bounded from below and the overall scheme to provably converge to a global minimum under standard assumptions. Since the update direction is computed using Krylov subspace methods which only require matrix-vector products with the linear model, the proposed method is applicable to large-scale problems. The numerical results show that our method is competitive in terms of accuracy, time-to-solution, and robustness with commonly-used solvers.

An interesting future direction is to extend the proposed scheme to log-sum-exp minimization for a nonlinear model. In each iteration, one can linearize the nonlinear model similarly to generalized Gauss-Newton methods, and thus the proposed scheme is applicable. Moreover, the proposed scheme can be interpreted as a trust-region Newton method where the trust region is defined by a semi-norm induced by the linear model. We can use schemes for determining the trust-region radius to select the parameter β_k.

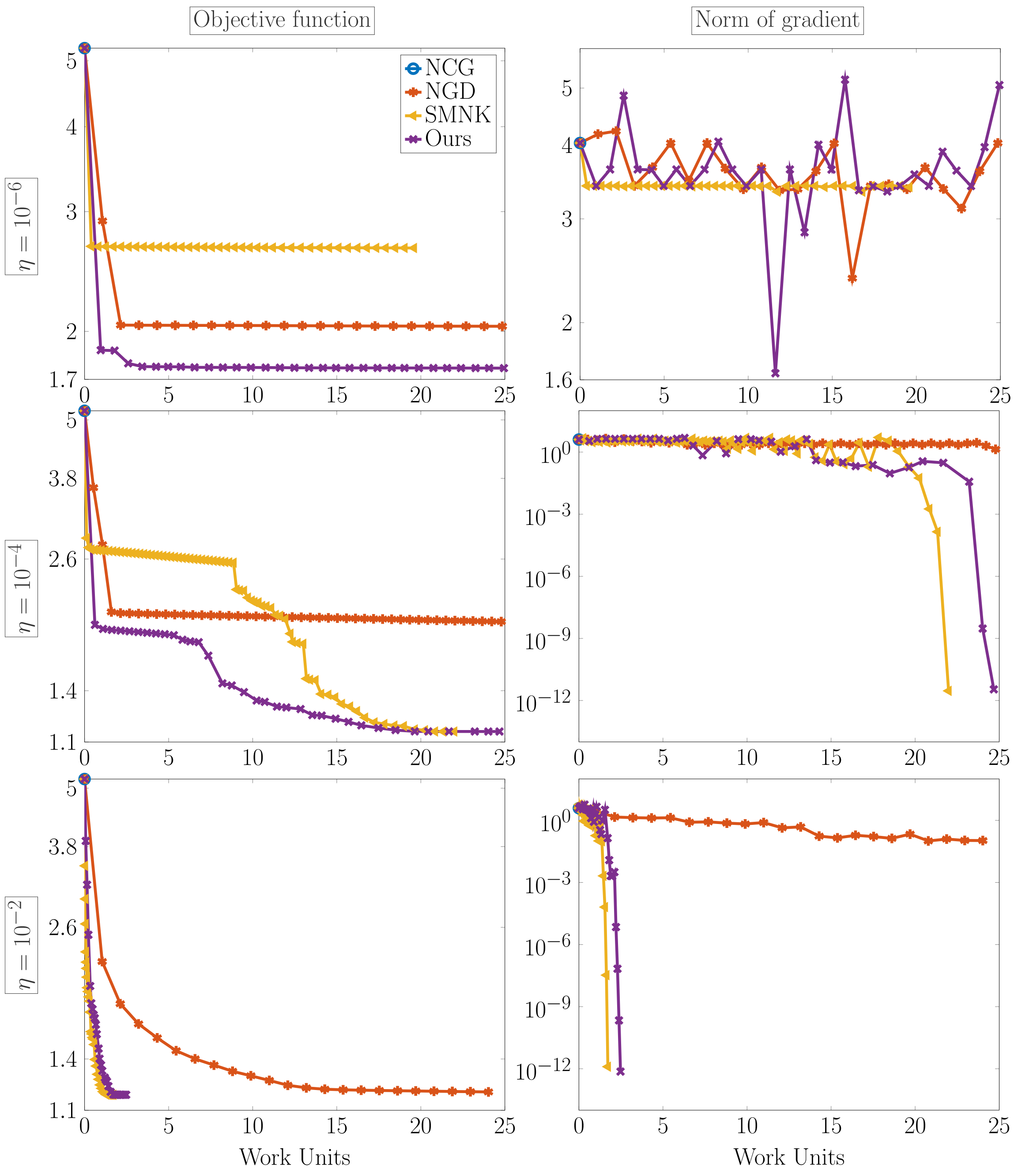


Figure 4.4: Experimental results on geometric programming. The x-axes report the number of work units (in thousands). The results for the standard Newton-CG scheme are not shown because it fails in the first iteration.

Chapter 5

Avoiding the Double Descent Phenomenon Using Hybrid Regularization

In this chapter, we present a work based on [82]. It was done in collaboration with James Nagy and Lars Ruthotto. This work concerns the double descent phenomenon arising in the least-squares fitting for the random feature model (RFM). The hallmark feature of the double descent phenomenon is a spike in the regularization gap at the interpolation threshold where the number of features of the RFM equals to the number of training samples. We adopt an inverse problems perspective and show that the double descent phenomenon can be explained by the ill-posedness of the learning problem. The ill-posedness can be addressed with effective techniques in inverse problems, such as iterative or Tikhonov regularization. However, they require parameter tuning, which typically requires solving the problem multiple times during a grid search and a dedicated validation dataset. In this work, we apply a hybrid scheme, which combines iterative and Tikhonov regularization methods in a synergized manner. Since in each iteration an increasingly accurate low-rank approximation to the data is built, the parameters are automatically and efficiently chosen by solving a low-dimensional parameter searching problem. While the benefits of hybrid methods have been well-documented for ill-posed inverse problems, to the best of our knowledge, this work presents the first use case in machine learning. We perform extensive experiments using the MNIST and CIFAR datasets and compare the effectiveness of iterative, Tikhonov, and hybrid regularization for small, medium, and large RFM models. In those examples, the hybrid scheme successfully avoids the double descent phenomenon. Also, it yields RFMs whose generalization error is comparable with that of classical regularization approaches even when their parameters are tuned to minimize the test loss, a procedure that is to be avoided in realistic applications.

This chapter is organized as follows. Firstly, we review related work and outline our main contributions. Secondly, we set up the training problem in RFMs, describe the double descent phenomenon, and relate it to the ill-posedness of the training problem. Thirdly, we demonstrate that - with a proper choice of parameters - iterative and Tikhonov regularization can effectively regularize the RFM training and avoid the double descent phenomenon. Fourthly, we present the hybrid regularization scheme and show its ability to avoid the double descent phenomenon without requiring parameter tuning. We then conclude the chapter by discussing and comparing the numerical results of the different schemes.

5.1 Related Work

The double descent phenomenon relates the generalization error of the random feature models (RFM) [131] to the number of random features. It is observed experimentally for least-squares fitting problems [7, 1]. In short, it manifests as a decreasing generalization error as the number of random features grows, a sharp spike when it reaches the number of training samples, followed by a decaying generalization error as the number of features is increased further.

The theoretical understanding of the double descent phenomenon has matured considerably in the last few years. In particular, [7] studies the phenomenon with a data-fitting perspective. In [1, 104], the generalization dynamics were analyzed for gradient flows, and iterative regularization was identified as a practical remedy; however, the success of this regularization requires a judicious choice of the stopping time by the user. Statistical analyses of the double descent phenomenon are provided in [8, 72]. A rigorous analysis of the asymptotic behavior of the test errors is performed in [108]. Bounds on the condition number of RFM matrices are derived in [22]. The phenomenon also arises in more general settings. For instance, the double descent pheonemon in neural network models training is studied in [51, 116]. The phenomenon for binary linear classification with different losses is analyzed in [35, 88]. A theoretical analysis of the double descent phenomenon in high dimensional kernel ridge regression problems is given in [103]. In [30], the double descent phenomenon is utilized in distillation to train teacher and student models that avoid overfitting.

As a new remedy to tackle the double descent phenomenon, we apply a computationally efficient hybrid method [27] that automatically tunes its parameters and avoids the double descent phenomenon. The development of hybrid methods has been a fruitful and important direction in inverse problems recently [25, 49, 26, 24]. We use the hybrid method implemented in [50], which combines the respective strengths of iterative regularization and Tikhonov regularization. Specifically, the method performs a few iterations of the numerically stable Krylov subspace method LSQR [125, 126] and adaptively selects the parameter for Tikhonov regularization at each iteration using generalized cross-validation (GCV) [57]. Notably, the scheme does not require a dedicated validation set. The effectiveness and the scalability of the method have been documented for various large-scale imaging problems [27, 25, 26].

5.2 Double Descent Phenomenon

In this section, we describe the problem setup for RFM, introduce the data used in our experiments, explain the double descent phenomenon, and relate the double descent phenomenon to ill-posed inverse problems.

Random Feature Model We consider a supervised learning problem where we are given the matrix of input features $\mathbf{Y} \in \mathbb{R}^{n \times n_\mathrm{f}}$ and the matrix of corresponding outputs $\mathbf{C} \in \mathbb{R}^{n \times n_\mathrm{c}}$. Here, n_f is the number of input features and n_c is the number of output features (e.g., the number of classes), and the n examples are stored row-wise. The idea in RFM is to transform the input features by applying a random nonlinear transformation $f : \mathbb{R}^{n_\mathrm{f}} \to \mathbb{R}^m$ to each row in $\mathbf{Y}$ and then train a linear model to approximate the relationship between $f(\mathbf{Y})$ and $\mathbf{C}$. Here, the transformation $f(\mathbf{Y})$ is applied row-wise and yields a new representation of the features in $\mathbb{R}^{n \times m}$. The dimension m controls the expressiveness of the RFM and can be chosen arbitrarily; generally, larger values of m increase the expressiveness of the RFM.

Similar to [104] we define our RFM using a randomly generated matrix $\mathbf{K} \in \mathbb{R}^{n_\mathrm{f} \times (m-1)}$, a bias vector $\mathbf{b} \in \mathbb{R}^{m-1}$, and an activation function $a : \mathbb{R} \to \mathbb{R}$, as

$$\mathbf{Z} = f(\mathbf{Y}) = \begin{bmatrix} a(\mathbf{Y}\mathbf{K} + \mathbf{1}_n \mathbf{b}^\top) & \mathbf{1}_n \end{bmatrix}, \tag{5.1}$$

where the activation function is applied element-wise, and $\mathbf{1}_n \in \mathbb{R}^n$ is a vector of all ones used to model a bias term.

The RFM training consists of finding the linear transformation $\mathbf{W} \in \mathbb{R}^{m \times n_\mathrm{c}}$ such that $\mathbf{Z}\mathbf{W} \approx \mathbf{C}$. As in [104], we measure the quality of the model using the least squares loss function. The double descent phenomenon arises when the weights are obtained by solving the unregularized regression problem, i.e.,

$$\mathbf{W}_{\text{LS}}^* \in \underset{\mathbf{W} \in \mathbb{R}^{m \times n_\text{c}}}{\arg\min} \frac{1}{2n} \|\mathbf{Z}\mathbf{W} - \mathbf{C}\|_\text{F}^2, \tag{5.2}$$

where $\|\cdot\|_\text{F}$ is the Frobenius norm. Problem (5.2) is separable, which means that it can be decoupled into n_c least-squares problems each of which determines one column of $\mathbf{W}_{\text{LS}}^*$. Therefore, without loss of generality, we focus our discussion on the case $n_\text{c} = 1$ and consider the problem

$$\mathbf{w}_{\text{LS}}^* \in \underset{\mathbf{w} \in \mathbb{R}^{m}}{\arg\min} \frac{1}{2n} \|\mathbf{Z}\mathbf{w} - \mathbf{c}\|_2^2. \tag{5.3}$$

Here, $\mathbf{c} \in \mathbb{R}^n$ are the output labels. For example, when choosing $\mathbf{c}$ as the ith column $\mathbf{C}$ the solution of (5.3) is the ith column of $\mathbf{W}_{\text{LS}}^*$.

The actual goal in learning is not necessarily to optimally solve (5.3) but to obtain an RFM model that generalizes beyond the training data. To gauge the generalization of the model defined by $\mathbf{w}_{\text{LS}}^*$, consider the test data set given by $\mathbf{Y}_\text{test} \in \mathbb{R}^{n_\text{test} \times n_\text{f}}$ and $\mathbf{C}_\text{test} \in \mathbb{R}^{n_\text{test} \times n_\text{c}}$. Then, the model defined by $\mathbf{w}_{\text{LS}}^*$, $\mathbf{K}$, and $\mathbf{b}$ generalizes well if the generalization gap

$$\frac{1}{2n_\text{test}} \|f(\mathbf{Y}_\text{test})\mathbf{w}_{\text{LS}}^* - \mathbf{c}_\text{test}\|_2^2 - \frac{1}{2n} \|f(\mathbf{Y})\mathbf{w}_{\text{LS}}^* - \mathbf{c}\|_2^2 \tag{5.4}$$

is sufficiently small. The main goal of our paper is to understand why solutions to (5.3) may fail to generalize and how to regularize (5.3) such that its solution reliably approximates the input-output relation for unseen data.

Examples from Image Classification Throughout the paper, we will use two common image classification benchmarks to illustrate the techniques. Specifically, we use the MNIST dataset consisting of 28×28 grayscale images of hand-written

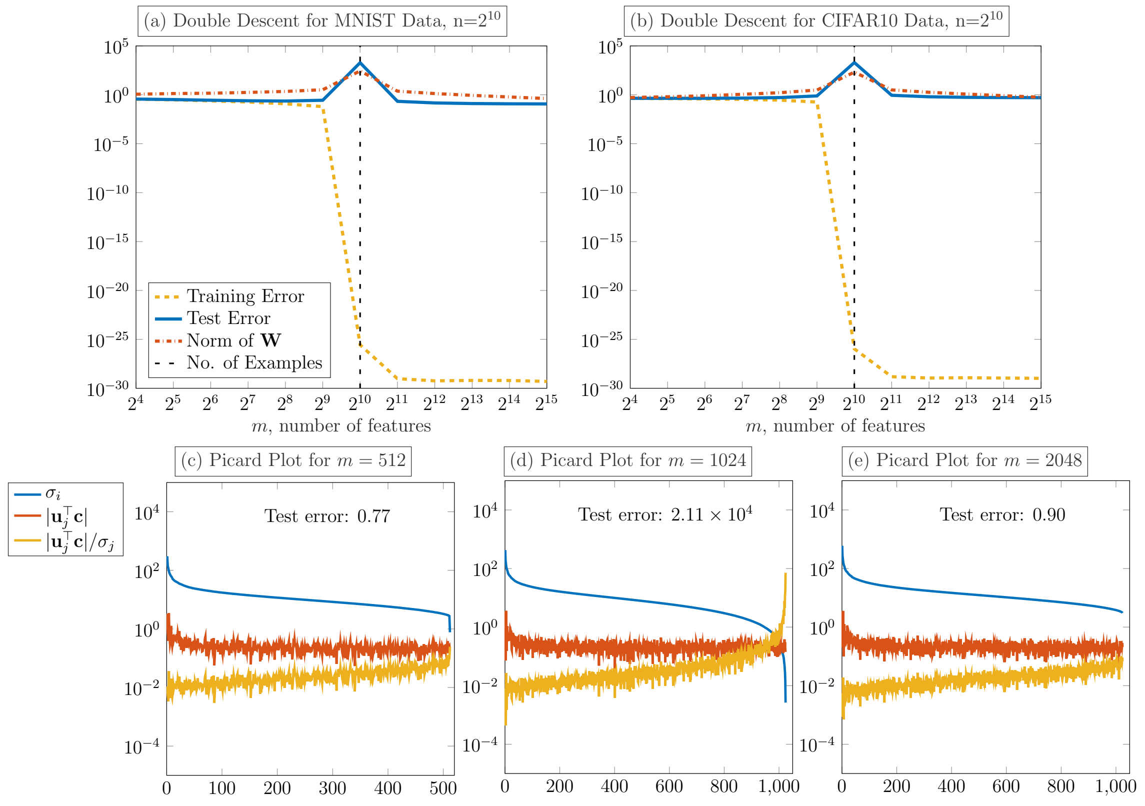


Figure 5.1: (a)-(b): The double descent phenomenon observed in random feature model on MNIST and CIFAR10 data. (c)-(e): The Picard plot for $\mathbf{Z}$ and $\mathbf{c}$ on CIFAR10 data with $n = 1024$ and $m = 512$ (overdetermined), 1024 (unique) and 2048 (underdetermined). All the values are averaged over 5 random trials. The top, middle and bottom curves are σ_j, $|\mathbf{u}_j^\top \mathbf{c}|$ and $|\mathbf{u}_j^\top \mathbf{c}|/\sigma_j$ defined in (5.5), respectively. When $m = n$, there is a plummet in σ_j at the end, and it renders $|\mathbf{u}_j^\top \mathbf{c}|/\sigma_j$ large. These large values dominate the optimal solution $\mathbf{W}$. Thus, there is a spike in the norm of $\mathbf{W}$ and hence the testing loss.

digits [98] and the CIFAR 10 dataset [92] consisting of 32×32 RGB images of objects divided into one of ten categories. From each dataset, we randomly sample $n = 1,024$ training images and their labels.

Each dataset also contains $n_{\text{test}} = 10,000$ labeled test images, which we use to compute the generalization gap of the trained model. To obtain competitive baseline results for our method we also use the test images to optimize the parameters of state-of-the-art methods. It is important to emphasize that our hybrid method selects parameters for regularization and stopping criteria automatically and hence does not use the test data.

In our experiments, we use the ReLU activation function $a(x) = \max(x, 0)$ and generate $\mathbf{K}$ and $\mathbf{b}$ using a uniformly random distribution drawn from a unit sphere; see also [104].

The Double Descent Phenomenon The key parameter of an RFM is the dimensionality of the feature space, m. The double descent phenomenon [7, 104, 51] is observed in the generalization gap for different choices m. This behavior can be described in three stages: when $m < n$, $m = n$ and $m > n$, see Figure 5.1(a)-(b). In the following, we explain these three stages using a data fitting viewpoint.

- When $m < n$, the learning problem (5.3) is overdetermined. That is, there are more equations than variables in $\mathbf{Zw} = \mathbf{c}$. There is no solution to perfectly describe the input-output relation in general. Hence as m increases, we are able to fit both the training and test data better, and the generalization gap decreases.

- When $m = n$, $\mathbf{Z}$ is square and, in our experience, invertible. Thus the optimal solution to (5.3) is unique and satisfies $\mathbf{Zw} = \mathbf{c}$. In order words, the training data can be fitted perfectly, and the training loss is essentially zero. However, the uniqueness of the optimal $\mathbf{w}$ also implies that we have no choice but to perfectly fit to the weakly present features in $\mathbf{Z}$, which are not relevant to the classification. The perfect loss on the training data combined with an increase in test loss then causes a spike in the generalization gap.

- When we further increase m such that $m > n$, the problem is underdetermined, and there are infinitely many optimal $\mathbf{w}$ to achieve an objective function value of zero. It has been observed that selecting the solution with the minimal norm reduces the risk of fitting weakly present features; see, e.g., [7]. Therefore, generally, the generalization gap decreases as m grows.

Double Descent and Ill-Posedness To better understand and avoid the double descent phenomenon, we use its connections to ill-posed inverse problems [41, 69, 71]. Consider the singular value decomposition (SVD) $\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Here, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m]$ are orthogonal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{n\times m}$ contains the singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$ in descending order on its diagonal and is zero otherwise. Let r be the rank of $\mathbf{Z}$, that is, the last index such that $\sigma_r > 0$. When the singular values decay to zero smoothly and without a significant gap, it is common to call problem (5.3) ill-posed, and a large generalization error has to be expected for some labels $\mathbf{c}$.

Using the SVD, the minimum norm solution of (5.3), can be written explicitly as

$$\mathbf{w}_{\text{LS}}^* = \sum_{j=1}^{r} \frac{\mathbf{u}_j^\top \mathbf{c}}{\sigma_j} \mathbf{v}_j. \tag{5.5}$$

This formulation shows that the contribution of $\mathbf{v}_j$ to $\mathbf{w}_{\text{LS}}^*$ is scaled by the ratio between $\mathbf{u}_j^\top \mathbf{c}$ and σ_j. If this ratio is large in magnitude, then the solution is highly sensitive to perturbations of the data $\mathbf{c}$ along the direction $\mathbf{u}_j$, and the corresponding singular vector $\mathbf{v}_j$ gets amplified in the solution. Also, σ_j quantifies the importance of the feature $\mathbf{u}_j$ in the data matrix $\mathbf{Z}$. Therefore, intuitively one wishes $|\mathbf{u}_j^\top \mathbf{c}|$ to decay as j grows. In other words, this observation also suggests that for ill-posed problems, the generalization gap depends on the decay of $|\mathbf{u}_j^\top \mathbf{c}|$.

We illustrate the quantities in (5.5) for the three stages of the double descent in Figures 5.1(c)-(e) using the CIFAR10 example. Here, we plot σ_j, $|\mathbf{u}_j^\top \mathbf{c}|$ and $|\mathbf{u}_j^\top \mathbf{c}|/\sigma_j$ in Figures 5.1(c)-(e); the resulting plot is known as a Picard plot [71]. From the decay of the singular values (see the blue line), we see that $m = n$ leads to an ill-posed problem as the σ_j decay to zero with no significant gap. Also, for $m = n$, the magnitude of $\mathbf{u}_j^\top \mathbf{c}$ (see red line) remains approximately constant. This combination causes a surge of $|\mathbf{u}_j^\top \mathbf{c}/\sigma_j|$ (see yellow line), which causes an increase of the norm of $\mathbf{w}$; see the red dashed line in Figure 5.1(a)-(b). When $m \neq n$, the problem is not ill-posed as the decay of the singular values is less pronounced. This correlation between the ill-posedness and the observed double descent phenomenon motivates us to employ state-of-the-art techniques for regularizing ill-posed inverse problems to improve the generalization of random feature models.

5.3 Iterative and Tikhonov Regularization

Regularization techniques are commonly used to improve the generalization of machine learning models (see, e.g., [60, Chapter 7]) and to enhance the solution of ill-posed inverse problems (see, e.g., [41, 69, 71]). Despite differences in notation and naming, the basic ideas in both domains are similar. This section aims to provide the background of the two most common forms of regularization: iterative regularization and Tikhonov regularization, respectively, which, in the machine learning literature, are better known as early stopping and weight decay, respectively. While the techniques in this section are standard, using hybrid approaches to train random feature models, as we discuss in the next section, is a novelty of our work.

Using the SVD of the feature matrix $\mathbf{Z} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, we can write and analyze most regularization schemes for (5.3) using their corresponding filter factors ϕ_j that control the influence of the jth term in (5.5). To be precise, the regularized solutions can be written as

$$\mathbf{w}_{\text{reg}} = \sum_{j=1}^{r} \phi_j \frac{\mathbf{u}_j^\top \mathbf{c}}{\sigma_j} \mathbf{v}_j. \tag{5.6}$$

A simple example is the truncated SVD, in which terms associated with small singular values are ignored by using the filter factors

$$\phi_{\text{TSVD},j}(\tau) = \begin{cases} 1, & \sigma_j > \tau \\ 0, & \sigma_j \leq \tau \end{cases}, \tag{5.7}$$

where the choice of $\tau \geq 0$ is crucial to trade off the reduction of the training loss and the regularity of the solution, which is needed to generalize. In the remainder of the section, we briefly review iterative regularization and Tikhonov regularization.

Iterative Regularization A common observation during the training of random feature models with iterative methods is an initially sharp decay of both the training and test losses, followed by a widening generalization gap in later iterations. For least-squares problems such as (5.3) this behavior, also known as semiconvergence, typically arises when the iterative method converges quicker on the subspace spanned by the singular vectors associated with large singular values than on those associated with small singular values. A straightforward and popular way to regularize the problem then is iterative regularization which stops the iteration early; see, e.g., [104] for RFMs, [156] for neural networks, and [27] for image recovery.

As a simple example to show the effect of iterative regularization, we consider the gradient flow (GF) applied to (5.3), which reads

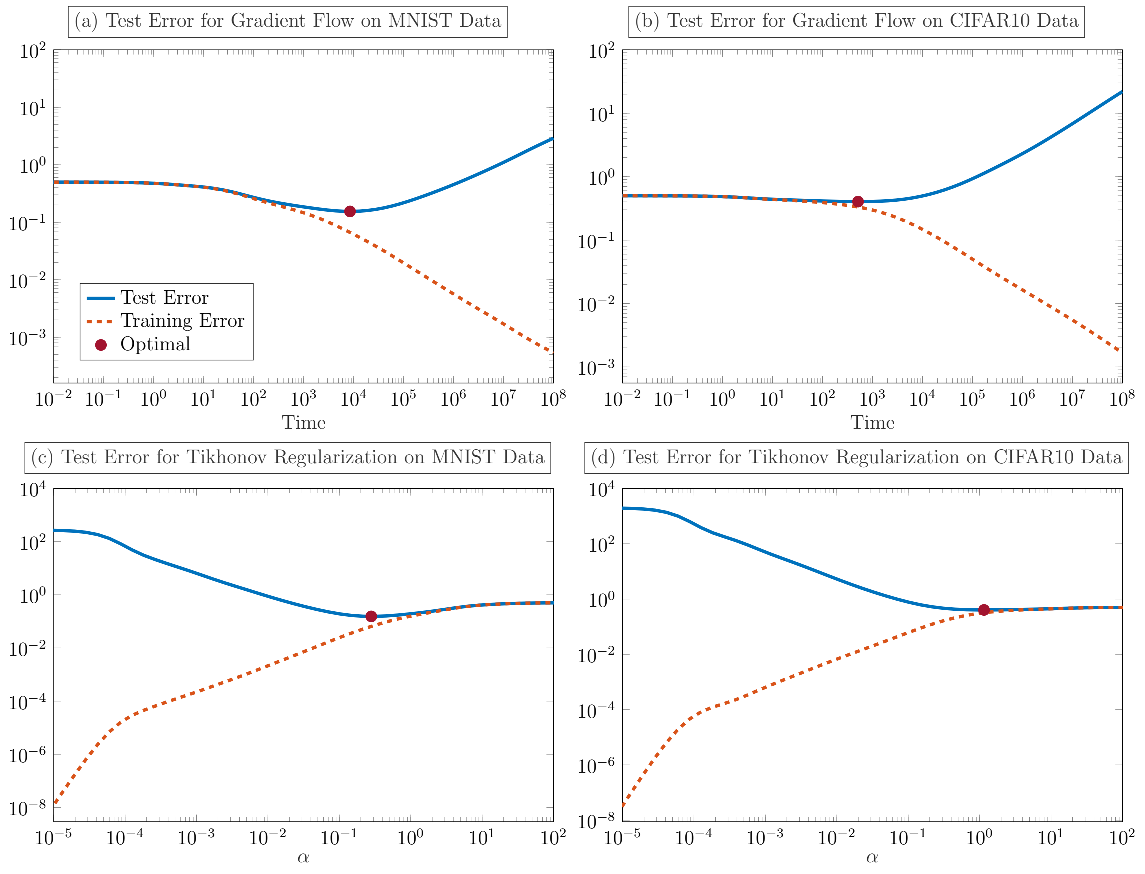
$$\partial_t \mathbf{w}_{\mathrm{GF}}(t) = -\frac{1}{n}\mathbf{Z}^\top(\mathbf{Z}\mathbf{w}_{\mathrm{GF}}(t) - \mathbf{c}), \quad \mathbf{w}_{\mathrm{GF}}(0) = \mathbf{0}. \tag{5.8}$$

As also shown in [104], when the SVD of the feature matrix is available, $\mathbf{w}_{\mathrm{GF}}(t)$ can be computed via (5.6) using the filter factors

$$\phi_{\mathrm{GF},j}(t) = 1 - e^{-\sigma_j^2 t/(mn)}. \tag{5.9}$$

From this observation, we can see that as t grows, the filter factors converge to one, and $\mathbf{w}_{\mathrm{GF}}(t)$ converges to the solution of the unregularized problem (5.3). Furthermore, we see that for any fixed time, the filter factors decay as j grows, which reduces the sensitivity to perturbations of the data $\mathbf{c}$ along the directions associated with small singular values. In the top row of Figure 5.2, we show numerical results for the

Figure 5.2: (a)-(b): The results of applying gradient flow (GF) to (5.2) when $m = n = 1024$ at different time t. (c)-(d): The results obtained from Tikhonov regularization (5.10) with different parameter α. The optimal stopping time/regularization parameter is highlighted. We can see that both methods exhibit a semiconvergence behavior. In particular, their generalizability depends on the choice of parameters, which varies from problem to problem and has to be made judiciously. Determining the optimal parameters requires access to the test data. Yet, in practice, test data are not allowed to use for training.



iterative regularization applied to our two test problems. The qualitative behavior is comparable for both datasets: Initially, both training and test losses decay with no noticeable gap but at later times, the test losses increase dramatically. A difference between the two datasets is that the optimal stopping time (i.e., the time with the smallest test loss) differs by about two orders of magnitude. Hence, the stopping time is the key parameter that needs to be chosen judiciously and depends on the problem. Determining an effective stopping time is even more difficult in realistic applications, when this decision must not be based on the test data set.

In addition to gradient descent and its stochastic versions, many other first-order optimization methods enjoy similar regularizing properties. In particular, Krylov subspace methods have been commonly used in ill-posed inverse problems due to their superior convergence properties; see, e.g., [69, 71].

The cross-validation of iterative regularization is easy to perform; for example, one can use a part of the training data for validation and stop the training process when the validation error is minimized. However, such an approach reduces the number of training data. The regularization properties and their analysis also depend heavily on the underlying iterative method. Moreover, iterative regularization generally prefers slowly converging schemes to have a broader range of optimal stopping points.

Tikhonov Regularization The idea in the direct regularization methods, known as Tikhonov regularization [41, 69] in inverse problems and weight decay in machine learning [60, Chapter 7], is to add an extra term to the objective in (5.3) such that the solution to the obtained regularized problem generalizes well. There are many options for choosing regularization terms. In our work, we consider the squared Euclidean norm of $\mathbf{w}$ and define the regularized solution as

$$\mathbf{w}_{\mathrm{WD}}(\alpha) = \arg\min_{\mathbf{w}} \frac{1}{2n}\|\mathbf{Z}\mathbf{w} - \mathbf{c}\|_2^2 + \frac{\alpha^2}{2}\|\mathbf{w}\|_2^2. \tag{5.10}$$

Here, the regularization parameter $\alpha \geq 0$ trades off minimization of the loss and the norm of $\mathbf{w}_{\mathrm{WD}}$. An advantage compared to iterative regularization is that any iterative or direct method used to solve (5.10) will ultimately provide the same solution.

Using the SVD of $\mathbf{Z}$ we can see that $\mathbf{w}_{\mathrm{WD}}(\alpha)$ can be computed using (5.6) and the filter factors

$$\phi_{\mathrm{WD},j}(\alpha) = \frac{\sigma_j^2}{\sigma_j^2 + n\alpha^2}, \tag{5.11}$$

which are also called the Tikhonov filter factors [69]. We can see that when α is chosen relatively small, the filter factors associated with large singular values remain almost unaffected, while those corresponding to small singular values may be close to zero. Thus, similar to TSVD and iterative regularization, Tikhonov regularization can reduce the sensitivity of $\mathbf{w}_{\mathrm{WD}}$ to perturbations of the data along the directions associated with small singular values.

We investigate the impact of choosing α on the generalization in a numerical experiment for the MNIST and CIFAR10 example; see Figure 5.2(c)-(d). The qualitative behavior is comparable for both datasets: as α increases, the training error increases monotonically, while the test error first decays and then finally grows. Due to this semiconvergence, a careful choice of α can improve generalization; we visualize the parameter α that yields the lowest test loss with red dots. A key difference between the examples is that the optimal values of α differ by about one order of magnitude, which highlights its problem-dependence. As in iterative regularization, we re-iterate that the test data must not be used to select the optimal value of α.

The solution of Tikhonov regularization has a simple representation (5.11). This renders its analysis simple. Moreover, the same solution is obtained regardless of the choice of solvers. Thus, in contrast to iterative regularization, the most efficient scheme can be used. Yet the optimal choice of the parameter α requires cross-validation [90] in which the problem has to be solved many times.

5.4 Hybrid Regularization: The Best of Both Worlds

Hybrid methods belong to the most effective solvers for ill-posed inverse problems and have been widely used, for example, in large-scale image recovery [27, 25, 26]. The key idea in hybrid methods is to combine the respective advantages of iterative and Tikhonov regularization while avoiding their disadvantages. In this section, we briefly review the technique `IRhybrid_lsqr` from the open source MATLAB package [50], and for the first time apply the scheme in the machine learning domain for training random feature models. This hybrid method employs LSQR [125, 126] that at each iteration projects the regularized least-squares problem (5.10) onto a small-dimensional subspace and adaptively selects the parameter for Tikhonov regularization using generalized cross-validation (GCV) [57]. The resulting hybrid method does not involve any parameter tuning and, in our experiments, successfully avoids the semiconvergence, hence, the double descent phenomenon.

LSQR Algorithm LSQR [125, 126] is an iterative method for solving (regularized) least-squares problems. With comparable computational costs per iteration, the numerical stability and convergence of LSQR generally are superior to gradient descent, particularly for ill-posed problems. The kth iteration of LSQR solves the projection of (5.10) onto the k-dimensional Krylov subspace

$$\mathcal{K}_k = \operatorname{span}\{\mathbf{Z}^\top \mathbf{c}, (\mathbf{Z}^\top \mathbf{Z})\mathbf{Z}^\top \mathbf{c}, \ldots, (\mathbf{Z}^\top \mathbf{Z})^{k-1}\mathbf{Z}^\top \mathbf{c}\}.$$

This projection is obtained using Golub-Kahan bidiagonalization [55] of the feature matrix $\mathbf{Z}$ with the initial vector $\mathbf{c}$, which reads

$$\mathbf{Z}^\top \mathbf{Q}_k = \mathbf{P}_k \mathbf{B}_k^\top + \gamma_{k+1}\mathbf{p}_{k+1}\mathbf{e}_{k+1}^\top, \tag{5.12}$$

$$\mathbf{Z}\mathbf{P}_k = \mathbf{Q}_k \mathbf{B}_k, \tag{5.13}$$

where $\mathbf{Q}_k \in \mathbb{R}^{n\times(k+1)}$ and $\mathbf{P}_k \in \mathbb{R}^{m\times k}$ have orthonormal columns, $\mathbf{B}_k \in \mathbb{R}^{(k+1)\times k}$ is a lower bidiagonal matrix, $\mathbf{e}_{k+1} \in \mathbb{R}^{k+1}$ is the $(k+1)$th standard basis vector, and γ_{k+1} and $\mathbf{p}_{k+1}$ will be the $(k+1)$th diagonal entry of $\mathbf{B}_{k+1}$ and the $(k+1)$th column of $\mathbf{P}_{k+1}$, respectively.

Using the bidiagonalization, we derive the projection of (5.10) as follows

$$\min_{\mathbf{x}\in\mathcal{K}_k} \frac{1}{2n}\|\mathbf{Z}\mathbf{x}-\mathbf{c}\|_2^2 + \frac{\alpha^2}{2}\|\mathbf{x}\|^2 = \min_{\mathbf{f}\in\mathbb{R}^k} \frac{1}{2n}\|\mathbf{Z}\mathbf{P}_k\mathbf{f}-\mathbf{c}\|_2^2 + \frac{\alpha^2}{2}\|\mathbf{f}\|^2 \tag{5.14}$$

where we used that the columns of $\mathbf{P}_k$ form an orthonormal basis of $\mathcal{K}_k$, which also implies that $\|\mathbf{P}_k\mathbf{f}\| = \|\mathbf{f}\|$. Next, using (5.13) gives

$$= \min_{\mathbf{f}\in\mathbb{R}^k} \frac{1}{2n}\|\mathbf{Q}_k\mathbf{B}_k\mathbf{f}-\mathbf{c}\|_2^2 + \frac{\alpha^2}{2}\|\mathbf{f}\|^2 \tag{5.15}$$

Using the orthonormality of the columns of $\mathbf{Q}_k$ and the fact that $\mathbf{Q}_k$ contains $\frac{\mathbf{c}}{\|\mathbf{c}\|_2}$ in its first column, we obtain the projected problem

$$= \min_{\mathbf{f}\in\mathbb{R}^k} \frac{1}{2n}\|\mathbf{B}_k\mathbf{f}-\beta\mathbf{e}_1\|_2^2 + \frac{\alpha^2}{2}\|\mathbf{f}\|^2, \tag{5.16}$$

where $\beta = \|\mathbf{c}\|_2$ and $\mathbf{e}_1 \in \mathbb{R}^{k+1}$ is the first standard basis vector. Here, the k-dimensional projected problem (5.16) is greatly reduced in size compared to the original m-dimensional problem (5.10). The kth iteration of LSQR is the solution to the projected problem and can be computed using the regularized pseudoinverse $\mathbf{B}_{k,\alpha}^{\dagger}$ via

$$\mathbf{P}_k\mathbf{f}_\alpha = \beta\mathbf{P}_k\mathbf{B}_{k,\alpha}^{\dagger}\mathbf{e}_1 \quad \text{with} \quad \mathbf{B}_{k,\alpha}^{\dagger} = \frac{1}{n}\left(\frac{1}{n}\mathbf{B}_k^\top\mathbf{B}_k + \alpha^2\mathbf{I}\right)^{-1}\mathbf{B}_k^\top. \tag{5.17}$$

Automatic Tikhonov Regularization In Tikhonov regularization, the choice of the parameter α is crucial in order to successfully filter the small singular values. One straightforward way to choose it is to perform cross-validation. Having the small-dimensional projected problem (5.16) allows us to test multiple candidate α's for cross-validation efficiently. Here, the parameter selection can be done even more effectively by using statistical criteria such as generalized cross-validation (GCV) [57, 41, 69, 144], weighted GCV [27], L-Curve [17] and discrepancy principle [144]. In this paper, we use GCV for simplicity.

The idea of GCV is to pick a parameter for Tikhonov regularization that gives good generalization power. Specifically, it performs n-fold (leave-one-out) cross-validation [90] without solving the problem n times by minimizing a loss function on the training data. Thus, it does not require validation data and is done highly efficiently using the low-rank projected solution (5.17).

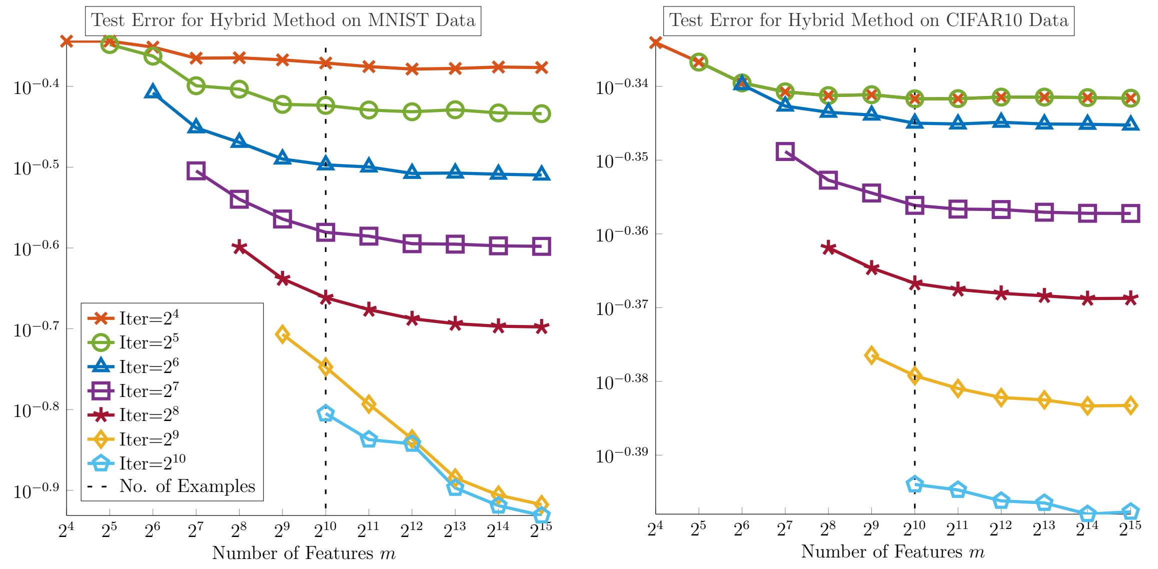
In particular, in each iteration of the hybrid scheme, we minimize the GCV function for the projected problem (5.16) given by

$$G_{\mathbf{B}_k,\beta\mathbf{e}_1}(\alpha) = \frac{k\|(\mathbf{I} - \mathbf{B}_k\mathbf{B}_{k,\alpha}^{\dagger})\beta\mathbf{e}_1\|_2^2}{(\mathrm{trace}(\mathbf{I} - \mathbf{B}_k\mathbf{B}_{k,\alpha}^{\dagger}))^2}. \tag{5.18}$$

Here, the SVD of $\mathbf{B}_k$ is performed quickly because it is small in size ($(k+1)$-by-k). We can then plug the SVD into (5.18). The minimization will become a simple one-dimensional problem and can be done by standard algorithms. This renders the GCV minimization effective, which needs to be done in each iteration. In principle, we can compute the GCV also for the full problem (5.10). However, this would be computationally very expensive because the full SVD of $\mathbf{Z}$ is required. For more details, see [27].

Advantages of the Hybrid Method The regularization imposed by the hybrid method provides important distinct advantages over previously discussed approaches. First, in (5.18), the hybrid method performs an adaptive Tikhonov regularization by dynamically selecting the parameter using information from the small (but increasing) dimension Krylov subspace. Specifically, the hybrid method chooses the Tikhonov filter factors in (5.6) based on the singular values of the projected problem (5.18), and these singular values are increasingly better approximations of the full dimension problem (5.10). Secondly, one can also use the GCV function value as criteria for iterative regularization, see [27, 11]. This effectively employs a safeguard regulariza-

Figure 5.3: The results obtained by the hybrid method with different numbers of iterations on MNIST and CIFAR10 data. The results when the iteration number is greater than m are not shown as the algorithm converges after m iterations. This is because the Krylov subspace is $\mathbb{R}^m$, and the projected problem becomes the original problem.



tion. However, the Tikhonov filter factors computed in the hybrid method ensure that the computed solutions are much less sensitive to the precise stopping iteration. This combination of safeguarded regularization, automatic tuning of parameters for Tikhonov regularization, and automatic iteration stopping criteria is very powerful.

Numerical Results We apply the hybrid method to the image classification problems. To demonstrate that the hybrid method avoids semi-convergence, we show the test error for an increasing number of iterations in Figure 5.3. For all m and for both datasets, we see that increasing the number of iterations reduces the test errors until the number of iterations reaches m when the bidiagonalization is exact. In particular, the hybrid scheme avoids the double descent phenomenon when $m = n$. In contrast to iterative and Tikhonov regularization, the parameter α of the hybrid method is automatically chosen in each iteration. We recommend choosing the number of iterations to match the computational budget.

5.5 Comparison and Discussion

In this section, we compare and discuss the numerical results achieved with the different regularization schemes.

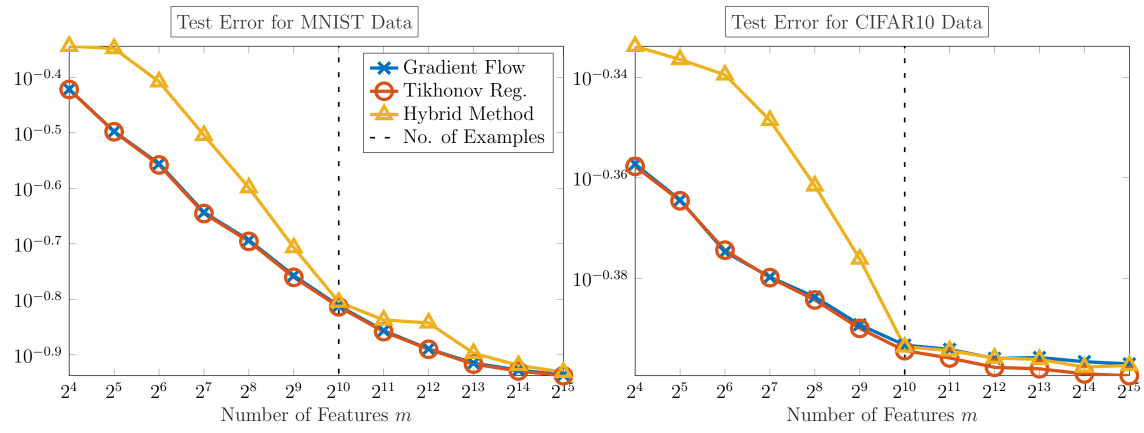
Baseline We optimize the parameters for the iterative and Tikhonov regularization presented in Section 5.3 using the test data. It is important to emphasize that this is neither practical nor advised in realistic applications. However, our goal is to obtain competitive baselines to compare with the hybrid scheme in which neither the test data nor some validation data is used.

We optimize the weights for each of the datasets and different widths of the RFM. To this end, we compute the (economic) SVD of the feature matrix $\mathbf{Z}$ and use the filter factors in (5.9) and (5.11), respectively, to efficiently compute the optimal weights for different choices of t and α. Then, we minimize the test error over the parameters using the one-dimensional optimization method `fminsearch` in MATLAB. To reduce the risk of being trapped in a suboptimal local minimum, we first evaluate the test loss at 100 points spaced equally on the logarithmic axes shown in Figure 5.2 and initialize the optimization method at the parameter with the lowest test loss.

Comparison We report the results for the different datasets, different m, and regularization schemes in Figure 5.4. For the hybrid method, we set the number of iterations to $\min(m, 1024)$; see also Figure 5.3. We see that the hybrid method achieves competitive test errors even though it does not use the test data set. Remarkably, the hybrid method's solution is on par with the other schemes at the interpolation threshold.

Discussion These numerical experiments demonstrate the potential of hybrid methods to reliably train random feature models of various sizes with automatic parameter tuning.

Figure 5.4: The results obtained by gradient flow, Tikhonov regularization and the hybrid method. For gradient flow and Tikhonov regularization, the optimal testing losses over time and α, respectively, are reported. Specifically, we minimize the testing losses with respect to the parameters. For the hybrid method, we determine the Tikhonov parameters using the training data only and report the test loss with $\min(m, 1024)$ iterations.



Most practical implementations of iterative regularization and Tikhonov regularization use parts of the training samples to tune parameters using (one-fold) cross-validation. This has the disadvantage of reducing the number of samples available during training, which generally leads to larger test errors. By contrast, the hybrid scheme implicitly performs an n-fold (leave-one-out) cross-validation.

As shown in Figure 5.3, the test error of the hybrid scheme decreases with the number of iterations. This is in stark contrast to the gradient flow scheme (see Figure 5.2) for which semiconvergence is observed, and an adequate stopping rule is needed to avoid large generalization errors.

In classical Tikhonov regularization, the learning problem has to be solved repeatedly until a regularization parameter α with a low validation error is found. The hybrid scheme saves this computational overhead and automatically selects α in one single pass. This is achieved by using the Golub-Kahan bidiagionalization to evaluate the GCV function efficiently for different values of α.

5.6 Summary

In this chapter, we present hybrid methods and show in numerical experiments that they avoid the double descent phenomenon arising in the training of random feature models. We demonstrate that the double descent phenomenon is related to the ill-posedness of the training problem. As such, it can be overcome with iterative regularization and Tikhonov regularization; however, these techniques typically require cumbersome parameter tuning, solutions of multiple instances of the learning problem, and a dedicated validation set. Hybrid methods overcome these disadvantages. They are computationally efficient thanks to stopping early and performing parameter searching on a low-dimensional subspace. Further, they automatically select parameters using statistical criteria. While these properties have made hybrid methods increasingly popular for solving large-scale inverse problems, our paper presents the first use case in machine learning. In our experiments, the hybrid method performs competitively to classical regularization methods with optimally chosen weights. In future works, we plan to extend hybrid methods to more general learning problems, particularly with other loss functions and machine learning models. We provide our MATLAB codes for the numerical experiments at `https://github.com/EmoryMLIP/HybridRFM`.

Chapter 6

Conclusion and Future Work

In this dissertation, we present three low-rank exploiting optimization methods for inverse problems and machine learning. These methods fall under two categories: Newton-type algorithms that exploit the low-rank approximation to the Hessian matrix and hybrid regularization methods that exploit the low-rank structures of data matrices.

In Chapter 3, we present PNKH-B for solving large-scale bound-constrained optimization. PNKH-B is a generalized one-metric scheme that effectively exploits the low-rank approximation to the Hessian matrix to compute the projection in a tractable way. A global convergence proof of PNKH-B is provided. Three numerical experiments on PDE parameter estimation, image classification, and image reconstruction demonstrate the effectiveness of PNKH-B.

In Chapter 4, we present a modified Newton-Krylov scheme geared toward log-sum-exp minimization for a linear model. The proposed scheme is applicable to large problem sizes since a Krylov subspace method is used to construct a low-rank approximation to the modified Hessian. The main novelty in the proposed scheme is a Hessian modification in the output space of the linear model. We prove that the proposed scheme globally converges to an optimal solution. Two numerical experiments on multinomial logistic regression and geometric programming illustrate that the proposed scheme has competitive performance.

In Chapter 5, we shift our focus and apply a hybrid regularization method to avoid the double descent phenomenon arising in the training of random feature models. Hybrid regularization methods synergistically combine iterative and Tikhonov regularization so that their respective strengths are utilized, and their respective drawbacks are avoided. Our analyses show that the double descent phenomenon is caused by the ill-posedness of the training problem. This motivates the use of hybrid regularization methods. Extensive numerical experiments show the effectiveness of the hybrid regularization method.

The work in this dissertation paves the way for future research directions. In addition to the future directions of each individual method discussed at the end of their respective chapters, we could also combine these applications. For instance, we could incorporate the Hessian shift in Chapter 4 into PNKH-B and investigate the performance on PDE-constrained optimization problems [123] and the log-sum-exp minimization problem in Chapter 4. We could explore using bound constraints to avoid the double descent phenomenon since they can serve as a regularization method. In this setting, PNKH-B could be applied to effectively solve the bound-constrained optimization problem.

Appendix A

Derivation of Gradient and Hessian of MLR Problem

Here we provide a derivation of the gradient and Hessian of the multinomial logistic regression (MLR) problem. We recall that for given input features and target outputs $\{\mathbf{a}_k, \mathbf{c}_k\}_{k=1}^n \subset \mathbb{R}^m \times \Delta_{n_c}$, the MLR problem is formulated as

$$\begin{aligned}\min_{\mathbf{W}\in\mathbb{R}^{n_c\times m}} F(\mathbf{W}) &= -\frac{1}{n}\sum_{k=1}^{n} \mathbf{c}_k^\top \log\left(\frac{\exp(\mathbf{W}\mathbf{a}_k)}{\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}_k)}\right) \\ &= \frac{1}{n}\sum_{k=1}^{n}\left[(\mathbf{c}_k^\top \mathbf{1}_{n_c})\log\left(\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}_k)\right) - \mathbf{c}_k^\top \mathbf{W}\mathbf{a}_k\right] \qquad \text{(A.1)} \\ &= \frac{1}{n}\sum_{k=1}^{n}\left[\log\left(\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{a}_k)\right) - \mathbf{c}_k^\top \mathbf{W}\mathbf{a}_k\right].\end{aligned}$$

Here Δ_{n_c} is the n_c-dimensional unit simplex, $\mathbf{1}_{n_c} \in \mathbb{R}^{n_c}$ is the vector of all ones, the log operation is applied element-wise, and we use the fact that $\mathbf{c}_k^\top \mathbf{1}_{n_c} = 1$ since $\mathbf{c}_k \in \Delta_{n_c}$.

By concatenating the data, the MLR problem (A.1) can be re-written as

$$\min_{\mathbf{W}\in\mathbb{R}^{n_c\times m}} F(\mathbf{W}) = \frac{1}{n}\left[\log\left(\mathbf{1}_{n_c}^\top \exp(\mathbf{W}\mathbf{A})\right) - \mathbf{1}_{n_c}^\top\left(\mathbf{C}\odot\mathbf{W}\mathbf{A}\right)\right]\mathbf{1}_n, \qquad \text{(A.2)}$$

where $\odot$ denotes the Hadamard (element-wise) product, and

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n] \in \mathbb{R}^{m\times n}, \mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n] \in \mathbb{R}^{n_c\times n}.$$

We then vectorize the the parameters $\mathbf{w} = \text{vec}(\mathbf{W})$ so that (A.2) becomes

$$\min_{\mathbf{w}\in\mathbb{R}^{mn_c}} f(\mathbf{w}) = \frac{1}{n}\Big[\mathbf{1}_n^\top \log\big((\mathbf{I}_n \otimes \mathbf{1}_{n_c}^\top)\exp(\mathbf{J}\mathbf{w})\big) - \mathbf{c}^\top\mathbf{J}\mathbf{w}\Big],$$

where

$$\mathbf{c} = \text{vec}(\mathbf{C}) \in \mathbb{R}^{nn_c}, \quad \text{and} \quad \mathbf{J} = \mathbf{A}^\top \otimes \mathbf{I}_{n_c} \in \mathbb{R}^{nn_c\times mn_c},$$

where $\mathbf{I}_{n_c}$ $\mathbb{R}^{n_n\times n_c}$ is the identity matrix.

Letting $\mathbf{K} = \mathbf{I}_n \otimes \mathbf{1}_{n_c}\mathbf{1}_{n_c}^\top$, the gradient and Hessian are given by

$$\begin{aligned}
\nabla f(\mathbf{w}_i) &= \frac{1}{n}\left[\mathbf{J}^\top\text{diag}(\exp(\mathbf{J}\mathbf{w}_i))(\mathbf{I}_n \otimes \mathbf{1}_{n_c})\text{diag}\left(\frac{1}{(\mathbf{I}_n \otimes \mathbf{1}_{n_c}^\top)\exp(\mathbf{J}\mathbf{w}_i)}\right)\mathbf{1}_n - \mathbf{J}^\top\mathbf{c}\right] \\
&= \frac{1}{n}\mathbf{J}^\top\left[\text{diag}(\exp(\mathbf{J}\mathbf{w}_i))(\mathbf{I}_n \otimes \mathbf{1}_{n_c})\frac{1}{(\mathbf{I}_n \otimes \mathbf{1}_{n_c}^\top)\exp(\mathbf{J}\mathbf{w}_i)} - \mathbf{c}\right] \\
&= \frac{1}{n}\mathbf{J}^\top\left[\text{diag}(\exp(\mathbf{J}\mathbf{w}_i))\frac{1}{(\mathbf{I}_n \otimes \mathbf{1}_{n_c}\mathbf{1}_{n_c}^\top)\exp(\mathbf{J}\mathbf{w}_i)} - \mathbf{c}\right] \\
&= \frac{1}{n}\mathbf{J}^\top\left[\frac{\exp(\mathbf{J}\mathbf{w}_i)}{\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i)} - \mathbf{c}\right] \\
&= \frac{1}{n}\mathbf{J}^\top\left(\mathbf{p}_i - \mathbf{c}\right),
\end{aligned}$$

and

$$\begin{aligned}
\nabla^2 f(\mathbf{w}_i) = \frac{1}{n}\mathbf{J}^\top\Bigg[&\text{diag}\left(\frac{1}{\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i)}\right)\text{diag}(\exp(\mathbf{J}\mathbf{w}_i)) \\
&- \text{diag}(\exp(\mathbf{J}\mathbf{w}_i))\text{diag}\left(\frac{1}{(\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i))^2}\right)\mathbf{K}\text{diag}(\exp(\mathbf{J}\mathbf{w}_i))\Bigg]\mathbf{J} \\
= \frac{1}{n}\mathbf{J}^\top\Bigg[&\text{diag}\left(\frac{\exp(\mathbf{J}\mathbf{w}_i)}{\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i)}\right) - \text{diag}\left(\frac{\exp(\mathbf{J}\mathbf{w}_i)}{(\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i))^2}\right)\mathbf{K}\,\text{diag}(\exp(\mathbf{J}\mathbf{w}_i))\Bigg]\mathbf{J} \\
= \frac{1}{n}&\mathbf{J}^\top\mathbf{H}_i\mathbf{J},
\end{aligned}$$

with

$$\mathbf{p}_i = \frac{\exp(\mathbf{J}\mathbf{w}_i)}{\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i)}, \quad \mathbf{H}_i = \operatorname{diag}(\mathbf{p}_i) - \operatorname{diag}\left(\frac{\exp(\mathbf{J}\mathbf{w}_i)}{(\mathbf{K}\exp(\mathbf{J}\mathbf{w}_i))^2}\right)\mathbf{K}\operatorname{diag}(\exp(\mathbf{J}\mathbf{w}_i)).$$

Bibliography

[1] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132: 428–446, 2020.

[2] Mosek ApS. MOSEK optimization toolbox for MATLAB. *User's Guide and Reference Manual, Version*, 4, 2019.

[3] Mosek ApS. MOSEK modeling cookbook, 2020.

[4] Amir Beck. *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*. SIAM, 2014.

[5] Stephen Becker and Jalal Fadili. A quasi-Newton proximal splitting method. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.

[6] Stephen Becker, Jalal Fadili, and Peter Ochs. On quasi-Newton forward-backward splitting: Proximal calculus and convergence. *SIAM Journal on Optimization*, 29(4):2445–2481, 2019.

[7] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[8] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020.

[9] Dimitri Bertsekas. *Convex optimization theory*, volume 1. Athena Scientific, 2009.

[10] Dimitri P Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2): 221–246, 1982.

[11] Åke Björck, Eric Grimme, and Paul Van Dooren. An implicit shift bidiagonalization algorithm for ill-posed systems. *BIT Numerical Mathematics*, 34(4): 510–534, 1994.

[12] Christian Boehm and Michael Ulbrich. A semismooth Newton-CG method for constrained parameter identification in seismic tomography. *SIAM Journal on Scientific Computing*, 37(5):S334–S364, 2015.

[13] Alfio Borzì and Volker Schulz. *Computational optimization of systems governed by partial differential equations*. SIAM, 2011.

[14] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[15] Peter N Brown. A local convergence theory for combined inexact-Newton/finite-difference projection methods. *SIAM Journal on Numerical Analysis*, 24(2): 407–434, 1987.

[16] R H Byrd, P Lu, J Nocedal, and C Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Control and Optimization*, 16(5): 1190–1208, September 1995.

[17] Daniela Calvetti, Gene Howard Golub, and Lothar Reichel. Estimation of the L-curve via Lanczos bidiagonalization. *BIT Numerical Mathematics*, 39(4): 603–619, 1999.

[18] Raymond H Chan, Chung-Wa Ho, and Mila Nikolova. Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization. *IEEE Transactions on image processing*, 14(10):1479–1485, 2005.

[19] Raymond H Chan, Kelvin K Kan, Mila Nikolova, and Robert J Plemmons. A two-stage method for spectral–spatial classification of hyperspectral images. *Journal of Mathematical Imaging and Vision*, 62(6):790–807, 2020.

[20] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

[21] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] Zhijun Chen and Hayden Schaeffer. Conditioning of random feature matrices: Double descent and generalization error. *arXiv preprint arXiv:2110.11477*, 2021.

[23] Margaret Cheney, David Isaacson, and Jonathan C Newell. Electrical impedance tomography. *SIAM review*, 41(1):85–101, 1999.

[24] Julianne Chung and Silvia Gazzola. Computational methods for large-scale inverse problems: A survey on hybrid projection methods. *arXiv preprint arXiv:2105.07221*, 2021.

[25] Julianne Chung and Katrina Palmer. A hybrid LSMR algorithm for large-scale Tikhonov regularization. *SIAM Journal on Scientific Computing*, 37(5):S562–S580, 2015.

[26] Julianne Chung and Arvind K Saibaba. Generalized hybrid iterative methods

for large-scale Bayesian inverse problems. *SIAM Journal on Scientific Computing*, 39(5):S24–S46, 2017.

[27] Julianne Chung, James G Nagy, and Dianne P O'leary. A weighted GCV method for Lanczos hybrid regularization. *Electronic Transactions on Numerical Analysis*, 28, 2008.

[28] Thomas F Coleman and Yuying Li. On the convergence of interior-reflective Newton methods for nonlinear minimization subject to bounds. *Mathematical programming*, 67(1):189–224, 1994.

[29] Thomas F Coleman and Yuying Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on optimization*, 6(2): 418–445, 1996.

[30] Andrew Cotter, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, Sashank J Reddi, and Yichen Zhou. Distilling double descent. *arXiv preprint arXiv:2102.06849*, 2021.

[31] Frank Curtis and Jorge Nocedal. Steplength selection in interior-point methods for quadratic programming. *Applied Mathematics Letters*, 20(5):516–523, 2007.

[32] Eric De Sturler, Serkan Gugercin, Misha E Kilmer, Saifon Chaturantabut, Christopher Beattie, and Meghan O'Connell. Nonlinear parametric inversion using interpolatory model reduction. *SIAM Journal on Scientific Computing*, 37(3):B495–B517, 2015.

[33] Alp Dener and Todd Munson. Accelerating limited-memory quasi-Newton convergence for large-scale optimization. In *International Conference on Computational Science*, pages 495–507. Springer, 2019.

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[35] Zeyu Deng, Abla Kammoun, and Christos Thrampoulidis. A model of double descent for high-dimensional binary linear classification. *Information and Inference: A Journal of the IMA*, 11(2):435–495, 2022.

[36] A Dey and H Frank Morrison. Resistivity modeling for arbitrarily shaped three-dimensional structures. *Geophysics*, 44(4):753–780, 1979.

[37] Nikita Doikov, Konstantin Mishchenko, and Yurii Nesterov. Super-universal regularized Newton method. *arXiv preprint arXiv:2208.05888*, 2022.

[38] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[39] Joseph C Dunn. Newton's method and the Goldstein step-length rule for constrained minimization problems. *SIAM Journal on Control and Optimization*, 18(6):659–674, 1980.

[40] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[41] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[42] Ioannis Epanomeritakis, Volkan Akçelik, Omar Ghattas, and Jacobo Bielak. A Newton-CG method for large-scale three-dimensional elastic full-waveform seismic inversion. *Inverse Problems*, 24(3):034015, 2008.

[43] Jin-yan Fan and Ya-xiang Yuan. On the quadratic convergence of the Levenberg-Marquardt method without nonsingularity assumption. *Computing*, 74:23–39, 2005.

[44] Michael C Ferris and Todd S Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2002.

[45] Shai Fine and Katya Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

[46] H P Flath, L C Wilcox, V Akçelik, J Hill, B van Bloemen Waanders, and O Ghattas. Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations. *SIAM Journal on Scientific Computing*, 33(1):407–432, January 2011.

[47] Eli M Gafni and Dimitri P Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22(6):936–964, 1984.

[48] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[49] Silvia Gazzola, Paolo Novati, and Maria Rosaria Russo. On Krylov projection methods and Tikhonov regularization. *Electron. Trans. Numer. Anal*, 44(1): 83–123, 2015.

[50] Silvia Gazzola, Per Christian Hansen, and James G Nagy. IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems. *Numerical Algorithms*, 81(3):773–811, 2019.

[51] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.

[52] Amin Ghodousian, Alireza Norouzi Azad, and Hadi Amiri. Log-sum-exp optimization problem subjected to Lukasiewicz fuzzy relational inequalities. *arXiv preprint arXiv:2206.09716*, 2022.

[53] Jean Charles Gilbert and Claude Lemaréchal. The module M1QN3. *INRIA Rep., version*, 3:21, 2006.

[54] Philip E Gill and Walter Murray. Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming*, 7:311–350, 1974.

[55] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

[56] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[57] Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[58] Gene H Golub, Alan Hoffman, and Gilbert W Stewart. A generalization of the Eckart-Young-Mirsky matrix approximation theorem. *Linear Algebra and its applications*, 88:317–327, 1987.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[60] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[61] Michael Grant, Stephen Boyd, and Yinyu Ye. CVX: MATLAB software for disciplined convex programming, 2008.

[62] John Greenstadt. On the relative efficiencies of gradient methods. *Mathematics of Computation*, 21(99):360–367, 1967.

[63] Stefanie Gunther, Lars Ruthotto, Jacob B Schroder, Eric C Cyr, and Nicolas R Gauger. Layer-parallel training of deep residual neural networks. *SIAM Journal on Mathematics of Data Science*, 2(1):1–23, 2020.

[64] Eldad Haber. *Computational methods in geophysical electromagnetics*. Mathematics in Industry (Philadelphia). Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015.

[65] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[66] Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. Learning across scales—multiscale methods for convolution neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[67] Eldad Haber, Keegan Lensink, Eran Treister, and Lars Ruthotto. IMEXnet a forward stable deep neural network. In *International Conference on Machine Learning*, pages 2525–2534. PMLR, 2019.

[68] William W Hager and Hongchao Zhang. A new active set algorithm for box constrained optimization. *SIAM Journal on Optimization*, 17(2):526–557, 2006.

[69] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, 1998.

[70] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, 1998.

[71] Per Christian Hansen. *Discrete inverse problems*, volume 7 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2010.

[72] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022.

[73] James Lincoln Herring, James Nagy, and Lars Ruthotto. Gauss–Newton optimization for phase recovery from the bispectrum. *IEEE Transactions on Computational Imaging*, 2019.

[74] Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

[75] Jiang Hu, Andre Milzarek, Zaiwen Wen, and Yaxiang Yuan. Adaptive quadratically regularized Newton method for Riemannian optimization. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1181–1207, 2018.

[76] Yunyi Hu. *Matrix Computations and Optimization for Spectral Computed Tomography*. PhD thesis, Emory University, 2019.

[77] Yunyi Hu, Martin S Andersen, and James G Nagy. Spectral computed tomography with linearization and preconditioning. *SIAM Journal on Scientific Computing*, 41(5):S370–S389, 2019.

[78] Yunyi Hu, James G Nagy, Jianjun Zhang, and Martin S Andersen. Nonlinear

optimization for mixed attenuation polyenergetic image reconstruction. *Inverse Problems*, 35(6):064004, 2019.

[79] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[80] Tao Huang, Weisheng Dong, Xuemei Xie, Guangming Shi, and Xiang Bai. Mixed noise removal via laplacian scale mixture modeling and nonlocal low-rank approximation. *IEEE Transactions on Image Processing*, 26(7):3171–3186, 2017.

[81] Hui Ji, Sibin Huang, Zuowei Shen, and Yuhong Xu. Robust video restoration by joint sparse and low rank matrix approximation. *SIAM Journal on Imaging Sciences*, 4(4):1122–1142, 2011.

[82] Kelvin Kan, James G Nagy, and Lars Ruthotto. Avoiding the double descent phenomenon of random feature models using hybrid regularization. *arXiv preprint arXiv:2012.06667*, 2020.

[83] Kelvin Kan, Samy Wu Fung, and Lars Ruthotto. PNKH-B: A projected Newton–Krylov method for large-scale bound-constrained optimization. *SIAM Journal on Scientific Computing*, 43(5):S704–S726, 2021.

[84] C T Kelley. *Iterative Methods for Optimization.* SIAM, Philadelphia, January 1999.

[85] D Kim, S Sra, and I S Dhillon. Tackling box-constrained optimization via a new projected quasi-Newton approach. *SIAM Journal on Control and Optimization*, 32(6):3548–3563, January 2010.

[86] Donghwan Kim and Jeffrey A Fessler. Adaptive restart of the optimized gra-

dient method for convex optimization. *Journal of Optimization Theory and Applications*, 178(1):240–263, 2018.

[87] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. A guide to sample average approximation. *Handbook of simulation optimization*, pages 207–243, 2015.

[88] Ganesh Ramachandra Kini and Christos Thrampoulidis. Analytic study of double descent in binary classification: The impact of loss. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2527–2532. IEEE, 2020.

[89] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[90] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[91] Weiwei Kong, Walid Krichene, Nicolas Mayoraz, Steffen Rendle, and Li Zhang. Rankmax: An adaptive projection alternative to the softmax function. *Advances in Neural Information Processing Systems*, 33:633–643, 2020.

[92] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90, 2017.

[94] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, (45):255–282, 1950.

[95] Germana Landi and E Loli Piccolomini. A projected Newton-CG method for nonnegative astronomical image deblurring. *Numerical Algorithms*, 48(4):279–300, 2008.

[96] Germana Landi and Elena Loli Piccolomini. An improved Newton projection method for nonnegative deblurring of Poisson-corrupted images with Tikhonov regularization. *Numerical Algorithms*, 60(1):169–188, 2012.

[97] Germana Landi and Elena Loli Piccolomini. NPTool: a MATLAB software for nonnegative image restoration with Newton projection methods. *Numerical Algorithms*, 62(3):487–504, 2013.

[98] Y LeCun, B E Boser, and J S Denker. Handwritten digit recognition with a back-propagation network. In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 396–404, 1990.

[99] Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[100] Jason D Lee, Yuekai Sun, and Michael A Saunders. Proximal Newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3):1420–1443, 2014.

[101] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[102] Dong-Hui Li, Masao Fukushima, Liqun Qi, and Nobuo Yamashita. Regularized Newton methods for convex minimization problems with singular solutions. *Computational optimization and applications*, 28:131–147, 2004.

[103] Fanghui Liu, Zhenyu Liao, and Johan Suykens. Kernel regression in high dimensions: Refined analysis beyond double descent. In *International Conference on Artificial Intelligence and Statistics*, pages 649–657. PMLR, 2021.

[104] Chao Ma, Lei Wu, and E Weinan. The slow deterioration of the generalization error of the random feature model. In *Mathematical and Scientific Machine Learning*, pages 373–389. PMLR, 2020.

[105] Andreas Mang and George Biros. An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration. *SIAM journal on imaging sciences*, 8(2):1030–1069, 2015.

[106] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11 (2):431–441, 1963.

[107] Peter Robert McGillivray. *Forward modeling and inversion of DC resistivity and MMR data*. PhD thesis, University of British Columbia, 1992.

[108] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

[109] Ekaterina Merkurjev, Tijana Kostic, and Andrea L Bertozzi. An MBO scheme on graphs for classification and image processing. *SIAM Journal on Imaging Sciences*, 6(4):1903–1930, 2013.

[110] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59, 1960.

[111] Konstantin Mishchenko. Regularized Newton method with global $\mathcal{O}(1/k^2)$ convergence. *arXiv preprint arXiv:2112.02089*, 2021.

[112] Bamdev Mishra, Gilles Meyer, Francis Bach, and Rodolphe Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23 (4):2124–2149, 2013.

[113] José Luis Morales and Jorge Nocedal. Remark on "algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–4, 2011.

[114] Jorge J Moré and Danny C Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, 16:1–20, 1979.

[115] Jorge J Moré and Gerardo Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1 (1):93–113, 1991.

[116] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021 (12):124003, 2021.

[117] Stephen G Nash. Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788, 1984.

[118] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro.

Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[119] Elizabeth Newman, Lars Ruthotto, Joseph Hart, and Bart van Bloemen Waanders. Train like a (var) pro: efficient training of neural networks with variable projection. *SIAM Journal on Mathematics of Data Science*, 3(4):1041–1066, 2021.

[120] Elizabeth Newman, Julianne Chung, Matthias Chung, and Lars Ruthotto. slimTrain—a stochastic approximation method for training separable deep neural networks. *SIAM Journal on Scientific Computing*, 44(4):A2322–A2348, 2022.

[121] Frank Nielsen and Ke Sun. Guaranteed bounds on the kullback–leibler divergence of univariate mixtures. *IEEE Signal Processing Letters*, 23(11):1543–1546, 2016.

[122] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[123] Levon Nurbekyan, Wanzhou Lei, and Yunan Yang. Efficient natural gradient descent methods for large-scale optimization problems. *arXiv preprint arXiv:2202.06236*, 2022.

[124] Brendan O'Donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[125] Christopher C Paige and Michael A Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[126] Christopher C Paige and Michael A Saunders. Algorithm 583: LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software (TOMS)*, 8(2):195–209, 1982.

[127] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[128] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[129] Roman A Polyak. Regularized Newton method for unconstrained convex optimization. *Mathematical programming*, 120:125–145, 2009.

[130] Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.

[131] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[132] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[133] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2019.

[134] Lars Ruthotto, Eran Treister, and Eldad Haber. jinv–a flexible julia package for PDE parameter estimation. *SIAM Journal on Scientific Computing*, 39(5): S702–S722, 2017.

[135] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[136] Mark Schmidt, Ewout Berg, Michael Friedlander, and Kevin Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In *Artificial Intelligence and Statistics*, pages 456–463, 2009.

[137] Mark Schmidt, Dongmin Kim, and Suvrit Sra. Projected Newton-type methods in machine learning. *Optimization for Machine Learning*, (1), 2012.

[138] Hui Shi, Yann Traonmilin, and Jean-François Aujol. Compressive learning for patch-based image denoising. *SIAM Journal on Imaging Sciences*, 15(3):1184–1212, 2022.

[139] Jos F Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[140] Chung-Li Tseng, Yiduo Zhan, Qipeng P Zheng, and Manish Kumar. A MILP formulation for generalized geometric programming using piecewise-linear approximations. *European Journal of Operational Research*, 245(2):360–370, 2015.

[141] Reha H Tütüncü, Kim-Chuan Toh, and Michael J Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical programming*, 95(2): 189–217, 2003.

[142] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

[143] Alexander Vidal, Samy Wu Fung, Luis Tenorio, Stanley Osher, and Levon Nurbekyan. Taming hyperparameter tuning in continuous normalizing flows using the JKO scheme. *arXiv preprint arXiv:2211.16757*, 2022.

[144] Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002.

[145] Chao Wang, Grey Ballard, Robert Plemmons, and Sudhakar Prasad. Joint 3D localization and classification of space debris using a multispectral rotating point spread function. *Applied Optics*, 58(31):8598–8611, 2019.

[146] Chao Wang, Raymond Chan, Mila Nikolova, Robert Plemmons, and Sudhakar Prasad. Nonconvex optimization for 3-dimensional point source localization using a rotating point spread function. *SIAM Journal on Imaging Sciences*, 12 (1):259–286, 2019.

[147] Zaiwen Wen, Andre Milzarek, Michael Ulbrich, and Hongchao Zhang. Adaptive regularized self-consistent field iteration with exact Hessian for electronic structure calculation. *SIAM Journal on Scientific Computing*, 35(3):A1299–A1324, 2013.

[148] Xinming Wu, Zaiwen Wen, and Weizhu Bao. A regularized Newton method for computing ground states of Bose–Einstein condensates. *Journal of Scientific Computing*, 73:303–329, 2017.

[149] Samy Wu Fung. *Large-Scale Parameter Estimation in Geophysics and Machine Learning*. PhD thesis, Emory University, 2019.

[150] Samy Wu Fung and Zichao Wendy Di. Multigrid optimization for large-scale ptychographic phase retrieval. *SIAM Journal on Imaging Sciences*, 13(1):214–233, 2020.

[151] Samy Wu Fung and Lars Ruthotto. A multiscale method for model order reduction in PDE parameter estimation. *Journal of Computational and Applied Mathematics*, 350:19–34, 2019.

[152] Samy Wu Fung and Lars Ruthotto. An uncertainty-weighted asynchronous ADMM method for parallel PDE parameter estimation. *SIAM Journal on Scientific Computing*, 41(5):S129–S148, 2019.

[153] Samy Wu Fung, Sanna Tyrväinen, Lars Ruthotto, and Eldad Haber. ADMM-softmax: An ADMM approach for multinomial logistic regression. *Electronic Transactions on Numerical Analysis*, 52:214–229, 2020.

[154] Xiangming Xi, Jun Xu, and Yunjiang Lou. Log-sum-exp optimization based on continuous piecewise linearization techniques. In *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, pages 600–605. IEEE, 2020.

[155] Eugene K Yang and Jon W Tolle. A class of methods for solving large, convex quadratic programs subject to box constraints. *Mathematical Programming*, 51 (1-3):223–228, 1991.

[156] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[157] Yiduo Zhan, Qipeng P Zheng, Chung-Li Tseng, and Eduardo L Pasiliao. An accelerated extended cutting plane approach with piecewise linear approximations for signomial geometric programming. *Journal of Global Optimization*, 70 (3):579–599, 2018.

[158] Richard Y Zhang and Javad Lavaei. Modified interior-point method for large-and-sparse low-rank semidefinite programs. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5640–5647. IEEE, 2017.

[159] Yong-Qiang Zhao and Jingxiang Yang. Hyperspectral image denoising via sparse representation and low-rank constraint. *IEEE Transactions on Geoscience and Remote Sensing*, 53(1):296–308, 2014.