The Dissertation Committee for Vatsal Nilesh Shah
certifies that this is the approved version of the following dissertation:

On Variants of Stochastic Gradient Descent

Committee:

Sujay Sanghavi, Supervisor

Constantine Caramanis

Sanjay Shakkottai

Aryan Mokhtari

Anastasios Kyrillidis

On Variants of Stochastic Gradient Descent

by

Vatsal Nilesh Shah

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2020

Dedicated to Mummy, Papa and Dada

In loving memory of Dadi

Acknowledgments

As I embarked upon the journey to pursue PhD from the familiar surroundings of Mumbai to a new country and a new city with almost no acquaintances, I was quite skeptical about whether it was the right choice. *The future is scary, but you can't just run back to the past because it's familiar*[1]. While this quote provided me with a temporary respite during the unfamiliar, uncomfortable start at UT, I have come to not only enjoy my time here but also grown as a person and a researcher. None of this would have been possible without the never-ending support of my family, friends, collaborators, and colleagues.

Prof. Sujay Sanghavi has been an amazing advisor who has provided me with the right mix of guidance and independence during the course of my PhD. Sujay's unique analytical approach to problem-solving, ability to always ask the right questions, focusing on the bigger picture while ensuring rigor in theoretical guarantees are some of the many many things that have immensely helped me during the course of my PhD. My PhD journey has had its fair share of ups and downs and I would like to especially thank Sujay for always making a point to talk about both my academic as well as personal well-being.

Without the constant guidance, support, and encouragement provided

[1]From How I Met Your Mother, Season 6 Episode 24: *Challenge Accepted*

by Prof. Anastasios Kyrillidis (Tasos), my thesis would have been non-existent. I first met Tasos when he was a post-doc at UT and I feel incredibly lucky that Sujay paired us to work together. He has been an amazing mentor, friend, and collaborator ever since. I am indebted to him for his immense influence on my writing style, critical thinking, time management, multi-tasking and so much more.

I have taken two classes under Prof. Sanjay Shakkottai and the enthusiasm he shows in both teaching and research inevitably rubs off onto you. Sanjay's easy approachablility and expertise in a wide variety of fields implies that I have had numerous discussions about different research topics in the corridors of EER and UTA. Prof. Constantine Caramanis' structured approach to both research and teaching means that there are numerous insights to be gained in every discussion I have had with him. I also enjoyed interacting with Constantine outside the academic environment especially at WNCG socials. Prof. Aryan Mokhtari has been an exciting addition to WNCG and his expertise in optimization has allowed me to discuss various research problems with him which has contributed invaluably to my research.

I would like to thank all my collaborators for their invaluable contributions to my thesis. During the course of our collaborations, I have really admired the numerous insights I have gained via technical discussions with Xiaoxia (Shirley) Wu, Soumya Basu, Erik Lindgren, Yanyao Shen, Ashish Katiyar, John Chen, and Megasthenis Asteris. Shirley, in particular, has been an inspiration to collaborate with due to her work ethic and ability to find a way to prove even the most challenging problems I have run into. I have also learned tremendously while collaborating with Prof. Alex Dimakis, Prof. Adam Klivans, and Prof. Rachel Ward. The numerous discussions I had with students from Sujay's lab group: Shanshan, Srinadh, Dohyung, Yanyao, Ashish, Shuo, Anish, and Rudrajit have been quite instrumental towards the completion of my thesis.

My three internships at Technicolor Research, Amazon, and Amazon Web Services (AWS) have broadened my horizons and provided a diverse glimpse of life in industry. I would like to thank my mentors at Technicolor, Nikhil Rao, and Weicong Ding for the countless discussions and invaluable guidance towards my first project in recommender systems. At Amazon, I had the opportunity to work with a project in production for the Homepage Personalization team. Ali Jalali's mentorship was pivotal in helping me navigate a steep learning curve that came with handling extremely large datasets. At AWS, I had the opportunity to work under Anima Anandkumar, where I had the learned about deep learning and computer vision, which came in handy in the latter part of my thesis. During the course of these internships, I had the amazing opportunity to have technical and non-technical discussions with Farhad, Julian, Tan, Rushil, Teja, Shijia, Shaozhe, Kamyar, Michael, Jeremy, Ben, Fanny, Thom, Yifan and many others.

I would also like to thank Karen, Melanie, Melody, Barry, Jaymie, and Apipol for helping me resolve all the academic formalities with minimal hassles. The efforts by Karen, Jaymie, and Apipol, in particular, have played

an important part in fostering a sense of togetherness in WNCG by organizing socials, potlucks, happy hours, and numerous other events.

The last six years at UT have been incredible thanks to some amazing people I met in WNCG. Talking with Avradeep is like reading an encyclopedia; you will always get something new out of the conversation despite the utility. Murat, thanks for being an amazing friend and more importantly introducing me to the world of coffee shops, I don't think I would have finished my PhD without them. Shalmali has always played the role of a caring *albeit nosy* guardian in WNCG and has been a source of inspiration throughout my time here. I am thankful to Eirini for providing an incredible support system both within and outside the walls of WNCG during some of my most difficult moments. Soumya is always up for spontaneous plans from going to coffee shops to live music events to taking last-minute road trips. My conversations with him, about life, research, and everything else, have been fun, insightful, and *kept me grounded*.

In the last three years, Ashish has helped me navigate all the ups and downs in my life with utmost patience. I have had so many profound conversations with him during our road trips, coffee shops, food trucks, Tous Les Jours detours, and everything else. Mónica has introduced me to climbing, exotic food trucks, Colombian delicacies, and often provided the best advice to all my problems. Oddwood Ales was definitely a start of an exciting series of nights with Ashish, Manan, Mónica and Soumya and I am hoping that the tradition keeps on continuing albeit with more positive updates. Watching

Champions League matches with Ajil and Manan, especially the legendary Liverpool comeback against Barcelona, is something I will never forget. I enjoyed hanging out with Mridula, Rajat, Ajil, Natasa, Erik, Sara, Derya, Abhishek, Alan, Dave, Megha, amongst others in EER, at coffee shops, pubs, dinners and over wine and trivia nights. I feel lucky to have met Divya, Suriya, Ankit, Ethan, Karthikeyan, Avhishek who have all provided me with invaluable guidance during my first few years at UT.

I would also like to thank my therapist Anita Stoll who has played a significant role in helping me deal with trauma and the subsequent anxiety during the course of my PhD. I have seen that graduate school can be a pretty isolated environment especially for international students. In such cases, asking for help is never easy and it took me a long time to recognize that. There was also the initial skepticism about whether therapy is for me or the usefulness of it, however I can confidently say that the benefits have heavily outweighed the initial struggle. Since I started regular sessions with Anita, I have been better equipped to deal with life and PhD.

In this terrifying world, all we have are the connections that we make[2]. And I feel incredibly lucky to have met so many wonderful people during the last six years. Pranav has been an amazing friend and I don't think anything I say here comes remotely close to doing justice to how much he has helped me navigate the ups and downs of PhD life. It all started with random walks

[2]From Bojack Horseman: Season 3 Episode 4: *Fish Out of Water*

and petty squabbles with Priyanka *the chief-event planner*, and with time our friendship has continued to blossom over long talks, dinners, board game nights, road trips and eventful bike rides. I hate Nitin for all the board games he has won and all the keys he has '*lost*', but admire his kindhearted nature along with his discipline and dedication in literally all his other endeavors. I have admired Esha right from *our first meeting** for her ambition, opinions, providing the best sounding board for all my problems as well as bringing much needed drama to my mundane life. Quarantining in 2020 would have been way harder without Esha and Pranav's *improvised* theatrics. I always enjoyed having ambivalent conversations with Vikas.

Amongst my roommates, I have learned a lot from Venky (now Prof. Venktesh) who also gets the title of *nicest person to exist.* I have grown tremendously as a person just by observing how he conducts himself in difficult situations. I will always cherish the fun conversations, cooking escapades and Netflix binging sessions with Bharath and I am grateful to him for taking care of me through the toughest phase of my life. Long conversations and countless road trips perfectly encapsulate my *happening* friendship with Devesh. Recreating the road trip through Spain in Zindagi Na Milegi Dobara with Devesh and Venky remains a highlight of grad school.

I had a lot of fun with Surbhi during our late-night work sessions at coffee shops, exploring tea places, dinners and board games primarily due to her relentless energy and enthusiasm. In the last couple of years, I have enjoyed all the spontaneous and non-spontaneous plans with Akanksha from long talks,

working at coffee shops, to dinners and board games. I had a lot of fun talking about life, Mumbai and life in Mumbai amongst many other things with Prerana. It was quite memorable hanging out with Aastha, Kartik, Madhumitha, Manan, Nihal, Nithin over dinners, badminton, trivia nights, etc. I will also cherish my countless interactions with Ankith, Anudhyan, Arjun, Arun, Bhavin, Brinda, Eddy, Luca, Parshu, Pooja, Preeti, Priya, Roshan, Shounak, Siddharth, and numerous other people who I have missed out on.

Akshay has been one of my closest confidantes for over 11 years as he has helped me every step of the way during my PhD. In fact, without him, I may not have begun this wonderful journey at UT. Long conversations with Yatish and Sneha along with impromptu dinner plans, trips, and travelling were a particular highlight of my Fall semester in the Bay area. Deeksha has not only been a great friend but also an inspiration in how she continues to have a social impact while pursuing her PhD. I have come to realize that I am not that great at keeping in touch with *long-distance* friends and I would like to thank Aakash, Adarsh, Ankush, Gurbaksh, Hiteshi, Jay, Jainik, Siddharth, Tejas(es), Udit, Vidhya, Vivek and Yashasvi who have kept in touch with me over the years.

I would like to thank Prof. Karandikar, who was my undergraduate advisor, for igniting my interest in the field of optimization. Prof. Borkar played an instrumental role in convincing me to pursue PhD at UT Austin and is always a delight to talk with regarding both academic and non-academic endeavors. I still cherish the publication of my first few papers with Prof. B.

K. Mohan and Surendar Verma which ultimately provided an early motivation in pursuing graduate school.

I feel extremely lucky to be blessed with the unconditional, unwavering love, support and encouragement by my parents. Without their countless sacrifices, hardships and struggles, none of this would have been possible. They have supported me every step of the way on this journey and always allowed me to express myself and I am really thankful for that. One of the last vivid memory I have of my late grandmother Indira dadi is that when I was having second thoughts about pursuing grad school, she was the one who provided the final push that finally convinced me and there was no looking back after that. I know that she would have been proud of what I have accomplished and the person I have become in the last 6 years. Much to the annoyance of my parents, my grandfather has always been my biggest defender and I always enjoy goofing around with him at home, on video calls and during trips. I would also like to thank Prakash fua, Sangeeta fai, Vrajesh mama, Pinki mami, Paresh kaka, Smita kaki and Manju ba who have constantly supported me throughout this journey. Special shoutout to my cousins Aarya, Ashit jiju, Binita didi, Ishita, Utsav, Urvi (aka Urvashi), and for ensuring that my India trips are always filled with memorable stories and adventures.

No matter how you get there or where you end up, human beings have this miraculous gift to make that place home[3]. When I first came to Austin, the weather was too hot and the food portions too big and everything miles

[3]From The Office (U.S.), Season 9 Episode 25: *Finale*

apart. However, as time passed by, I have come around to enjoy the same things which annoyed me at first. Austin has just been like a home away from home and I will truly miss this stupid, wonderful, crazy, exciting city.

On Variants of Stochastic Gradient Descent

Vatsal Nilesh Shah, Ph.D.
The University of Texas at Austin, 2020

Supervisor: Sujay Sanghavi

Stochastic Gradient Descent (SGD) has played a crucial role in the success of modern machine learning methods. The popularity of SGD arises due to its ease of implementation, low memory and computational requirements, and applicability to a wide variety of optimization problems. However, SGD suffers from numerous issues; chief amongst them are high variance, slow rate of convergence, poor generalization, non-robustness to outliers, and poor performance for imbalanced classification. In this thesis, we propose variants of stochastic gradient descent, to tackle one or more of these issues for different problem settings.

In the first chapter, we analyze the trade-off between variance and complexity to improve the convergence rate of SGD. A common alternative in the literature to SGD is Stochastic Variance Reduced Gradient (SVRG), which achieves linear convergence. However, SVRG involves the computation of a full gradient every few epochs, which is often intractable. We propose the Cheap Stochastic Variance Reduced Gradient (CheapSVRG) algorithm that attains linear convergence up to a neighborhood around the optimum without requiring a full gradient computation step.

In the second chapter, we aim to compare the generalization capabilities of adaptive and non-adaptive methods for over-parameterized linear regression. Of the many possible solutions, SGD tends to gravitate towards the solution with minimum $\ell 2$-norm while adaptive methods do not. We provide specific conditions on the pre-conditioner matrices under which a subclass of adaptive methods has the same generalization guarantees as SGD for over-parameterized linear regression. With synthetic examples and real data, we show that minimum norm solutions are not an excellent certificate to guarantee better generalization.

In the third chapter, we propose a simple variant of SGD that guarantees robustness. Instead of considering SGD with one sample, we take a mini-batch and choose the sample with the lowest loss. For the noiseless framework with and without outliers, we provide conditions for the convergence of MKL-SGD to a provably better solution than SGD in the worst case. We also perform the standard rate of convergence analysis for both noiseless and noisy settings.

In the final chapter, we tackle the challenges introduced by imbalanced class distribution in SGD. In place of using all the samples to update the parameter, our proposed Balancing SGD (B-SGD) algorithm rejects samples with low loss as they are redundant and do not play a role in determining the separating hyperplane. Imposing this label-dependent loss-based thresholding

scheme on incoming samples allows us to improve the rate of convergence and achieve better generalization.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The growing popularity of social networks, streaming services, and e-commerce websites has led to the development of ranking, recommendation, and personalization algorithms for customer retention. These algorithms often rely on the availability of large datasets for their successful implementation. Similarly, the availability of big data, such as the publicly available ImageNet dataset [8] was one of the primary reasons behind the deep learning revolution. These modern datasets often involve hundreds of thousands of examples with thousand of features. The ImageNet dataset consists of more than a million images for more than 20000 categories; the Netflix prize dataset consisted of a training data set with $100,480,507$ ratings that $480,189$ users gave to $17,770$ movies.

Classical optimization techniques rely on full gradient computations to perform the parameter update step. These techniques may often be intractable or impractical with increasing size of datasets due to their large memory and computation requirements. Computing the gradient over all samples would involve computing and storing the gradients of each of these million data samples either in conjunction or tracking their sum by computing the gradient per data sample sequentially. The former would require a lot of memory, while the latter will be slow and inefficient.

Stochastic gradient descent (SGD) arose as a logical alternative to full gradient descent based optimization algorithms for large datasets. The idea behind SGD is straightforward: in each epoch, randomly draw a sample from the available training data, compute the gradient of that chosen sample, and use that gradient to update the unknown underlying parameter. Nowadays, SGD is one of the most widely used stochastic optimization techniques to solve convex and non-convex optimization problems in machine learning. The popularity of SGD arises from its ease of implementation as well as low computational and memory requirements. The universal applicability to a large class of problems ranging from linear regression, support vector machines, reinforcement learning to deep learning is another reason behind its widespread utility. However, SGD suffers from many issues spanning from the high variance of the iterates, poor generalization, slow convergence, and non-robustness to outliers.

1.1 Contributions and Organization

In this dissertation, we address the problems of high variance, convergence, generalization, and robustness for SGD under different optimization frameworks. In the introduction, we first discuss the issues with existing approaches in dealing with the problem in hand. We then suggest fast, practical variants of stochastic gradient descent while providing theoretical guarantees for convergence and generalization to alleviate the highlighted issues. The second, fourth, and fifth chapters focus on the more standard stochastic optimization setup; the emphasis in the third chapter is on understanding the behavior of stochastic and adaptive methods specifically for over-parameterized problems.

1.2 Chapter 2: Trading-off Variance and Complexity in Stochastic Gradient Descent

In the first chapter, we analyze the trade-off between variance and complexity in stochastic gradient descent based methods. Running SGD on a typical dataset, we observe that the gradients of a randomly chosen sample (or mini-batch of random subset of samples) behave as perturbed estimates drawn from a normal distribution centered around the full gradient. The variance term necessitates the use of decreasing step-size for SGD, leading to sub-linear convergence guarantees. In fact, higher the variance of the gradient of these samples, slower is the rate of convergence of SGD. Consequently, under the assumptions of strong convexity gradient descent enjoys linear convergence but requires high computational complexity. On the others hand, SGD requires low computational complexity but achieves slower convergence.

The popular Stochastic Variance-Reduced Gradient (SVRG) [3] method mitigates this shortcoming, adding a new update rule which requires infrequent passes over the entire input dataset to compute the full-gradient. SVRG consists of an outer loop and an inner loop. The outer loop requires a full gradient computation step, which makes the algorithm intractable for large datasets. In the inner loop, there is $O(b)$ gradient computations per epoch, where b is the size of the mini-batch. SVRG allows us to achieve both linear convergence and low computational complexity per epoch on an average. However, SVRG involves the computation of a full gradient step in the outer loop, which makes it intractable for large datasets. The high computational complexity of SVRG is thus at odds against the primary motivation of using stochastic methods for large datasets.

In this chapter, we propose the CheapSVRG algorithm that guarantees linear convergence and has lower computational complexity requirements than SVRG. The CheapSVRG algorithm consists of a simple tweak where we replace the full gradient step in the outer loop with a large mini-batch. We observe that CheapSVRG is tractable for large datasets and demonstrates linear convergence. The linear convergence shown by the CheapSVRG algorithm is up to a neighborhood around the optimum. The radius of the neighborhood depends on the size of the mini-batch in the outer loop. In this chapter, we analyze the delicate trade-off between variance and the computational complexity of the CheapSVRG algorithm. Lastly, we also back up our theoretical guarantees with experiments on synthetic as well as real datasets.

1.3 Chapter 3: On the Generalization of Adaptive Methods for Over-parameterized Linear Regression

Over-parameterized linear regression possesses infinite global minima. However, each of these minima generalizes differently. SGD has a propensity to seek the solution with the minimum $\ell 2$ norm amongst all these infinite solutions. However, we show empirically and theoretically, that minimum $\ell 2$ norm is not a good certificate to guarantee better generalization for over-parameterized linear regression.

Adaptive methods, defined as any stochastic gradient descent method multiplied by a (non-identity) pre-conditioner matrix, played a very critical role in the success of deep learning. Traditionally, adaptive methods allow us to eliminate the hyper-parameter tuning step and guarantee faster convergence. However, it is not yet clear why adaptive methods generalize well. We show that in addition to faster convergence, adaptive methods can potentially provide better generalization performance than SGD depending on the problem at hand.

In this chapter, we provide conditions on the pre-conditioner matrices needed to ensure the convergence of adaptive methods to a stationary point. Also, we propose an explicit criterion on the pre-conditioner matrix that can determine if the adaptive methods will converge to the minimum $\ell - 2$ norm solution. We also prove and showcase the existence of a strange descent phenomenon for adaptive methods in over-parameterized linear regression.

In the experimental section, we provide examples using both over-parameterized linear regression with both continuous and discrete labels where adaptive methods have better generalization than SGD. We also ran extensive experiments using MNIST, CIFAR-10, and CIFAR-100 datasets on different architectures to demonstrate the potential of adaptive approaches to guarantee better generalization in over-parameterized frameworks.

1.4 Chapter 4: Choosing the Sample with Lowest Loss makes SGD Robust

Annotating large datasets with correct labels is a time-consuming and computationally expensive process. Despite the best precautions, these datasets are often riddled with a few mislabeled samples or outliers caused due to human or instrumentation errors. In this chapter, we focus on corruption via errors in labels. We assume that the data samples are left untouched and uncorrupted. We also differentiate between the noiseless, noisy, and outlier data-samples based on the distance of the true optimum of clean samples from the optimal set of the given data sample.

With only clean samples, SGD converges to the unique optimum that minimizes the average loss of the clean samples. In the presence of outliers, SGD converges to a solution that may be arbitrarily far from the desired optimum of clean samples. This occurs as SGD tends to treat all samples equally irrespective of whether they belong to the set of uncorrupted or corrupted samples.

In this chapter, we propose Min-k Loss SGD (MKL-SGD) Algorithm, where we sample a batch of k samples, evaluate the losses of each of these k samples. We then pick the sample with the smallest loss to perform the gradient update step. We observe that this simple tweak to the classic SGD algorithm makes SGD more robust.

However, the expected gradient using MKL-SGD is no longer biased. The biasedness introduces complications in providing theoretical guarantees for generalization and convergence for MKL-SGD. To avoid this issue, we construct a surrogate objective function that is piece-wise, continuous, and non-convex such that the expected MKL-SGD gradient is unbiased.

For the noiseless setting without outliers, we show using Restricted Secant Inequality that MKL-SGD converges to the unique optimum of the clean samples. We show that the surrogate loss landscape has many local minima for the noiseless settings in the presence of outliers. However, we show that if it is possible to avoid bad local minima when the functions satisfy certain conditions depending on the condition number of the data matrix and fraction of outliers. Moreover, we show that any solution attained by MKL-SGD will be closer to the desired optimum of the clean samples than the unique SGD solution irrespective of the initialization. To the best of our knowledge, this is the first research that proposes worst-case guarantees in the stochastic (SGD) setup.

Next, we show the in-expectation rate of convergence bounds for all four frameworks described in the paper: noiseless without outliers, noiseless with outliers, noisy without outliers, and noisy with outliers. We prove that

MKL-SGD has linear convergence around a neighborhood of the optimum similar to SGD, albeit with slightly worse constants. Lastly, we bolster the theoretical guarantees by demonstrating the superior performance of MKL-SGD on synthetic linear regression as well as small scale neural networks.

1.5 Chapter 5: Balancing SGD: Faster Optimization for Imbalanced Classification

Imbalanced datasets are quite common, especially in the fields of medicine, finance, engineering, etc. While the real world is often characterized by symmetry, the process of data-collection can introduce asymmetries in the training data distribution. Procuring balanced datasets requires significant efforts in terms of cost and time. For example, the number of surviving marmosets and pandas is identical in number. However, it is easier to obtain 1000 high-quality images of pandas than marmosets. Taking additional pictures of marmoset might be an expensive and time-consuming process.

In the presence of separable data with skewed empirical distribution and balanced test distribution, classical optimization methods, including SGD, suffer from extremely slow convergence and poor generalization. Most of the popular state-of-the-art algorithms designed to address the problem of imbalance rely on balancing the distributions using resampling, reweighting, cost-based classification, or ensemble strategies. These methods need access to either data distribution [9, 10, 11] or label distribution [12] or both. These approaches suffer from many issues, such as catering to specific applications, expensive pre-computations, access to label/data distribution, and slow convergence in the stochastic setting.

In this chapter, we propose a simple, memory-efficient, computationally inexpensive variant of SGD called Balancing SGD (B-SGD). B-SGD is an ensemble approach that combines undersampling with a label-based loss thresholding scheme. We provide a strong theoretical basis for designing the B-SGD algorithm. Additionally, we guarantee an upper bound on the number of gradient computation steps required by B-SGD, as well as a sound analysis for loss threshold selection. Experiments on synthetic as well as real datasets indicate that B-SGD outperforms traditional label-unaware methods in terms of both gradient computations and generalization performance.

Chapter 2

Trading-off Variance and Complexity in Stochastic Gradient Descent

Stochastic gradient descent is the method of choice for large-scale machine learning problems, by virtue of its low complexity per iteration. However, it lags behind its non-stochastic counterparts with respect to the convergence rate, due to the high variance introduced by the stochastic updates. The popular Stochastic Variance-Reduced Gradient (SVRG) method mitigates this shortcoming, adding a new update rule which requires infrequent passes over the entire input dataset to compute the full-gradient. Other popular methods proposed to resolve the issue of high variance and slow convergence include importance sampling-based gradient updates [14], Stochastic Average Gradient (SAG) [15], SAGA [16], Stochastic Dual Coordinate Ascent (SDCA) [17], etc. However, these methods have either high memory or computational complexity requirements or both. As highlighted previously, high complexity and storage demands go against the primary motivation of using stochastic updates for large datasets.

Parts of this chapter are available at [13]. The author was a part of formulating the problem, designing and analyzing the algorithms, writing up the results, and performed the simulations presented in the paper.

In this work, we propose CheapSVRG, a stochastic variance-reduction optimization scheme. Our algorithm is similar to SVRG, but instead of the full gradient, it uses a surrogate, which can be efficiently computed on a small subset of the input data. It achieves a linear convergence rate --up to some error level, depending on the nature of the optimization problem--and features a trade-off between the computational complexity and the convergence rate. Empirical evaluation shows that CheapSVRG performs at least competitively compared to state of the art.

2.1 Introduction

Several machine learning and optimization problems involve the minimization of a smooth, convex and *separable* cost function $F : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} F(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w}), \tag{2.1}$$

where the d-dimensional variable $\boldsymbol{w}$ represents model parameters, and each of the functions $f_i(\cdot)$ depends on a single data point. Linear regression is such an example: given points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ in $\mathbb{R}^{p+1}$, one seeks $\mathbf{w} \in \mathbb{R}^d$ that minimizes the sum of $f_i(\boldsymbol{w}) = (y_i - \boldsymbol{w}^\top \mathbf{x}_i)^2$, $i = 1, \dots, n$. Training of neural networks [18, 3], multi-class logistic regression [3, 19], image classification [20], matrix factorization [21] and many more tasks in machine learning entail an optimization of similar form.

Batch gradient descent schemes can effectively solve small- or moderate-scale instances of (2.1). Often though, the volume of input data outgrows our computational capacity, posing major challenges. Classic batch optimization methods [22, 23] perform several passes over the entire input dataset to compute the full gradient, or even the Hessian[1], in each iteration, incurring a prohibitive cost for very large problems.

Stochastic optimization methods overcome this hurdle by computing only a surrogate of the full gradient $\nabla F(\mathbf{w})$, based on a small subset of the input data. For instance, the popular SGD [24] scheme in each iteration takes a small step in a direction determined by a single, randomly selected data point. This imperfect gradient step results in smaller progress per-iteration, though manyfold in the time it would take for a batch gradient descent method to compute a full gradient [25].

Nevertheless, the approximate 'gradients' of stochastic methods introduce variance in the course of the optimization. Notably, vanilla SGD methods can deviate from the optimum, even if the initialization point is the optimum [3]. To ensure convergence, the learning rate has to decay to zero, which results to sublinear convergence rates [24], a significant degradation from the linear rate achieved by batch gradient methods.

A recent line of work [19, 3, 26, 27] has made promising steps towards the middle ground of these two extremes. A full gradient computation is occasionally interleaved with the inexpensive steps of SGD, dividing the course of the optimization in *epochs*. Within an epoch, descent directions are formed

[1]In this work, we will focus on first-order methods only. Extensions to higher-order schemes is left for future work.

as a linear combination of an approximate gradient (as in vanilla SGD) and a full gradient vector computed at the beginning of the epoch. Though not always up-to-date, the full gradient information reduces the variance of gradient estimates and provably speeds up the convergence.

Yet, as the size of the problem grows, even an infrequent computation of the full gradient may severely impede the progress of these variance-reduction approaches. For instance, when training large neural networks [28, 29, 18]), the volume of the input data rules out the possibility of computing a full gradient within any reasonable time window. Moreover, in a distributed setting, accessing the entire dataset may incur significant tail latencies [30]. On the other hand, traditional stochastic methods exhibit low convergence rates and in practice frequently fail to come close to the optimal solution in reasonable amount of time.

Contributions. The above motivate the design algorithms that try to compromise the two extremes (i) circumventing the costly computation of the full gradient, while (ii) admitting favorable convergence rate guarantees. In this work, we reconsider the computational resource allocation problem in stochastic variance-reduction schemes: *given a limited budget of atomic gradient computations, how can we utilize those resources in the course of the optimization to achieve faster convergence?* Our contributions can be summarized as follows:

(i) We propose CheapSVRG, a variant of the popular Svrg scheme [3].

Similarly to Svrg, our algorithm divides time into epochs, but at the beginning of each epoch computes only a surrogate of the full gradient using a subset of the input data. Then, it computes a sequence of estimates using a modified version of SGD steps. Overall, CheapSVRG can be seen as a family of stochastic optimization schemes encompassing Svrg and vanilla SGD. It exposes a set of tuning knobs that control trade-offs between the per-iteration computational complexity and the convergence rate.

(ii) Our theoretical analysis shows that CheapSVRG achieves linear convergence rate in expectation and up to a constant factor, that depends on the problem at hand. Our analysis is along the lines of similar results for both deterministic and stochastic schemes [31, 32].

(iii) We supplement our theoretical analysis with experiments on synthetic and real data. Empirical evaluation supports our claims for linear convergence and shows that CheapSVRG performs at least competitively with the state of the art.

2.2 Related work

There is extensive literature on classic SGD approaches. We refer the reader to [25, 33] and references therein for useful pointers. Here, we focus on works related to variance reduction using gradients, and consider only primal methods; see [17, 34, 35] for dual.

Roux et al. in [19] are among the first that considered variance reduction methods in stochastic optimization. Their proposed scheme, SAG, achieves linear convergence under smoothness and strong convexity assumptions and is computationally efficient: it performs only one atomic gradient calculation per iteration. However, it is not memory efficient[2] as it requires storing all intermediate atomic gradients to generate approximations of the full gradient and, ultimately, achieve variance reduction.

In [3], Johnson and Zhang improve upon [19] with their Stochastic Variance-Reduced Gradient (SVRG) method, which both achieves linear convergence rates and does not require the storage of the full history of atomic gradients. However, SVRG requires a full gradient computation per epoch. The S2GD method of [26] follows similar steps with SVRG, with the main difference lying in the number of iterations within each epoch, which is chosen according to a specific geometric law. Both [3] and [26] rely on the assumptions that $F(\cdot)$ is strongly convex and $f_i(\cdot)$'s are smooth.

Recently, Defazio et al. propose SAGA [16], a fast incremental gradient method in the spirit of SAG and SVRG. SAGA works for both strongly and plain convex objective functions, as well as in proximal settings. However, similarly to its predecessor [19], it does not admit low storage cost.

Finally, we note that proximal [27, 16, 36, 4] and distributed [37, 38, 14] variants have also been proposed for such stochastic settings. We leave these

[2]The authors show how to reduce memory requirements in the case where f_i depends on a linear combination of $\boldsymbol{w}$.

variations out of comparison and consider similar extensions to our approach as future work.

2.3 Our variance reduction scheme

We consider the minimization in (2.1). In the kth iteration, vanilla SGD generates a new estimate

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta_k \cdot \nabla f_{i_k}(\boldsymbol{w}_{k-1}),$$

based on the previous estimate $\boldsymbol{w}_{k-1}$ and the atomic gradient of a component f_{i_k}, where index i_k is selected uniformly at random from $\{1, \dots, n\}$. The intuition behind SGD is that in expectation its update direction aligns with the gradient descent update. But, contrary to gradient descent, SGD is not guaranteed to move towards the optimum in each single iteration. To guarantee convergence, it employs a decaying sequence of step sizes η_k, which in turn impacts the rate at which convergence occurs.

Svrg [3] alleviates the need for decreasing step size by dividing time into epochs and interleaving a computation of the full gradient between consecutive epochs. The full gradient information $\widetilde{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\widetilde{\boldsymbol{w}}_t)$, where $\widetilde{\boldsymbol{w}}_t$ is the estimate available at the beginning of the tth epoch, is used to steer the subsequent steps and counterbalance the variance introduced by the randomness of the stochastic updates. Within the tth epoch, Svrg computes a sequence of estimates $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \cdot \boldsymbol{v}_k$, where $\boldsymbol{w}_0 = \widetilde{\boldsymbol{w}}_t$, and

$$\boldsymbol{v}_k = \nabla f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}$$

Algorithm 1 CheapSVRG

```
1: Input: $\widetilde{\boldsymbol{w}}_0, \eta, s, K, T$.
2: Output: $\widetilde{\boldsymbol{w}}_T$.
3: for $t = 1, 2, \dots, T$ do
4:     Randomly select $\mathcal{S}_t \subset [n]$ with cardinality $s$.
5:     Set $\widetilde{\boldsymbol{w}} = \widetilde{\boldsymbol{w}}_{t-1}$ and $\mathcal{S} = \mathcal{S}_t$.
6:     $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}} = \frac{1}{s}\sum_{i \in \mathcal{S}} \nabla f_i(\widetilde{\boldsymbol{w}})$.
7:     $\boldsymbol{w}_0 = \widetilde{\boldsymbol{w}}$.
8:     for $k = 1, \dots, K-1$ do
9:         Randomly select $i_k \subset [n]$.
10:        $\boldsymbol{v}_k = \nabla f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}$.
11:        $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \cdot \boldsymbol{v}_k$.
12:    end for
13:    $\widetilde{\boldsymbol{w}}_t = \frac{1}{K}\sum_{j=0}^{K-1} \boldsymbol{w}_j$.
14: end for
```

is a linear combination of full and atomic gradient information. Based on this sequence, it computes the next estimate $\widetilde{\boldsymbol{w}}_{t+1}$, which is passed down to the next epoch. Note that $\boldsymbol{v}_k$ is an *unbiased* estimator of the gradient $\nabla F(\boldsymbol{w}_{k-1})$, *i.e.*, $\mathbb{E}_{i_k}[\boldsymbol{v}_k] = \nabla F(\boldsymbol{w}_{k-1})$.

As the number of components $f_i(\cdot)$ grows large, the computation of the full gradient $\widetilde{\boldsymbol{\mu}}$, at the beginning of each epoch, becomes a computational bottleneck. A natural alternative is to compute a surrogate $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}$, using only a small subset $\mathcal{S} \subset [n]$ of the input data.

Our scheme. We propose CheapSVRG, a variance-reduction stochastic optimization scheme. Our algorithm can be seen as a unifying scheme of existing stochastic methods including Svrg and vanilla SGD. The steps are outlined in Algorithm 1.

CheapSVRG divides time into epochs. The tth epoch begins at an estimate $\widetilde{\boldsymbol{w}} = \widetilde{\boldsymbol{w}}_{t-1}$, inherited from the previous epoch. For the first epoch, that estimate is given as input, $\widetilde{\boldsymbol{w}}_0 \in \mathbb{R}^p$. The algorithm selects a set $\mathcal{S}_t \subseteq [n]$ uniformly at random, with cardinality s, for some parameter $0 \leq s \leq n$. Using only the components of $F(\cdot)$ indexed by $\mathcal{S}$, it computes

$$\widetilde{\boldsymbol{\mu}}_{\mathcal{S}} \stackrel{\text{def}}{=} \frac{1}{s} \sum_{i \in \mathcal{S}} \nabla f_i(\widetilde{\boldsymbol{w}}), \tag{2.2}$$

a surrogate of the full-gradient $\widetilde{\boldsymbol{\mu}}$.

Within the tth epoch, the algorithm generates a sequence of K estimates $\boldsymbol{w}_k$, $k = 1, \ldots, K$, through an equal number of SGD-like iterations, using a modified, 'biased' update rule. Similarly to Svrg, starting from $\boldsymbol{w}_0 = \widetilde{\boldsymbol{w}}$, in the kth iteration, it computes

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \cdot \boldsymbol{v}_k,$$

where $\eta > 0$ is a constant step-size and

$$\boldsymbol{v}_k = \nabla f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}.$$

The index i_k is selected uniformly at random from $[n]$, independently across iterations.[3] The estimates obtained from the iterations of the inner loop (lines 8-12), are averaged to yield the estimate $\widetilde{\boldsymbol{w}}_t$ of the current epoch, and is used to initialize the next.

[3]In the Appendix, we also consider the case where the inner loop uses a mini-batch $\mathcal{Q}_k$ instead of a single component i_k. The cardinality $q = |\mathcal{Q}_k|$ is a user parameter.

Note that during this SGD phase, the index set $\mathcal{S}$ is fixed. Taking the expectation w.r.t. index i_k, we have

$$\mathbb{E}_{i_k}\left[\boldsymbol{v}_k\right] = \nabla F(\boldsymbol{w}_{k-1}) - \nabla F(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}.$$

Unless $\mathcal{S} = [n]$, the update direction $\boldsymbol{v}_k$ is a biased estimator of $\nabla F(\boldsymbol{w}_{k-1})$. This is a key difference from the update direction used by Svrg in [3]. Of course, since $\mathcal{S}$ is selected uniformly at random in each epoch, then across epochs $\mathbb{E}_{\mathcal{S}}\left[\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}\right] = \nabla F(\widetilde{\boldsymbol{w}})$, where the expectation is with respect to the random choice of $\mathcal{S}$. Hence, on expectation, the update direction $\boldsymbol{v}_k$ can be considered an unbiased surrogate of $\nabla F(\boldsymbol{w}_{k-1})$.

Our algorithm can be seen as a unifying framework, encompassing existing stochastic optimization methods. If the tunning parameter s is set equal to 0, the algorithm reduces to vanilla SGD, while for $s = n$, we recover Svrg. Intuitively, s establishes a trade-off between the quality of the full-gradient surrogate generated at the beginning of each epoch and the associated computational cost.

2.4 Convergence analysis

In this section, we provide a theoretical analysis of our algorithm under standard assumptions, along the lines of [31, 32]. We begin by defining those assumptions and the notation used in the remainder of this section.

Notation. We use $[n]$ to denote the set $\{1, \ldots, n\}$. For an index i in $[n]$, $\nabla f_i(\boldsymbol{w})$ denotes the atomic gradient on the ith component f_i. We use $\mathbb{E}_i[\cdot]$ to denote the expectation with respect the random variable i. With a slight abuse of notation, we use $\mathbb{E}_{[i]}[\cdot]$ to denote the expectation with respect to $i_1, \ldots, i_{K-1}$.

Assumptions. Our analysis is based on the following assumptions, which are common across several works in the stochastic optimization literature.

Assumption 1 (Lipschitz continuity of ∇f_i)**.** Each f_i in (2.1) has L-Lipschitz continuous gradients, *i.e.*, there exists a constant $L > 0$ such that for any $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$,

$$f_i(\boldsymbol{w}) \leq f_i(\boldsymbol{w}') + \nabla f_i(\boldsymbol{w}')^\top (\boldsymbol{w} - \boldsymbol{w}') + \tfrac{L}{2}\|\boldsymbol{w} - \boldsymbol{w}'\|_2^2.$$

Assumption 2 (Strong convexity of F)**.** The function $F(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{w})$ is γ-strongly convex for some constant $\gamma > 0$, *i.e.*, for any $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$,

$$F(\boldsymbol{w}) - F(\boldsymbol{w}') - \nabla F(\boldsymbol{w}')^\top (\boldsymbol{w} - \boldsymbol{w}') \geq \tfrac{\gamma}{2}\|\boldsymbol{w} - \boldsymbol{w}'\|_2^2.$$

Assumption 3 (Component-wise bounded gradient)**.** There exists $\xi > 0$ such that $\|\nabla f_i(\boldsymbol{w})\|_2 \leq \xi$, $\forall \boldsymbol{w}$ in the domain of f_i, for all $i \in [n]$.

Observe that Asm. 3 is satisfied if the components $f_i(\cdot)$ are ξ-Lipschitz functions. Alternatively, Asm. 3 is satisfied when $F(\cdot)$ is ξ'-Lipschitz function and $\max_i \{\|\nabla f_i(\boldsymbol{w})\|_2\} \leq C \cdot \|\nabla F(\boldsymbol{w})\|_2 \leq C \cdot \xi' =: \xi$. This is known as the *strong growth condition* [15].[4]

Assumption 4 (Bounded Updates)**.** For each of the estimates $\boldsymbol{w}_k$, $k = 0, \dots, K-1$, we assume that the expected distance $\mathbb{E}\left[\|\boldsymbol{w}_k - \boldsymbol{w}^*\|_2\right]$ is upper bounded by a constant. Equivalently, there exists $\zeta > 0$ such that

$$\sum_{j=0}^{K-1} \mathbb{E}_{[i]}\left[\|\boldsymbol{w}_j - \boldsymbol{w}^*\|_2\right] \leq \zeta.$$

We note that Asm. 4 is non-standard, but was required for our analysis. An analysis without this assumption is an interesting open problem.

2.4.1 Convergence Guarantees

We show that, under Asm. 1-4, the algorithm will converge –in expectation– with respect to the objective value, achieving a linear rate, up to a constant neighborhood of the optimal, depending on the configuration parameters and the problem at hand. Similar results have been reported for SGD [31], as well as deterministic incremental gradient methods [32].

Theorem 2.4.1 (Convergence)**.** *Let $\boldsymbol{w}^*$ be the optimal solution for minimization* (2.1)*. Further, let s, η, T and K be user defined parameters such that*

$$\rho \overset{def}{=} \frac{1}{\eta \cdot (1 - 4L \cdot \eta) \cdot K \cdot \gamma} + \frac{4L \cdot \eta \cdot \left(1 + \frac{1}{s}\right)}{(1 - 4L \cdot \eta)} < 1.$$

[4]This condition is rarely satisfied in many practical cases. However, similar assumptions have been used to show convergence of Gauss-Newton-based schemes [39], as well as deterministic incremental gradient methods [40, 41].

Under Asm. 1-4, CheapSVRG *outputs* $\widetilde{\boldsymbol{w}}_T$ *such that*

$$\mathbb{E}\big[F(\widetilde{\boldsymbol{w}}_T) - F(\boldsymbol{w}^*)\big] \leq \rho^T \cdot (F(\widetilde{\boldsymbol{w}}_0) - F(\boldsymbol{w}^*)) + \kappa,$$

where $\kappa \overset{def}{=} \frac{1}{1-4L\eta} \cdot \left(\frac{2\eta}{s} + \frac{\zeta}{K}\right) \cdot \max\left\{\xi, \xi^2\right\} \cdot \frac{1}{1-\rho}$.

We remark the following:

(i) The condition $\rho < 1$ ensures convergence up to a neighborhood around $\boldsymbol{w}^\star$. In turn, we require that

$$\eta < \frac{1}{4L\left((1+\theta) + \frac{1}{s}\right)} \text{ and } K > \frac{1}{(1-\theta)\eta\,(1-4L\eta)\,\gamma},$$

for appropriate $\theta \in (0, 1)$.

(ii) The value of ρ in Thm. 2.4.1 is similar to that of [3]: for sufficiently large K, there is a $(1 + \frac{1}{s})$-factor deterioration in the convergence rate, due to the parameter s. We note, however, that our result differs from [3] in that Thm. 2.4.1 guarantees convergence *up to a neighborhood around* $\boldsymbol{w}^\star$. To achieve the same convergence rate with [3], we require $\kappa = O(\rho^T)$, which in turn implies that $s = \Omega(n)$. To see this, consider a case where the condition number L is constant and $\frac{L}{\gamma} = n$. Based on the above, we need $K = \Omega(n)$. This further implies that, in order to bound the additive term in Thm. 2.4.1, $s = \Omega(n)$ is required for $O(\rho^T) \ll 1$.

(iii) When ξ is sufficiently small, Thm. 2.4.1 implies that

$$\mathbb{E}\big[F(\widetilde{\boldsymbol{w}}_T) - F(\boldsymbol{w}^*)\big] \lesssim \rho^T \cdot (F(\widetilde{\boldsymbol{w}}_0) - F(\boldsymbol{w}^*)),$$

i.e., that even $s = 1$ leads to (linear) convergence; In Sec. 4.5, we empirically show cases where even for $s = 1$, our algorithm works well in practice.

The following theorem establishes the *analytical complexity* of CheapSVRG; the proof is provided in the Appendix.

Theorem 2.4.2 (Complexity)**.** *For some accuracy parameter ϵ, if $\kappa \leq \frac{\epsilon}{2}$, then for suitable η, K, and*

$$T \geq \left(\log \tfrac{1}{\rho}\right)^{-1} \cdot \log\left(\tfrac{2(F(\widetilde{\boldsymbol{w}}_0) - F(\boldsymbol{w}^*))}{\epsilon}\right),$$

Alg. 1 outputs $\widetilde{\boldsymbol{w}}_T$ such that $\mathbb{E}\left[F(\widetilde{\boldsymbol{w}}_T) - F(\boldsymbol{w}^*)\right] \leq \epsilon$. Moreover, the total complexity is $O\left((2K + s)\log \frac{1}{\epsilon}\right)$ atomic gradient computations.

2.5 Experiments

We empirically evaluate CheapSVRG on synthetic and real data and compare mainly with Svrg [3]. We show that in some cases it improves upon existing stochastic optimization methods, and discuss its properties, strengths and weaknesses.

2.5.1 Properties of CheapSVRG

We consider a synthetic linear regression problem: given a set of training samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, we seek the

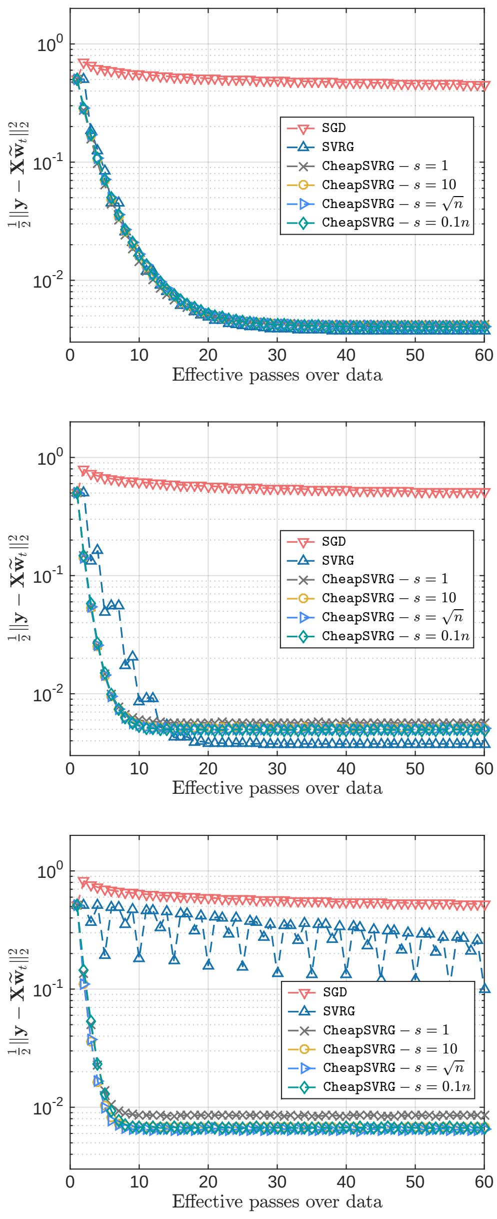


Figure 2.1: Convergence performance w.r.t. $\frac{1}{2}\|\mathbf{y} - \mathbf{X}\widetilde{\boldsymbol{w}}_t\|_2^2$ vs the number of effective data passes – *i.e.*, the number of times n data points were accessed – for $\eta = (100L)^{-1}$ (left), $\eta = (300L)^{-1}$ (middle), and $\eta = (500L)^{-1}$ (right). In all experiments, we generate noise such that $\|\boldsymbol{\varepsilon}\|_2 = 0.1$. The plotted curves depict the median over 50 Monte Carlo iterations: 10 random independent instances of (2.3), 5 executions/instance for each scheme.

solution to

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}\frac{n}{2}\left(y_i - \mathbf{x}_i^\top \boldsymbol{w}\right)^2. \tag{2.3}$$

We generate an instance of the problem as follows. First, we randomly select a point $\boldsymbol{w}^\star \in \mathbb{R}^p$ from a spherical Gaussian distribution and rescale to unit ℓ_2-norm; this point serves as our 'ground truth'. Then, we randomly generate a sequence of $\mathbf{x}_i$'s i.i.d. according to a Gaussian $\mathcal{N}\left(0, \frac{1}{n}\right)$ distribution. Let $\mathbf{X}$ be the $p \times n$ matrix formed by stacking the samples $\mathbf{x}_i$, $i = 1, \dots, n$. We compute $\mathbf{y} = \mathbf{X}\boldsymbol{w}^\star + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is a noise term drawn from $\mathcal{N}(0, \mathbf{I})$, with ℓ_2-norm rescaled to a desired value controlling the noise level.

We set $n = 2 \cdot 10^3$ and $d = 500$. Let $L = \sigma_{\max}^2(\mathbf{X})$ where $\sigma_{\max}$ denotes the maximum singular value of $\mathbf{X}$. We run (i) the classic SGD method with decreasing step size $\eta_k \propto \frac{1}{k}$, (ii) the SVRG method of Johnson and Zhang [3] and, (iii) our CHEAPSVRG for parameter values $s \in \{1,\ 10,\ \sqrt{n},\ 0.1n\}$, which covers a wide spectrum of possible configurations for s.

Step size selection. We study the effect of the step size on the performance of the algorithms; see Figure 2.1. The horizontal axis represents the number *effective passes* over the data: evaluating n component gradients, or computing a single full gradient is considered as *one* effective pass. The vertical axis depicts the progress of the objective in (2.3).

We plot the performance for three step sizes: $\eta = (cL)^{-1}$, for $c = 100, 300$ and 500. Observe that SVRG becomes slower if the step size is either too big or too small, as also reported in [3, 4]. The middle value $\eta = (300L)^{-1}$ was the best[5] for Svrg in the range we considered. Note that each algorithm achieves its peak performance for a different value of the step size. In subsequent experiments, however, we will use the above value which was best for Svrg.

Overall, we observed that CheapSVRG is more 'flexible' in the choice of the step size. In Figure 2.1 (right), with a suboptimal choice of step size, Svrg oscillates and progresses slowly. On the contrary, CheapSVRG converges nice even for $s = 1$. It is also worth noting CheapSVRG with $s = 1$, *i.e.*, effectively combining two datapoints in each stochastic update, achieves a substantial improvement compared to vanilla SGD.

Resilience to noise. We study the behavior of the algorithms with respect to the noise magnitude. We consider the cases $\|\boldsymbol{\varepsilon}\|_2 \in \{0, 0.5, 10^{-2}, 10^{-1}\}$. In Figure 2.3, we focus on four distinct noise levels and plot the distance of the estimate from the ground truth $\boldsymbol{w}^\star$ vs the number of effective data passes. For SGD, we use the sequence of step sizes $\eta_k = 0.1 \cdot L^{-1} \cdot k^{-1}$.

We also note the following surprising result: in the noiseless case, it appears that $s = 1$ is sufficient for linear convergence in practice; see Figure 2.3. In contrast, CheapSVRG is less resilient to noise than Svrg – however, we can still get to a good solution with less computational complexity per iteration.

[5]Determined via binary search.

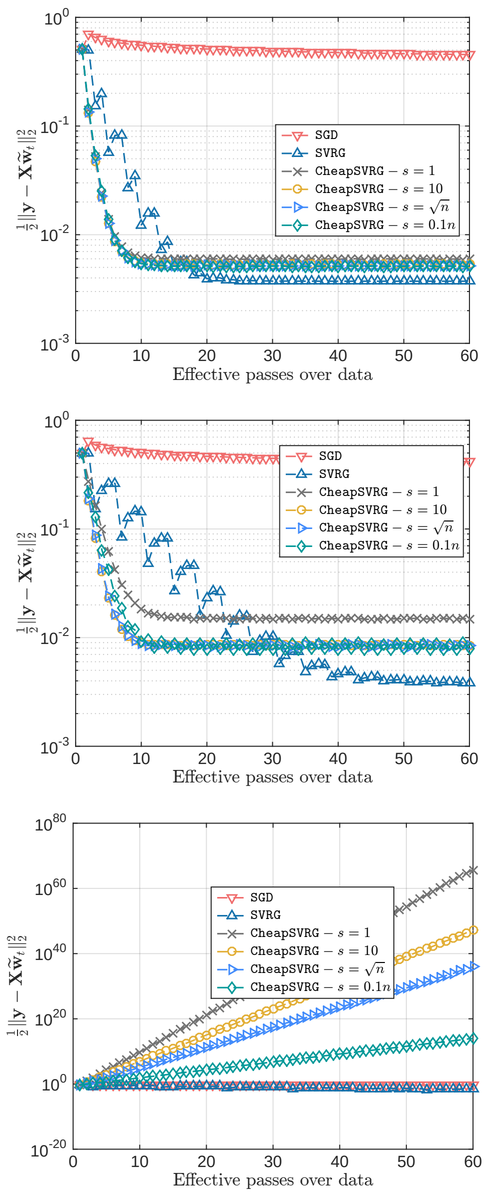


Figure 2.2: Convergence performance w.r.t. $\frac{1}{2}\|\mathbf{y} - \mathbf{X}\widetilde{\boldsymbol{w}}_t\|_2^2$ vs. effective number of passes over the data. We set an upper bound on total atomic gradient calculations spent as $\nabla_{\text{total}} = 60n = 12 \cdot 10^4$ and vary the percentage of these resources in the inner loop two-stage SGD schemes. Left: `perc` $= 60\%$. Middle : `perc` $= 75\%$. Right: `perc` $= 90\%$. In all experiments, we set $\|\boldsymbol{\varepsilon}\|_2 = 0.1$. The plotted curves depict the median over 50 Monte Carlo iterations: 10 random independent instances of (2.3), 5 executions/instance for each scheme.

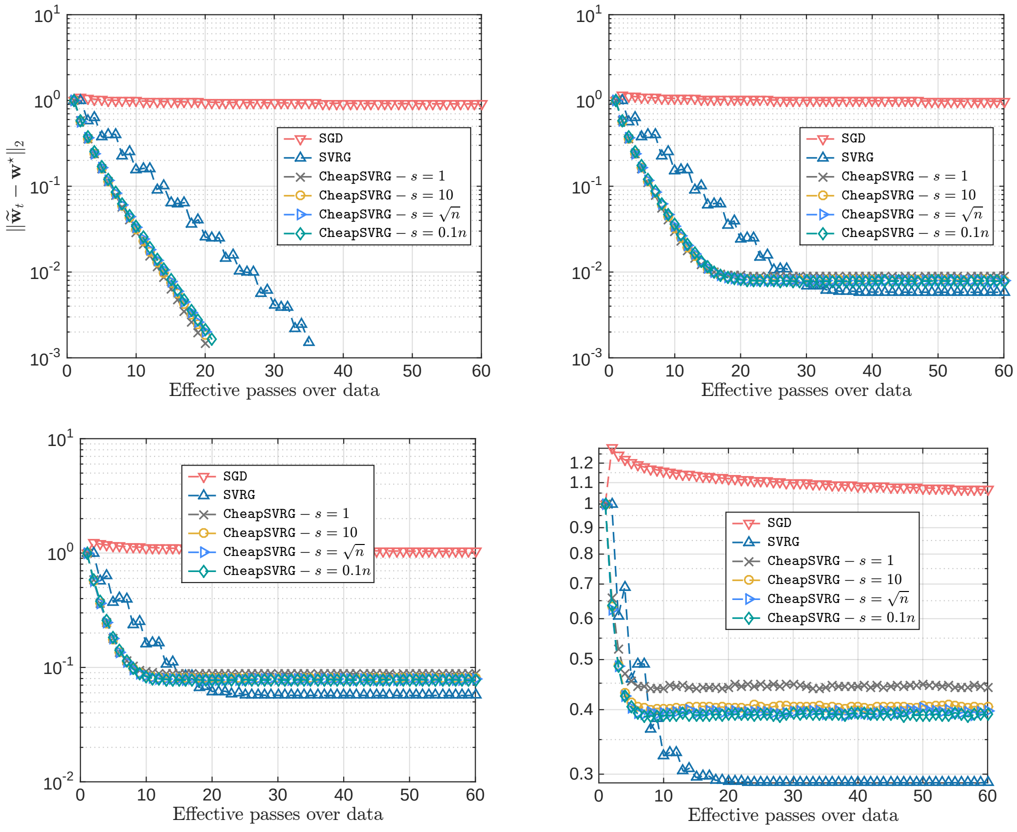


Figure 2.3: Distance from the optimum vs the number of effective data passes for the linear regression problem. We generate 10 independent random instances of (2.3). From left to right, we use noise noise $\boldsymbol{\varepsilon}$ with standard deviation $\|\boldsymbol{\varepsilon}\|_2 = 0$ (noiseless), $\|\boldsymbol{\varepsilon}\|_2 = 10^{-2}$, $\|\boldsymbol{\varepsilon}\|_2 = 10^{-1}$, and $\|\boldsymbol{\varepsilon}\|_2 = 0.5$. Each scheme is executed 5 times/instance. We plot the median over the 50 Monte Carlo iterations.

Number of inner loop iterations. Let ∇_{total} denote a budget of atomic gradient computations. We study how the objective value decreases with respect to percentage `perc` of the budget allocated to the inner loop. We first run a classic gradient descent with step size $\eta = \frac{1}{L}$ which converges within ~ 60 iterations. Based on this, we choose our global budget to be $\nabla_{\text{total}} = 60n = 12 \cdot 10^4$. We consider the following values for `perc`: $60\%, 75\%, 90\%$. *E.g.*, when `perc` $= 90\%$, only 12000 atomic gradient calculations are spent in outer loop iterations. The results are depicted in Fig. 2.2.

We observe that convergence is slower as fewer computations are spent in outer iterations. Also, in contrast to Svrg, our algorithm appears to be sensitive to the choice of `perc`: for `perc` $= 90\%$, our scheme diverges, while Svrg finds relatively good solution.

2.5.2 ℓ_2-regularized logistic regression

We consider the regularized logistic regression problem, *i.e.*, the minimization

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \cdot \mathbf{x}_i^\top \boldsymbol{w}}\right) + \lambda \cdot \|\boldsymbol{w}\|_2^2. \tag{2.4}$$

Here, $(y_i, \mathbf{x}_i) \in \{-1, 1\} \times \mathbb{R}^d$, where $\mathbf{y}_i$ indicates the binary label in a classification problem, $\boldsymbol{w}$ represents the predictor, and $\lambda > 0$ is a regularization parameter.

We focus on the training loss in such a task. From [3], we already know that such two-stage SGD schemes perform better than vanilla SGD.

Dataset	n	d
`marti0`	1024	500
`reged0`	999	500
`sido0`	12678	4932

Table 2.1: Summary of datasets [1].

We use the real datasets listed in Table 2.1. We pre-process the data so that $\|\mathbf{x}_i\|_2 = 1, \forall i$, as in [4]. This leads to an upper bound on Lipschitz constants for each f_i such that $L_i \leq L := \frac{1}{4}$. We set $\eta = 0.1/L$ for all algorithms under consideration, according to [3, 4], `perc` $= 75\%$ and, $\lambda = 10^{-6}$ for all problem cases. Table 2.1 depicts the convergence results for the `marti0`, `reged0` and `sido0` datasets. CheapSVRG achieves comparable performance to Svrg, while requiring less computational 'effort' per epoch: though smaller values of s, such that $s = 1$ or $s = 10$, lead to slower convergence, CheapSVRG still performs steps towards the solution, while the complexity per epoch is significantly diminished.

2.6 Conclusions

We proposed CheapSVRG, a new variance-reduction scheme for stochastic optimization, based on [3]. The main difference is that instead of computing a full gradient in each epoch, our scheme computes a surrogate utilizing only part of the data, thus, reducing the per-epoch complexity. CheapSVRG comes with convergence guarantees: under assumptions, it achieves a linear convergence rate up to some constant neighborhood of the optimal. We em-

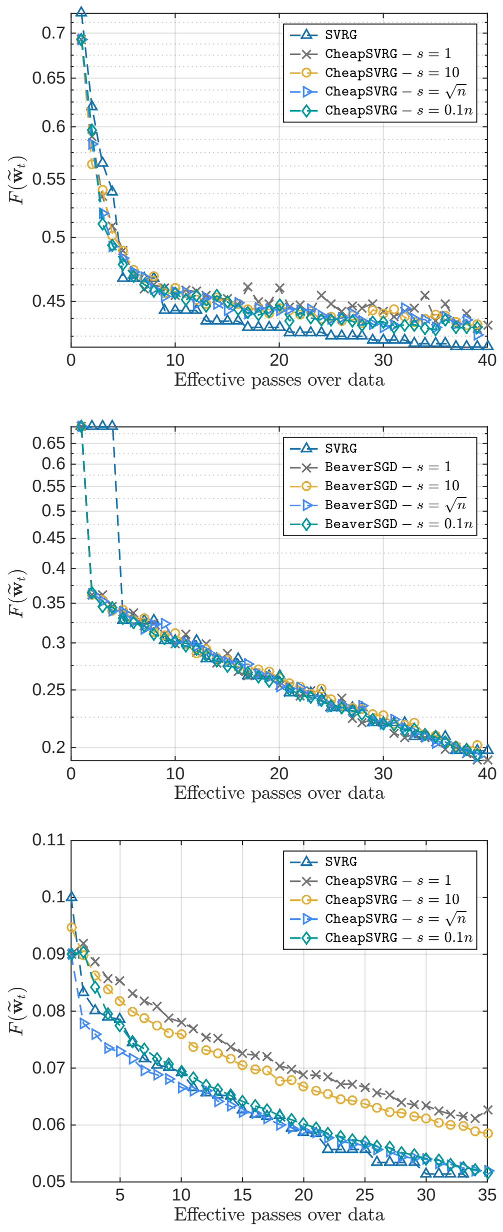


Figure 2.4: Convergence performance of algorithms for the ℓ_2-regularized logistic regression objective. From left to right, we used the `marti0`, `reged0`, and `sido0` dataset; the description of the datasets is given in Table 2.1. Plots depict $F(\widetilde{\boldsymbol{w}}_t)$ vs the number of effective data passes. We use step size $\eta = 0.1/L$ for all algorithms, as suggested by [3, 4]. The curves depict the median over 10 Monte Carlo iterations.

pirically evaluated our method and discussed its strengths and weaknesses.

There are several future directions. In the theory front, it would be interesting to maintain similar convergence guarantees under fewer assumptions, extend our results beyond the smooth convex optimization, *e.g.*, to the proximal setting, or develop distributed variants. Finally, we seek to apply our CheapSVRG to large-scale problems, *e.g.*, for training large neural networks. We hope that this will help us better understand the properties of CheapSVRG and the trade-offs associated with its various configuration parameters.

Chapter 3

On the Generalization of Adaptive Methods

Over-parameterization and adaptive methods have played a crucial role in the success of deep learning in the last decade. The widespread use of over-parameterization has forced us to rethink generalization by bringing forth new phenomena, such as implicit regularization of optimization algorithms and double descent with training progression. A series of recent works have started to shed light on these areas in the quest to understand – *why do neural networks generalize well?* The setting of over-parameterized linear regression has provided key insights into understanding this mysterious behavior of neural networks.

In this paper, we aim to characterize the performance of adaptive methods in the over-parameterized linear regression setting. First, we focus on two sub-classes of adaptive methods depending on their generalization performance. For the first class of adaptive methods, the parameter vector remains in the span of the data and converges to the minimum norm solution like

Parts of this chapter was published in the Integration of Deep Learning Theories Workshop at Neurips, 2018 and is also available at [42]. The author was a part of formulating the problem, analyzing the theory, designing the experiments, writing up the results, and performing the simulations presented in the paper.

gradient descent (GD). On the other hand, for the second class of adaptive methods, the gradient rotation caused by the pre-conditioner matrix results in an in-span component of the parameter vector that converges to the minimum norm solution and the out-of-span component that saturates. Our experiments on over-parameterized linear regression and deep neural networks support this theory.

3.1 Introduction

The success of deep learning has uncovered a new mystery of *benign overfitting* [43, 44], i.e., systems with a large number of parameters can not only achieve zero training error but are also able to generalize well. Also, over-parameterized systems exhibit a double descent-behavior [43, 45]; *as the number of parameters/epochs increases, the test error first decreases, then increases before falling again.* This goes against the conventional wisdom of overfitting in machine learning, which stems from the classical bias-variance tradeoff [46, 47, 48].

In the absence of explicit regularization, a typical over-parameterized setting possesses multiple global minima. Classical gradient descent based methods can achieve one of these many global minima [49, 50, 51], however not all optima generalize equally. [52, 53, 54] suggest many practical approaches to improve generalization; however, there remains a considerable gap between theory and practice [55, 49].

In this paper, we will focus on two categories of optimization algorithms: pure gradient descent based (non-adaptive[1]) methods and adaptive methods. The primary distinguishing factor between these two methods is determined by the update step. For the class of non-adaptive methods, the expected gradient update step is given as follows:

$$\mathbb{E}\left[\boldsymbol{w}(t+1)|\boldsymbol{w}(t)\right] = \boldsymbol{w}(t) - \eta \nabla f(\boldsymbol{w}(t)), \tag{3.1}$$

where $\boldsymbol{w}(t)$ indicates the estimate of the underlying parameter vector, η represents the learning rate and $f(\boldsymbol{w}(t)), \nabla f(\boldsymbol{w}(t))$ represent the loss function and its gradient, respectively. Popular methods like gradient descent, stochastic gradient descent (SGD), batch gradient descent fall under this class. Training any model using non-adaptive methods involves tuning over many hyperparameters, of which *step size* is the most essential one [29, 56]. The step size could be set as constant, or could be changing per iteration $\eta(t)$ [53], usually based on a predefined learning rate schedule [25, 57, 58].

During the past decade, we have also witnessed the rise of a family of algorithms called adaptive methods that argue for *automatic* hyper-parameter adaptation [59] during training (including step size). The list includes AdaGrad [60], Adam [61], AdaDelta [62], RMSProp [63], AdaMax [61], Nadam [64], just to name a few. These algorithms utilize current and past gradient information $\{\nabla f(\boldsymbol{w}(i))\}_{i=t}^{k}$, for $t < k$, to design preconditioning matrices $\boldsymbol{D}(t) \succeq 0$ that better pinpoint the local curvature of the objective function as

[1]Now onwards, optimization methods that satisfy equation (3.1) will be referred to as non-adaptive purposes.

follows:

$$\mathbb{E}\left[\boldsymbol{w}(t+1)|\boldsymbol{w}(t)\right] = \boldsymbol{w}(t) - \eta \boldsymbol{D}(t)\nabla f(\boldsymbol{w}(t)) \tag{3.2}$$

Usually, the main argument for using adaptive methods is that $\boldsymbol{D}(t)$ eliminates pre-setting a learning rate schedule, or diminishes initial bad step size choices, thus, detaching the time-consuming part of step size tuning from the practitioner [65].

[66] was one of the first papers to discuss the implicit bias introduced by optimization methods for over-parameterized systems and how the choice of optimization algorithm affects the global minima it attains. However, the generalization behavior of these optimization methods remains a mystery. As a result, researchers have re-focussed their attention on understanding the most straightforward over-parameterized setting of linear regression [43, 67, 68, 44] as a first step in unraveling the mysterious behavior of neural networks.

Gradient descent-based methods converge to the minimum norm interpolated solution [44] for over-parameterized linear regression. Under certain assumptions on the data distribution, the minimum norm solution achieves near-optimal accuracy for unseen data [43]. Unlike SGD, the presence of $\boldsymbol{D}(t)$ in adaptive methods can alter the span of the final converged solution in the presence of any non-trivial initialization, which makes the task of commenting on adaptive methods challenging.

Despite being a key reason behind the success of deep learning, the convergence behavior of adaptive methods is not well understood. The con-

$d = 50$		GD	AM1	AM2	AM3
$n = 10$	Training Error	$1.27 \cdot 10^{-28}$	$\mathbf{1.42 \cdot 10^{-29}}$	$8.64 \cdot 10^{-4}$	$8.64 \cdot 10^{-29}$
	Test Error	81.56	**76.94**	79.62	81.65
	$\|\boldsymbol{w} - \boldsymbol{w}^*\|$	9.08	**8.92**	9.03	9.08
$n = 40$	Training Error	$4.77 \cdot 10^{-5}$	$\mathbf{6.07 \cdot 10^{-7}}$	$3.31 \cdot 10^{-3}$	$8.64 \cdot 10^{-4}$
	Test Error	**18.62**	19.56	20.35	18.65
	$\|\boldsymbol{w} - \boldsymbol{w}^*\|$	**4.31**	4.37	4.51	**4.31**

Table 3.1: Table illustrating differing generalization guarantees of three distinct Adaptive Methods (AM) with SGD in over-parameterized setting, i.e. $d > n$, where n: number of examples, d: dimension,

vergence bounds for most adaptive methods hold for only a specific preconditioner matrix [61, 69, 66]. Besides, theoretical guarantees for adaptive methods often minimize regret [70, 60], which makes it further challenging to comment on the generalization of adaptive methods. As a result, the generalization of adaptive methods for a general $\boldsymbol{D}(t)$ remains an open problem even for an over-parameterized linear regression setting. In this paper, we aim to explicitly characterize the sub-class of adaptive methods that mimic the *convergence*, and *generalization* behaviors seen in SGD and the sub-class that does not. In addition, we observe a double descent like phenomena for a sub-class of adaptive methods as the number of training epochs increases.

In this paper, we would like to understand *how adaptive methods affect generalization guarantees of over-parameterized problems.* To motivate this, we consider a toy example for simple linear regression in the under-determined/over-parameterized framework in Table 3.1. As is evident, some adaptive methods have the same generalization as SGD, while others can yield quite different generalization guarantees.

Key Contributions: For the *theoretical contribution*, we focus on over-parameterized linear regression. Here, plain gradient descent methods converge to the *minimum Euclidean norm solution*, while adaptive methods may or may not. In this paper, we provide explicit conditions on the structure of pre-conditioner matrices, $\boldsymbol{D}(t)$, which allow us to distinguish between two classes of adaptive methods, the ones which behave similarly to SGD and the ones that do not. Based on these conditions, we compare the generalization performance between adaptive and non-adaptive methods.

For the *experimental component*, we begin by revisiting the mystery posed by Table 3.1, and demonstrate that the experimental results are in line with our theoretical guarantees. Further, we show using a toy example that the adaptive methods can have a superior generalization performance than SGD. The discussion *"which method is provably better"*, however, is inconclusive and ultimately depends on the problem/application at hand. Lastly, we empirically demonstrate the validity of our claims for over-parameterized neural networks as well and recommend exercising caution when proposing or choosing adaptive methods for training, depending on the goal in hand.

3.2 Problem Setup

Notation: For any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, A_{pq} indicates the element corresponding to the p-th row and q-th column. The $\text{rank}(\boldsymbol{A})$ denotes the rank of $\boldsymbol{A}$.

For a sequence of matrices $\boldsymbol{A}_0$ to $\boldsymbol{A}_n$, we have the definition $\prod_{k=i+m}^{i} \boldsymbol{A}_k = \boldsymbol{A}_{(i+m)}\boldsymbol{A}_{(i+m-1)} \dots \boldsymbol{A}_i$. Note that, $a(t)$ indicates the value of the the function $a(\cdot)$ after the t-th update. Note that λ without any subscript indicates the regularizer, and λ_i with a subscript denotes the i^{th} eigenvalue. We consider an over-parameterized noisy linear regression (possibly with regularization), where the relationship between the data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, the noise vector $\boldsymbol{\zeta} \in \mathbb{R}^n$, and the labels $\boldsymbol{y} \in \mathbb{R}^n$ is as follows:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}^{\star} + \boldsymbol{\zeta}. \tag{3.3}$$

We are concerned with the following optimization problem:

$$f(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left\{ \mathbb{E}\left[\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2\right] + \tfrac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \right\}. \tag{3.4}$$

In particular, we study the convergence of the following iterative updates

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}(t)\nabla f(\boldsymbol{w}(t)), \tag{3.5}$$

where the pre-conditioner matrices are *bounded*, *positive definite* and hence full rank; i.e., $\inf_t \operatorname{rank}(\boldsymbol{D}(t)) = d$.

There are two different settings, depending on the number of samples and dimension:

- **Over-parameterized case**: The system is assumed over-parameterized; if $R = \operatorname{rank}(\boldsymbol{X}) < d$ or there are more parameters than the number of effective samples: $d \geq n$. In this case, assuming that $\boldsymbol{X}$ is in general position, $\boldsymbol{X}\boldsymbol{X}^\top$ is full rank.

- **Under-parameterized case**: Here, the effective number of samples is larger than the number of parameters: $n \geq d$. In this case, usually $\boldsymbol{X}^\top \boldsymbol{X}$ is full rank.

The most studied case is when $n \geq d$: the problem has solution $\boldsymbol{w}^* = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$, under full rankness assumption on $\boldsymbol{X}$. In the case where the problem is over-parameterized $d \geq n$, there is a solution of similar form that has received significant attention, despite the infinite number of optimal solutions: This is the so-called *minimum $\ell 2$ norm* solution. The optimization instance to obtain this solution is:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \; \|\boldsymbol{w}\|_2^2 \;\; \text{subject to} \;\; \boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}.$$

Let us denote its solution as $\boldsymbol{w}_{\text{mn}}$, which has the form $\boldsymbol{w}_{\text{mn}} = \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{y}$. Any other solution has to have equal or larger Euclidean norm than $\boldsymbol{w}_{\text{mn}}$.

Observe that the two solutions, $\boldsymbol{w}^*$ and $\boldsymbol{w}_{\text{mn}}$, differ between the two cases: in the under-parameterized case, the matrix $\boldsymbol{X}^\top \boldsymbol{X}$ is well-defined (full-rank) and has an inverse, while in the over-parameterized case, the matrix $\boldsymbol{X}\boldsymbol{X}^\top$ is full rank. Importantly, there are differences on how we obtain these solutions in an iterative fashion. We next show how both simple and adaptive gradient descent algorithms find $\boldsymbol{w}^*$ for well-determined systems. This does not hold for the over-parameterized case: there are infinite solutions, and the question which one they select is central in the recent literature [71, 72, 66].

Studying iterative routines in simple tasks provides intuitions on how they might perform in more complex problems, such as neural networks. Next, we set the background with the well-known under-parameterized linear regression, before we move onto the over-parameterized case.

3.2.1 Non-adaptive Methods in Under-parameterized Linear Regression

Here, $n \geq d$ and $\boldsymbol{X}^\top \boldsymbol{X}$ is assumed to be full rank. Simple gradient descent with step size $\eta > 0$ satisfies: $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \cdot \nabla f(\boldsymbol{w}(t)) = \boldsymbol{w}(t) - \eta \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{w}(t) - \boldsymbol{y})$. Unfolding for T iterations, we get:

$$\boldsymbol{w}(T) = \left(\sum_{i=1}^{t} (-1)^{i-1} \cdot \binom{t}{i} \cdot \eta^i \cdot (\boldsymbol{X}^\top \boldsymbol{X})^{i-1} \right) \boldsymbol{X}^\top y.$$

The expression in the parentheses satisfies:

$$\sum_{i=1}^{T} (-1)^{i-1} \cdot \binom{T}{i} \cdot \eta^i \cdot (\boldsymbol{X}^\top \boldsymbol{X})^{i-1} = (-\boldsymbol{X}^\top \boldsymbol{X})^{-1} \cdot \left((I - \eta \boldsymbol{X}^\top \boldsymbol{X})^T - I \right)$$

Therefore, we get the closed form solution:

$$\boldsymbol{w}(T) = (-\boldsymbol{X}^\top \boldsymbol{X})^{-1} \cdot \left((I - \eta \boldsymbol{X}^\top \boldsymbol{X})^T - I \right) \boldsymbol{X}^\top \boldsymbol{y}$$

. In order to prove that gradient descent converges to the minimum norm solution, we need to prove that:

$$(I - \eta \boldsymbol{X}^\top \boldsymbol{X})^T - I = -I \quad \Rightarrow \quad (I - \eta \boldsymbol{X}^\top \boldsymbol{X})^T \quad \xrightarrow{n,T \text{ large}} \quad 0.$$

This is equivalent to showing that $\|(I - \eta \boldsymbol{X}^\top \boldsymbol{X})^T\|_2 \to 0$. From optimization theory [73], we need $\eta < \frac{1}{\lambda_1(\boldsymbol{X}^\top \boldsymbol{X})}$ for convergence, where $\lambda_i(\cdot)$ denotes the eigenvalues of the argument. Then, $\boldsymbol{H} := \boldsymbol{I} - \eta \boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{d \times d}$ has spectral norm that is smaller than 1, *i.e.*, $\|\boldsymbol{H}\| \leq 1$. Combining the above, we use the following theorem.

Theorem 3.2.1 (Behavior of square matrix $\|\boldsymbol{M}^T\|_2$). *[74, 75] Let $\boldsymbol{M}$ is a $d \times d$ matrix. Let $\tau(\boldsymbol{M}) = \max_i |\lambda_i(\boldsymbol{M})|$ denote the spectral radius of the matrix $\boldsymbol{M}$. Then, there exists a sequence $\varepsilon(t) \geq 0$ such that: $\|\boldsymbol{M}^T\|_2 \leq (\tau(\boldsymbol{M}) + \varepsilon(t))^T$, and $\lim_{T\to\infty} \varepsilon(t) = 0$.*

Using the above theorem, $\boldsymbol{H}$ has $\tau(\boldsymbol{H}) < 1$. Further, for sufficiently large $t < T$, $\varepsilon(t)$ has a small value such that $\tau(\boldsymbol{H}) + \varepsilon(t) < 1$; *i.e.*, after some $t_1 < T$, $(\tau(\boldsymbol{H}) + \varepsilon_{t_1})^{t_1}$, will be less than zero, converging to zero for increasing t_1. As T is going towards infinity, this concludes the proof, and leads to the *left inverse* solution: $\boldsymbol{w}_\infty = (-\boldsymbol{X}^\top \boldsymbol{X})^{-1} \cdot (-I)\, \boldsymbol{X}^\top y = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top y \equiv \boldsymbol{w}^*$, as $T \to \infty$. This is identical to the closed for solution of linear regression.

3.2.2 Adaptive Methods in Under-parameterized Linear Regression

When $\boldsymbol{D}(t)$ is varying, we end up with the following proposition (folklore); the proof is in Section B.3.:

Proposition 3.2.2. *Consider the under-parameterized linear regression setting with data matrix $\boldsymbol{X}$, noise $\boldsymbol{\zeta}$, and regularizer $\lambda > 0$. Suppose for all $t \geq 0$, the pre-conditioner matrix $\boldsymbol{D}(t)$ is positive definite, i.e. $\boldsymbol{D}(t) \succ 0$. Assume the recursion $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}(t) \nabla f(\boldsymbol{w}(t))$. Then, after T iterations, $\boldsymbol{w}(t)$ satisfies:*

$$\boldsymbol{w}(t) = \left(-\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \cdot \left(\prod_{i=T-1}^{0} \left(I - \eta \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{D}(i)\right) - \boldsymbol{I} \right) \boldsymbol{X}^\top \boldsymbol{y}.$$

Using Theorem 3.2.1, we can again infer that, for sufficiently large T and for sufficiently small $\eta < \max_i \frac{1}{\lambda_1(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{D}_i)}$, such that $\|I - \eta\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{D}_i\| < 1 \;\forall\; i$, we have: $\prod_{i=T-1}^{0}\left(I - \eta\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{D}_i\right) \;\to\; 0$. Thus, for sufficiently large T and assuming $\eta < \max_i \frac{1}{\lambda_1(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{D}_i)}, \forall i$: $w_\infty = (\boldsymbol{X}^\top\boldsymbol{X})^{-1} \cdot \boldsymbol{X}^\top\boldsymbol{y} \equiv \boldsymbol{w}^*$, which is the same as the plain gradient descent approach. Thus, in this case, under proper η assumptions (which might seem stricter than plain gradient descent), *adaptive methods have the same generalization capabilities as gradient descent.*

3.3 Over-parameterized linear regression

Over-parameterized systems possess more degrees of freedom than the number of training samples. As a result, we know that standard gradient descent based methods fit perfectly to the training set.

3.3.1 Performance on Training Set for Non-Adaptive Methods

For completeness, we briefly provide the analysis for the over-parameterized setting, where $d \geq n$ and $\boldsymbol{X}\boldsymbol{X}^\top$ is assumed to be full rank. By inspection, unfolding gradient descent recursion gives after T iterations:

$$\boldsymbol{w}(T) = \boldsymbol{X}^\top \left(\sum_{i=1}^{T} (-1)^{i-1} \cdot \binom{T}{i} \cdot \eta^i \cdot (\boldsymbol{X}\boldsymbol{X}^\top)^{i-1} \right) \boldsymbol{y}.$$

Similarly, the summation can be simplified to:

$$\sum_{i=1}^{T} (-1)^{i-1} \cdot \binom{T}{i} \cdot \eta^i \cdot (\boldsymbol{X}\boldsymbol{X}^\top)^{i-1} = (-\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \cdot \left((\boldsymbol{I} - \eta\boldsymbol{X}\boldsymbol{X}^\top)^K - I \right),$$

and, therefore:

$$\boldsymbol{w}(T) = \boldsymbol{X}^\top (-\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \cdot \left((\boldsymbol{I} - \eta\boldsymbol{X}\boldsymbol{X}^\top)^K - \boldsymbol{I} \right) \boldsymbol{y}.$$

Under similar assumption on the spectral norm of $(\boldsymbol{I} - \eta \boldsymbol{X}\boldsymbol{X}^\top)^K$ and using Theorem 3.2.1, we obtain the *right inverse* solution: $\boldsymbol{w}_\infty = \boldsymbol{X}^\top(-\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \cdot (-\boldsymbol{I})\boldsymbol{y} = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{y} \equiv \boldsymbol{w}_{\text{mn}}$, as $K \to \infty$. *Bottomline, in both cases, gradient descent converges to left and right inverse solutions, related to the Moore-Penrose inverse.*

Before we discuss the generalization of adaptive methods in over parameterized settings, let us first answer the following question: *what is the predictive power of adaptive methods within the training set?*

3.3.2 Performance on Training Set for Adaptive Methods

For linear regression with $\ell 2$ norm regularization, we observe that adaptive methods with any full rank pre-conditioner matrix $\boldsymbol{D}(t)$ will converge to the same solution as its non-adaptive counterpart and thus mimic their performance. However, for unregularized linear regression, adaptive methods can converge to entirely different solutions than SGD. Before we discuss generalization, we will first show that both SGD and adaptive methods can achieve zero training error despite attaining different stationary points.

Proposition 3.3.1. *Consider the over-parameterized linear regression setting with data matrix $\boldsymbol{X}$, noise $\boldsymbol{\zeta}$, and regularizer $\lambda > 0$. Suppose for all $t \geq 0$, the pre-conditioner matrix $\boldsymbol{D}(t)$ is positive definite, i.e. $\boldsymbol{D}(t) \succ 0$. Assume the recursion $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}(t)\nabla f(\boldsymbol{w}(t))$. If $\boldsymbol{D}(t)$ is positive definite for* *all t and the regularizer $\lambda = 0$, then,*

$$\lim_{t\to\infty} \widehat{\boldsymbol{y}}(t) = \boldsymbol{y}.$$

where $\widehat{\boldsymbol{y}}(t) = \boldsymbol{X}\hat{\boldsymbol{w}}$ and $\hat{w} = \lim_{T\to\infty} \boldsymbol{w}_t$.

Proposition 3.3.1 implies that adaptive methods with full-rank positive definite preconditioners perform as well as their pure gradient based counter parts when it comes to fitting their training data. However, this proposition gives no information regarding the distance between the converged $\boldsymbol{w}(t)$ and $\boldsymbol{w}^\star$.

Proposition 3.3.2. *Consider the over-parameterized linear regression setting with data matrix $\boldsymbol{X}$, noise $\boldsymbol{\zeta}$, and regularizer $\lambda > 0$. Suppose for all $t \geq 0$, the pre-conditioner matrix $\boldsymbol{D}(t)$ is positive definite, i.e. $\boldsymbol{D}(t) \succ 0$. Assume the recursion $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}_t \nabla f(\boldsymbol{w}(t))$. If $\boldsymbol{D}(t)$ is positive definite for all t and the regularizer satisfies $\lambda > 0$, then,*

$$\lim_{t\to\infty} \boldsymbol{w}(t) = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

Proposition 3.3.2 implies that for the regularized case, the problem becomes strongly convex and the adaptive methods converge to the same solution as their SGD counterparts under certain conditions.

To summarize, for linear regression with ℓ_2-norm regularization, we observe that adaptive methods with any full rank pre-conditioner matrix $\boldsymbol{D}(t)$ will converge to the same solution as its non-adaptive counterpart and thus mimic their performance. However, for unregularized linear regression, adaptive methods can converge to entirely different solutions than SGD. Both SGD and adaptive methods can achieve zero training error despite attaining different stationary points. This leads us to the following question: *What is the predictive power of adaptive methods on unseen data for over-parameterized linear regression?*

3.4 Performance on Unseen Data

Our primary focus in this chapter is to understand the generalization capabilities of adaptive methods. We observe that the generalization depends on two key factors: *i) Does $\boldsymbol{w}^\star$ lie in the span of the data matrix, $\boldsymbol{X}$? ii) How does pre-multiplying with the pre-conditioner matrix alter the span of final converged $\boldsymbol{w}$?*

3.4.1 Spectral Representation

The switch to the spectral domain allows us to simplify and understand the relationship between the final converged solution with the span of data matrix $\boldsymbol{X}$, pre-conditioner matrix $\tilde{\boldsymbol{D}}(t)$ and the initialization $w(0)$. We express the data matrix using its singular value decomposition (SVD): $\boldsymbol{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{u}_r \boldsymbol{v}_r^T$, $\lambda_r \neq 0$ for all r where $\lambda_r, \boldsymbol{u}_r, \boldsymbol{v_r}$ represent the r^{th} largest eigenvalue and the corresponding right and left eigenvectors respectively. We complete the basis using the left eigenvectors of the data matrix to form a complete orthogonal spectral basis of $\mathbb{R}^d$, $\{\boldsymbol{v}_r : r = 1 \ldots, d\}$ form the basis vectors and denote it by $\boldsymbol{V}$. Similarly, $\boldsymbol{U}$ forms the complete orthogonal spectral basis of $\mathbb{R}^n$ using the right eigenvectors of the data matrix as $\{\boldsymbol{u}_r : r = 1 \ldots, n\}$. The eigenvalue matrix is $\boldsymbol{\Lambda}$ where $\boldsymbol{\Lambda}_{rr} = \lambda_r$ if $1 \leq r \leq R$ and 0 otherwise. We next express useful quantities in the above bases in Table 3.2.

Data matrix	$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T$
True parameter	$\boldsymbol{w}^\star = \boldsymbol{V}\tilde{\boldsymbol{w}}^\star$
Noise vector	$\boldsymbol{\zeta} = \boldsymbol{U}\tilde{\boldsymbol{\zeta}}$
Adaptive pre-conditioner matrices	$\boldsymbol{D}(t) = \sum_{r=1}^{d}\sum_{s=1}^{d} \tilde{D}_{rs}(t)\boldsymbol{v}_r\boldsymbol{v}_s^T,$
	$\tilde{\boldsymbol{D}}(t) = \boldsymbol{V}^T\boldsymbol{D}(t)\boldsymbol{V}, \tilde{\boldsymbol{D}}(t) \in \mathbb{R}^{d\times d}$
Weight vectors	$\boldsymbol{w}(t) = \boldsymbol{V}\tilde{\boldsymbol{w}}(t)$

Table 3.2: Notation in spectral domain

The definition of adaptive pre-conditioner matrices in the above table holds since $\boldsymbol{V}$ represents a complete orthogonal spectral basis of $\mathbb{R}^d$. Additionally, we also have the following property, where we show that pre- and post-multiplication by an orthogonal matrix $\boldsymbol{V}$ does not alter the eigenvalues of the original matrix. In other words, we have the following proposition:

Proposition 3.4.1. *The set of eigenvalues for $\tilde{\boldsymbol{D}}(t)$ is identical to the set of eigenvalues of $\boldsymbol{D}(t)$.*

The proof of Proposition 3.4.1 is available in Appendix.

3.4.2 Closed Form Expression for the Iterates

Our objective is to understand how the iterates evolve depending on the space spanned by the data matrix. First, we establish a closed-form expression for the updates of the vector $\tilde{\boldsymbol{w}}(t)$.

Proposition 3.4.2. *Consider the over-parameterized linear regression setting with data matrix $\boldsymbol{X}$, noise $\boldsymbol{\zeta}$, and regularizer $\lambda > 0$. If the pre-conditioner matrix $\boldsymbol{D}(t) \succ 0$ for all $t \geq 0$, then, for any $T \geq 0$, the iterate $\tilde{\boldsymbol{w}}(T)$ admits the following closed form expression:*

$$\begin{aligned}\tilde{\boldsymbol{w}}(T) = &\prod_{i=0}^{T-1} \Big(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}(i)(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I})\Big)\, \tilde{\boldsymbol{w}}(0) \\ &+ \sum_{i=0}^{T-1} \prod_{j=(i+1)}^{T-1} \Big(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}(j)(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I})\Big)\, \eta\tilde{\boldsymbol{D}}(i)(\boldsymbol{\Lambda}^2\tilde{\boldsymbol{w}}^\star + \boldsymbol{\Lambda}\boldsymbol{\zeta}) \end{aligned} \tag{3.6}$$

The final expression of $\boldsymbol{w}(T)$ implies that the final solution depends on the initialization point, the span of the data matrix in $\mathbb{R}^d$ space, and the pre-conditioner matrix. Further, the closed-form indicates that the presence of pre-conditioning matrices $\tilde{\boldsymbol{D}}(j)$ may cause $\boldsymbol{w}(t)$ to lie outside of the span of the data in the complete $\mathbb{R}^d$ space.

We observe that the presence or absence of regularizer can significantly alter the stationary points to which adaptive methods converge. In the presence of ℓ_2-norm regularization, we observe that the adaptive methods converge to the same solution independent of the initialization or the step-size. However, in the absence of regularization, things are not as straight-forward. Here, we will try to capture the convergence of over parameterized linear regression using dynamics described by equation (3.5).

3.4.3 ℓ_2-norm Regularized Linear Regression.

In presence of ℓ_2-norm regularization, the over-parameterized linear regression problem becomes strongly convex and possesses a unique global optima. Proposition 2 serves a sanity check; where we show the convergence to this unique optima for any positive definite pre-conditioner matrix $\boldsymbol{D}(t)$ in the spectral domain.

Proposition 3.4.3. *Consider the over-parameterized linear regression setting with data matrix $\boldsymbol{X}$, noise $\boldsymbol{\zeta}$, and regularizer $\lambda > 0$. Assume the recursion $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}_t \nabla f(\boldsymbol{w}(t))$. Suppose for all $t \geq 0$, the pre-conditioner matrix $\boldsymbol{D}(t) \succ 0$, and the learning rate satisfies*

$$\eta \in \left(0, 2\left(\lambda_{\max}(\boldsymbol{D}(t))(\lambda_{\max}^2(\boldsymbol{X}) + \lambda)\right)^{-1}\right)$$

where $\lambda_{\max}(\cdot)$ indicates the maximum eigenvalue. Then, $\tilde{\boldsymbol{w}}(t)$ converges to the following fixed point

$$\lim_{t\to\infty} \tilde{\boldsymbol{w}}(t) = (\boldsymbol{\Lambda}^2 + \lambda \boldsymbol{I})^{-1}(\boldsymbol{\Lambda}^2 \tilde{\boldsymbol{w}}^\star + \boldsymbol{\Lambda}\boldsymbol{\zeta}). \tag{3.7}$$

Proposition 3.4.3 states that like the gradient descent based methods, the adaptive methods will perfectly capture the component of the generative $\boldsymbol{w}^*$ that lies in the subspace formed by the data matrix. In other words, with regularization the parameter vector converges in the span of $\boldsymbol{X}$.

In the proof of the proposition presented in the Appendix, we use contraction properties to show convergence where $\lambda > 0$ plays a significant role.

Further, as $\inf_t \operatorname{rank}(\boldsymbol{D}(t))$, a simple fixed-point analysis provides us with the in-span component and shows that for $\lambda > 0$ there is no out-of-span component of the solution. Note that this proposition acts as a proof of concept for the well-known result that adaptive methods and non-adaptive methods converge to the same solution in the presence of ℓ_2-norm regularization.

Lastly, different regularization techniques alter the implicit bias of the final converged solution differently. The claims made in this sub-section are only valid for ℓ_2-norm regularization.

3.4.4 Unregularized Linear Regression.

Next, we focus on the problem of unregularized linear regression in the over-parameterized regime. The optimization problem with squared loss is no longer strongly convex, and there are infinite solutions that can achieve zero training error. In this case, the convergence of unregularized linear regression depends on the initialization $\tilde{\boldsymbol{w}}(0)$. Further, as $\lambda_{\min}(\tilde{\boldsymbol{D}}(t)\boldsymbol{\Lambda}^2) = 0$ we cannot directly prove (using contraction mapping) convergence for general unregularized over-parameterized linear regression. However, when the pre-conditioner matrices satisfy a block matrix structure, then we can say something about the converged solution. Note that most of the popular adaptive algorithms [60, 61, 70, 76] satisfy the block matrix structure.

The out-of-span component behavior depends subtly on the interplay of the pre-conditioner matrices and the span of data. Now, we establish sufficient conditions on the class of pre-conditioner matrices for which the convergence is guaranteed (for more details refer Appendix).

We define some notations useful to state our main theorems. We use $\tilde{\boldsymbol{w}}_{(1)}(\infty) = \lim_{t\to\infty} \tilde{\boldsymbol{w}}_{(1)}(t)$ to denote the in-span component, and $\tilde{\boldsymbol{w}}_{(2)}(\infty) = \lim_{t\to\infty} \tilde{\boldsymbol{w}}_{(2)}(t)$ to denote the in-span component of the stationary point.[2] Let, $e_{(1)}(t) = \|\tilde{\boldsymbol{w}}_{(1)}(\infty) - \tilde{\boldsymbol{w}}_{(1)}(i)\|_2 = \mathcal{O}\left(\frac{1}{t^{\beta}}\right)$ be the ℓ_2-norm distance of in-span component of the iterate from the in-span stationary point at time $t \geq 1$. We further define:

Definition 3.4.4. *For a data matrix* $\mathbf{X}$ *and an adaptive method, with pre-conditioning matrices* $\{\mathbf{D}(t) : t \geq 1\}$*, we call the adaptive method* (α, β)*-converging on data, for any* $\alpha, \beta \geq 0$*, if and only if: i) the out-of-span component of the pre-condition matrices decays as* $|\lambda_{\max}(\tilde{\boldsymbol{D}}_2(t))| = \mathcal{O}\left(\frac{1}{t^{\alpha}}\right)$*; ii) the in in-span component of the iterates converges as,* $e_{(1)}(t) = \mathcal{O}\left(\frac{1}{t^{\beta}}\right)$ *(under Eq.* (3.5)*).*

Any adaptive method with a pre-conditioner matrix that lies entirely in the span of the matrix will have α set to ∞. Full-matrix Adagrad, GD, and Newton all fall under this class of adaptive methods. Popular adaptive methods, such as diagonalized Adagrad, RMSProp, and methods with a diagonal pre-conditioner matrix with non-zero entries, the convergence depends on the rate of decay of both the $\tilde{\boldsymbol{D}}_2(t)$ as well as the rate of decay of the error of the in-span component.

[2]Note that $\tilde{\boldsymbol{w}}_{(2)}(\infty) \in (\mathbb{R} \cup \{\infty\})^d$ for $i = 1, 2$, as we can not assume convergence of the iterates a pirori.

Theorem 3.4.5. *Consider the problem of over-parameterized linear regression with data matrix $\boldsymbol{X}$ and noise $\boldsymbol{\zeta}$ in the absence of regularization $\lambda = 0$. Assume the recursion $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}_t \nabla f(\boldsymbol{w}(t))$. If the preconditioner matrix $\boldsymbol{D}(t) \succ 0 \ \forall t \geq 0$, and $\eta \in \left(0, \frac{2}{\lambda_{\max}(\boldsymbol{D}(t))\lambda^2_{\max}(\boldsymbol{X})}\right)$, then in-span component of $\tilde{\boldsymbol{w}}(t)$ converges as follows*

$$\tilde{\boldsymbol{w}}_{(1)}(\infty) = (\tilde{\boldsymbol{w}}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}).$$

Furthermore, for an adaptive method (in Eq. (3.5)*) which is* (α, β)*-converging on data, if* $\alpha + \beta > 1$ *the out-of-span component converges to a stationary point that satisfies*

$$\|\tilde{\boldsymbol{w}}_{(2)}(\infty) - \tilde{\boldsymbol{w}}_{(2)}(0)\|_2 \leq \mathcal{O}\left(\|\tilde{\boldsymbol{w}}_{(1)}(0)\|_2 + \frac{1}{\alpha + \beta - 1}\right).$$

Remark on Theorem 3.4.5: Let us deconstruct the claims made in Theorem 3.4.5. Theorem 1 says that if η is set appropriately, then adaptive methods will perfectly fit the noisy training data. This is consistent with the claims in [44]. The convergence of out-of-span component depends on the decay rate of the pre-conditioner matrix $\tilde{\boldsymbol{D}}_2(t)$ as well as the decay rate of the error term in the in-span component $e_{(1)}(t) = \|(\boldsymbol{w}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i)\|_2$. For the simple case, when $\beta \geq 1$ and $\tilde{\boldsymbol{D}}_{(2)}(t) = \mathbf{0}$ for all t, we have that the out-of-span component converges to $\tilde{\boldsymbol{w}}_{(2)}(0)$. Next, if $\lim_{T\to\infty} \sum_{t=0}^{T} \max|\lambda(\tilde{\boldsymbol{D}}_{(2)}(t))| < \infty$ and $\alpha + \beta \geq 1$, then the out-of-span component converge. For all other cases, it is difficult to comment whether the out-of-span component will converge or diverge. Specifically, for $\alpha + \beta < 1$, we may not have divergence as the pre-conditioner matrices may align cancel the cumulative errors.

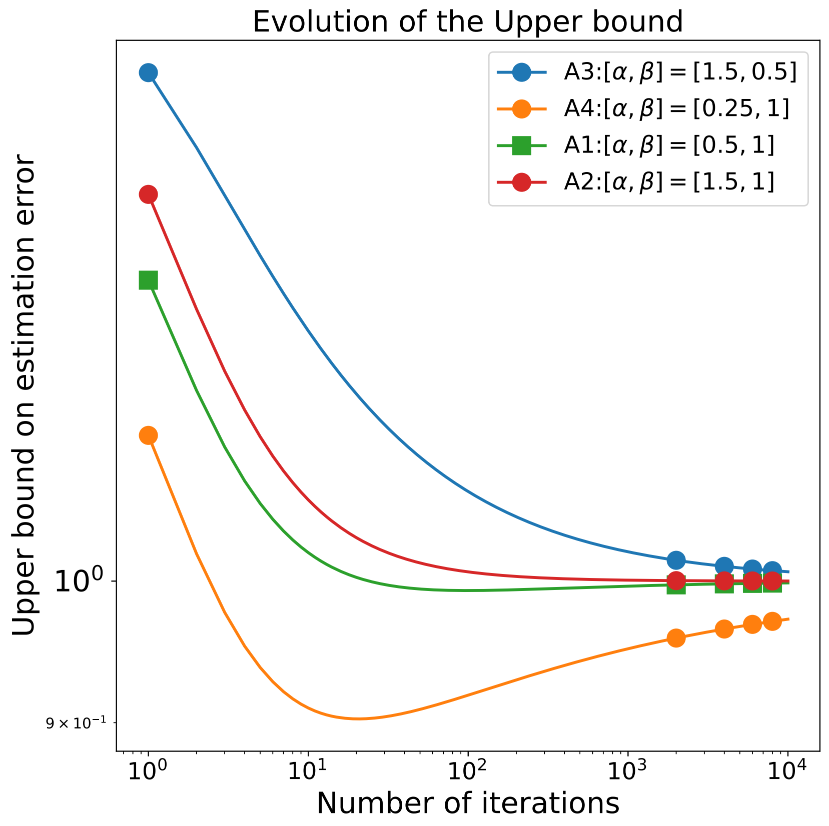


Figure 3.1: Evolution of upper bound dynamics for adaptive methods with different rates of (α, β) convergence, with $a = 1$, $b = 0.7$, and $c = 0.1$.

Remark on Dynamics: It is also interesting to note that the error in the in-span component converges to 0 and the error in the out-of-span component increases and saturates to a particular value. Our derived upper bound on the training error at time T for an adaptive method that is (α, β) converging on data is given as :

$$a + \frac{b}{(T+1)^{\beta}} \left(1 - \frac{c}{\alpha + \beta - 1} \frac{1}{(T+1)^{\alpha - 1}} \right)$$

for appropriate constants $a, b, c > 0$. The dynamics is shown in the Figure 5.6. Depending on the values of the constants, α, and β, adaptive methods demonstrate variable convergence dynamics.

3.5 Experiments

In this section, we focus on replicating the theoretical claims made in the previous parts using synthetic experiments for over-parameterized linear regression. Next, we show that these observations can be extended to the deep learning setup as well. We empirically compare two classes of algorithms:

- Plain gradient descent algorithms, including the mini-batch stochastic gradient descent and the accelerated stochastic gradient descent, with constant momentum.
- Adaptive methods like AdaGrad [60], RMSProp [63], and Adam [61], and the AdaGrad variant. Adaptive methods can include anything with a time-varying pre-conditioner matrix.

$d = 50$		GD	AM1	AM2	AM3
	$\boldsymbol{D}(t)$	$\boldsymbol{I}$	$\bar{\boldsymbol{D}}_1(t)$	$\boldsymbol{D}_2(t)$	$\mathcal{P}_{\boldsymbol{X}}(\bar{\boldsymbol{D}}_1(t))$
	Training Error	$1.27 \cdot 10^{-28}$	$\mathbf{1.42 \cdot 10^{-29}}$	$8.64 \cdot 10^{-4}$	$8.64 \cdot 10^{-29}$
$n = 10$	Test Error	81.56	**76.94**	79.62	81.65
	$\lVert \boldsymbol{w} - \boldsymbol{w}^* \rVert$	9.08	**8.92**	9.03	9.08
	Training Error	$4.77 \cdot 10^{-5}$	$\mathbf{6.07 \cdot 10^{-7}}$	$3.31 \cdot 10^{-3}$	$8.64 \cdot 10^{-4}$
$n = 40$	Test Error	**18.62**	19.56	20.35	18.65
	$\lVert \boldsymbol{w} - \boldsymbol{w}^* \rVert$	**4.31**	4.37	4.51	**4.31**

Table 3.3: Illustrating the varying performances of adaptive methods for over-parameterized linear regression. The final values are the average of 5 runs. AM1: Diagonalized Adagrad, AM2: Adagrad (AM1) Variant (where we square the diagonal terms instead of taking the square root), AM3: Projected version of AM1 onto the span of X. For AM3, $\tilde{\boldsymbol{D}}_2(t) = 0,\ \forall\ t$ and consistent with Theorem 3.4.5 it converges to the same point as SGD. AM1 and AM2 satisfy the (α, β) convergence criterion leading to convergence to a different point and different generalization than SGD.

3.5.1 Linear Regression

In the first part, we consider a simple linear regression example generated where the elements of both $\boldsymbol{X}$ and $\boldsymbol{\zeta}$ are generated from $\mathcal{N}(0, 1)$ distribution in an i.i.d. manner. First, we revisit Table 3.1 and demonstrate the varying performances of different adaptive methods.

In the first set of experiments, we showed how adaptive methods converging to a different solution might lead to solutions farther from $\boldsymbol{w}_b^*$, i.e. with higher L2-norm . Thus, the pre-conditioner matrices satisfying $\boldsymbol{D}_{ij}(t) = 0, \boldsymbol{D}(t) \succ 0$ have different generalization than their gradient based counterparts. In this section, we empirically demonstrate that pre-conditioner matrices of the form: $\boldsymbol{D}_{ij}(t) = 0, \boldsymbol{D}(t) \succ 0$ can guarantee better generalization than gradient based methods depending on the problem in hand. As a direct consequence of this, we show that solutions with a minimum norm should not be used as a yardstick to guarantee good generalization.

Here, we show in Table 3.3 and Figure 3.2 that different adaptive methods can yield better performance in terms of test error.

3.5.1.1 Counter Example

Next, we consider a linear regression problem with binary outputs, which is a variant of the example proposed in [71]. In this section, we show that even in terms of test accuracy, adaptive algorithms can yield better generalization performance than SGD. This supports the claim made recently in [77] for non-adaptive methods; *the testing criterion can play a crucial role in determining the generalization performance.* We observe that this claim holds for adaptive methods as well.

Toy example illustrating how to achieve different generalization from gradient descent based methods: To prove otherwise, we could either find a completely different counterexample (since adaptive methods do not perform well in the one in [71]), or we find an alternative adaptive method that performs well in that counter-example.

Here, we follow the second path and alter the previous counterexample in [71] by slightly changing the problem setting: at first, we reduce the margin between the two classes; the case where we increase the margin is provided in the Appendix. We *empirically* show that gradient-descent methods fail

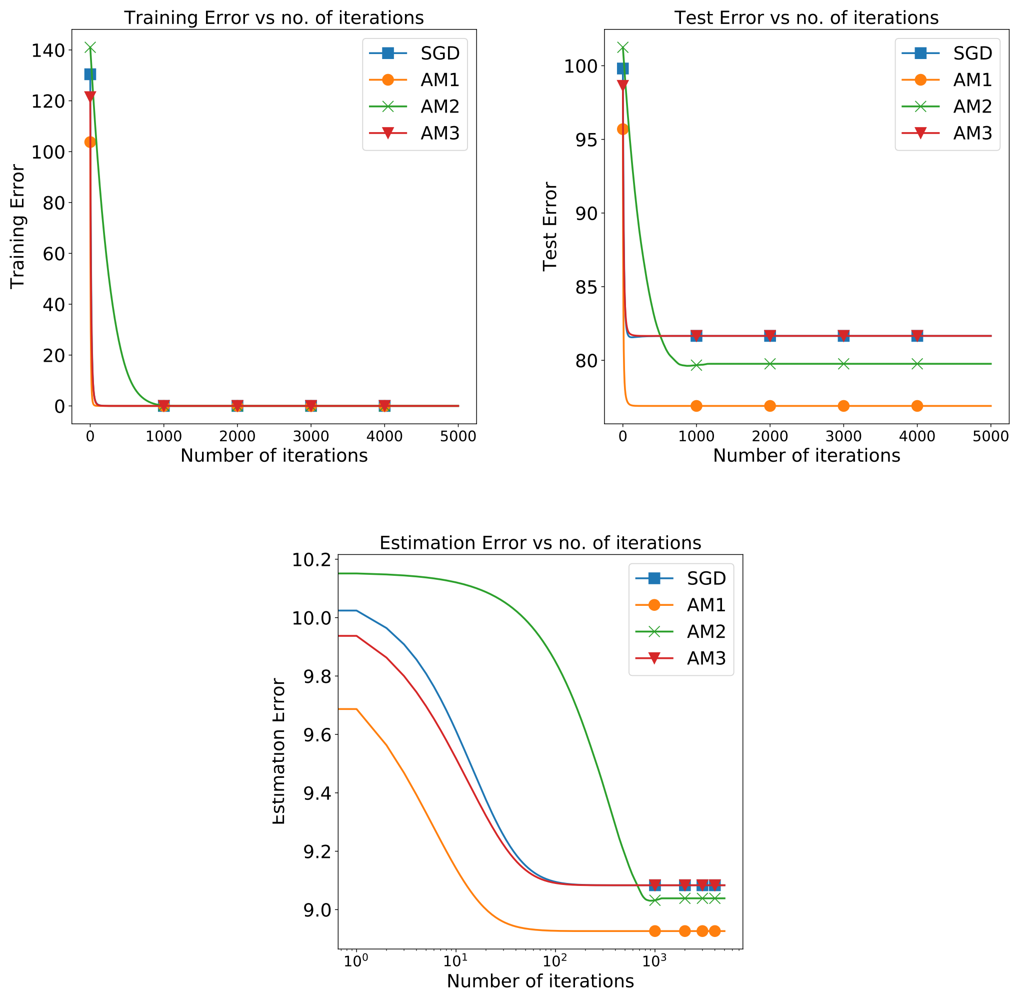


Figure 3.2: Synthetic example of over-parameterized linear regression where adaptive methods show better test error performance. Notice that adaptive method AM1 not only allows us to achieve faster convergence but also better generalization. Estimation error, $\|\|\boldsymbol{w}(t) - \boldsymbol{w}^*\|\|$ is in the semilog scale on the x axis (to highlight the double descent like phenomena in AM2 as predicted by the Remark at the end of Section 2). The reported results are the average of 5 runs with different initializations for a given realization of data.

to generalize as well as adaptive methods –with a slightly different D_k than AdaGrad. In particular, for the responses, we consider two classes $y_i \in \{\pm\ell\}$ for some $\ell \in (0,1)$; *i.e.*, we consider a smaller margin between the two classes. One can consider classes in $\{\pm 1\}$, but the rest of the problem settings need to be weighted accordingly. We selected to weight the classes differently in order not to drift much from the couterexample from [71]. ℓ can take different values, and still we get the same performance, as we show in the experiments below.

$$(x_i)_j = \begin{cases} y_i\ell, & j = 1, \\ 1, & j = 2,3, \\ 1, & j = 4+5(i-1), \\ 0, & \text{otherwise.} \end{cases} \qquad \text{if } y_i = 1,$$

$$(x_i)_j = \begin{cases} y_i\ell, & j = 1, \\ 1, & j = 2,3, \\ 1, & j = 4+5(i-1), \\ & \quad \cdots, 8+5(i-1), \\ 0, & \text{otherwise.} \end{cases} \qquad \text{if } y_i = -1. \tag{3.8}$$

Given this generative model, we construct n samples $\{y_i, x_i\}_{i=1}^n$, and set $d = 6n$, for different n values. We compare two simple algorithms: ***i)*** the plain gradient descent, for $\eta = \frac{1}{\lambda_1(\boldsymbol{X}^\top \boldsymbol{X})}$; ***ii)*** the recursion $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{D}(t)\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}(t) - \boldsymbol{y})$, where η is set as above, and $\boldsymbol{D}(t)$ follows the rule:

$$\boldsymbol{D}(t) = \texttt{diag}\left(1/\left(\sum_{j=t-J}^{t} \nabla f(\boldsymbol{w}(j)) \odot \nabla f(\boldsymbol{w}(j)) + \varepsilon\right)^2\right)$$
$$\succ 0, \qquad \text{for some } \varepsilon > 0, \text{ and } J < t \in \mathbb{N}_+ \tag{3.9}$$

Observe that $\boldsymbol{D}(t)$ uses the dot product of gradients, *squared.* A variant of this preconditioner is found in [78]; however our purpose is not to recommend a particular preconditioner but to show that *there are D_k that lead to better performance than the minimum norm solution.* We denote as w^{ada}, w^{adam} and w^{GD} the estimates of the ADAM, Adagrad Variant (ADAVAR) and simple gradient descent (GD), respectively.

The experiment obeys the following steps: ***i)*** we train both gradient and adaptive gradient methods on the same training set, ***ii)*** we test models on new data $\{y_i^{\text{test}}, x_i^{\text{test}}\}_{i=1}^{Q}$. We define performance in terms of the classification error: for a new sample $\{y_i^{\text{test}}, x_i^{\text{test}}\}$ and given w^{ada}, w^{adam} and w^{GD}, the only features that are non-zeros in both x_i^{test} and w's are the first 3 entries [71, pp. 5]. This is due to the fact that, for gradient descent and given the structure in $\boldsymbol{X}$, only these 3 features affects the performance of gradient descent. Thus, the decision rules for both algorithms are:

$$\begin{aligned}
\widehat{y_i}^{\text{ada}} &= \texttt{quant}_\ell \left(w_1^{\text{ada}} \cdot y_i^{\text{test}} + w_2^{\text{ada}} + w_3^{\text{ada}} \right), \\
\widehat{y_i}^{\text{GD}} &= \texttt{quant}_\ell \left(w_1^{\text{GD}} \cdot y_i^{\text{test}} + w_2^{\text{GD}} + w_3^{\text{GD}} \right), \\
\widehat{y_i}^{\text{adam}} &= \texttt{quant}_\ell \left(w_1^{\text{adam}} \cdot y_i^{\text{test}} + w_2^{\text{adam}} + w_3^{\text{adam}} \right),
\end{aligned}$$

where $\texttt{quant}_\ell(\alpha)$ finds the nearest point w.r.t. $\{\pm\ell\}$. With this example, our aim is to show that adaptive methods lead to models that have better generalization than gradient descent.

Table 3.4 summarizes the empirical findings. In order to cover a wider range of settings, we consider $n = [10,\ 50,\ 100]$ and set $d = 6n$, as dictated by

[71]. We generate $\boldsymbol{X}$ as above, where instances in the positive class, $y_i \in +\ell$, are generated with probability $p = 7/8$; the cases where $p = 5/8$ and $p = 3/8$ are provided in the appendix, and also convey the same message as in Table 3.4. Further details on the experiments are provided in the Appendix.

		Algorithm	GD	ADAVAR	ADAM
$n = 10$	$\ell = 1/32$	Acc. (%)	63	**100**	91
		$\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_{\text{mn}}\|_2$	$1.015 \cdot 10^{-16}$	$4.6924 \cdot 10^{4}$	0.1007
	$\ell = 1/16$	Acc. (%)	53	**100**	87
		$\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_{\text{mn}}\|_2$	$1.7401 \cdot 10^{-16}$	$1.1504 \cdot 10^{3}$	0.0864
	$\ell = 1/8$	Acc. (%)	58	**99**	84
		$\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_{\text{mn}}\|_2$	$4.08 \cdot 10^{-16}$	112.03	0.0764
$n = 50$	$\ell = 1/32$	Acc. (%)	77	**100**	88
		$\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_{\text{mn}}\|_2$	$4.729 \cdot 10^{-15}$	$3.574 \cdot 10^{3}$	0.0271
	$\ell = 1/16$	Acc. (%)	80	**100**	89
		$\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_{\text{mn}}\|_2$	$6.9197 \cdot 10^{-15}$	$4.44 \cdot 10^{2}$	0.06281
	$\ell = 1/8$	Acc. (%)	91	**100**	89
		$\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_{\text{mn}}\|_2$	$9.7170 \cdot 10^{-15}$	54.93	0.1767

Table 3.4: Prediction accuracy and distances from the minimum norm solution for plain gradient descent and adaptive gradient descent methods. We set $p = 7/8$ and $J = 10$, as in the main text. The adaptive method uses D_k according to (3.9). The distances shown are median values out of 100 different realizations for each setting; the accuracies are obtained by testing 10^4 predictions on unseen data.

The proposed AdaGrad variant described in equation (3.9) falls under the broad class of adaptive algorithms with D_k. However, for the counter example in the AdaGrad variant neither satisfies the convergence guarantees of Lemma 3.1 in [71, pp. 5], nor does it converge to the minimum norm solution evidenced by its norm in Table 3.4.

Proposition 3.5.1. *Suppose $\boldsymbol{X}^\top \boldsymbol{y}$ has no zero components. Define $\boldsymbol{D} = \boldsymbol{diag}(|\boldsymbol{X}^\top \boldsymbol{y}|^3)$ and assume there exists a scalar c such that $\boldsymbol{X}\boldsymbol{D}^{-1}\boldsymbol{sign}(\boldsymbol{X}^\top \boldsymbol{y}) =$*

$c\boldsymbol{y}$. Then, when initialized at 0, the AdaGrad variant in (3.9) *converges to the unique solution $\boldsymbol{w} \propto \boldsymbol{D}^{-1}\boldsymbol{sign}(\boldsymbol{X}^\top \boldsymbol{y})$.*

To buttress our claim that the AdaGrad variant in (3.9) converges to a solution different than that of minimum norm (which is the case for plain gradient descent), we provide the following proposition for a specific class of problems[3]; the proof is provided in Appendix Section B.10.

3.5.2 Deep Learning

Next, we observe that the theoretical claims made for the generalization of adaptive methods for over-parameterized linear regression extend over to over-parameterized neural networks. We perform extensive experiments on CIFAR-100 for CIFAR-100 datasets we explore four different architectures; PreActResNet18 [5], MobileNet [6], MobileNetV2 [7], GoogleNet [7]. Computing weight norms of all layers is memory intensive and computationally expensive typically, we use the weight vector norms on the last layer. We observe that, like linear regression, here also adaptive methods can be clubbed into different categories based on the evolution of their pre-conditioner matrices, $\boldsymbol{D}(t)$. For deep learning experiments, we do not have a generative parameter. We used the norm of the weights of the last layer as a surrogate for estimation error to comment on the convergence of optimization algorithms. It is evident that algorithms that have similar weight norms have similar training loss performance; however the other side of the claim need not be true.

[3]Not the problem proposed in the counter-example 1 on pg 5.

We empirically compare two classes of algorithms:

- Plain gradient descent algorithms, including the mini-batch stochastic gradient descent and the accelerated stochastic gradient descent, with constant momentum.
- Adaptive methods like AdaGrad [60], RMSProp [63], and Adam [61], and the AdaGrad variant. Adaptive methods can include anything with a varying pre-conditioning matrix.

In this section, we will extend the experiments to over-parameterized and under-parameterized neural networks without regularization. We begin with a detailed description of the datasets and the architectures we use along with comprehensive set of experiments with hyperparameter tuning.

Name	Network type	Dataset
M1-UP	Shallow CNN + FFN	MNIST
M1-OP	Shallow CNN + FFN	MNIST
C1-UP	Shallow CNN + FFN	CIFAR-10
C1-OP	ResNet18	CIFAR-10
C2-OP	PreActResNet18	CIFAR-100
C3-OP	MobileNet	CIFAR-100
C4-OP	MobileNetV2	CIFAR-100
C5-OP	GoogleNet	CIFAR-100

Table 3.5: Summary of the datasets and the architectures used for experiments. CNN stands for convolutional neural network, FF stands for feed forward network. More details are given in the main text.

MNIST dataset and the M1 architecture. Each experiment for M1 is simulated over 50 epochs and 10 runs for both under- and over-parameterized settings. The MNIST architectures consisted of two convolutional layers (the second one with dropouts [79]) followed by two fully connected layers. M1-OP had $\sim$ 73K parameters. Top row corresponds row to the M1-OP case. We plot training errors and the test accuracy results on unseen data. For the M1-OP case, SGD momentum performs less favorably compared to plain SGD, and we conjecture that this is due to non-optimal tuning. In this case, all adaptive methods perform similarly to SGD.

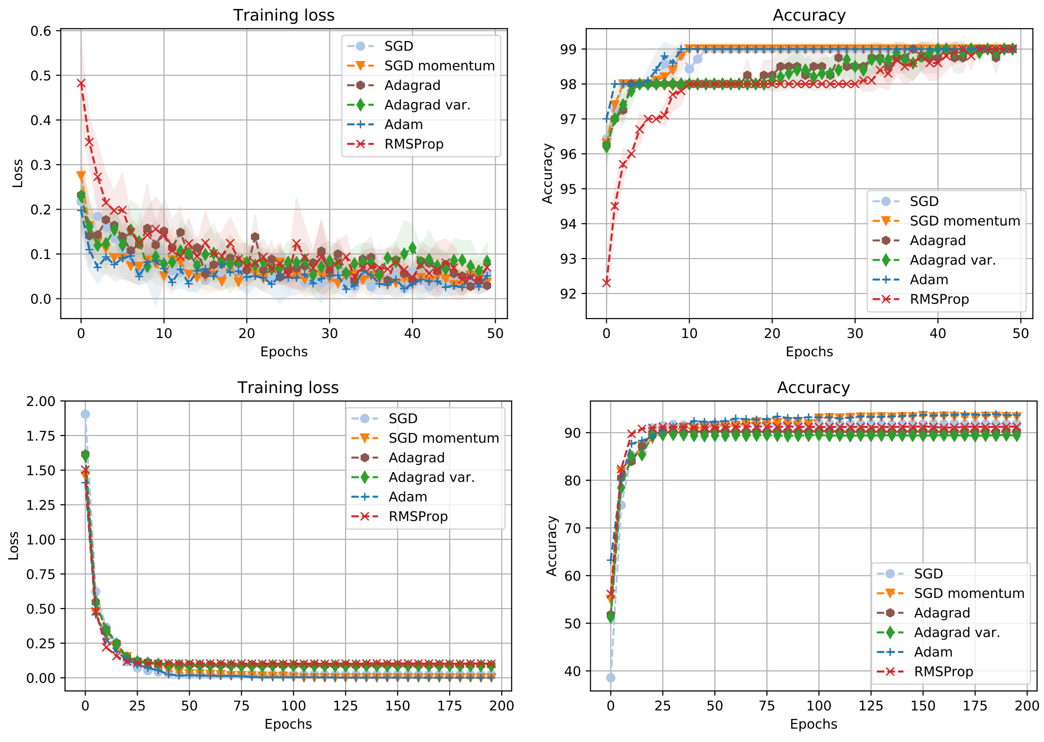


Figure 3.3: Accuracy results on unseen data, for different NN architectures and datasets for over-parameterized configurations. *Left two panels:* Accuracy and training loss for MNIST; *Right two panels:* Accuracy and training loss for CIFAR10.

CIFAR10 dataset and the C1 architecture. The over-parameterized CIFAR-10 setting, C1-OP was trained over 200 epochs. Here, we implement a Resnet [80] + dropout architecture (~ 0.25 million parameters) [4] and obtained top-1 accuracy of $\sim 93\%$. Adam and RMSProp achieves the best performance than their non-adaptive counterparts for both the under-parameterized and over-parameterized settings.

Figure 3.3, right panel, follows the same pattern with the MNIST data; it reports the results over 10 Monte-Carlo realizations. Again, we observe that AdaGrad methods do not perform as well as the rest of the algorithms. Nevertheless, adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants. Further experiments on CIFAR-100 for different architecture are provided in the Appendix.

CIFAR100 and other deep architectures (C{2-5}-OP). In this experiment, we focus only on the over-parameterized case: DNNs are usually designed over-parameterized in practice, with ever growing number of layers, and, eventually, a larger number of parameters [81]. We again completed 10 runs for each of the set up we considered. C2-OP corresponds to PreActResNet18 from [5], C3-OP corresponds to MobileNet from [6], C4-OP is MobileNetV2 from [7], and C5-OP is GoogleNet from [82]. The results are depicted in Figure 3.4. After a similar hyper-parameter tuning phase, we se-

[4]Some parts of the code are from the following github repository was used for experiments: https://github.com/kuangliu/pytorch-cifar

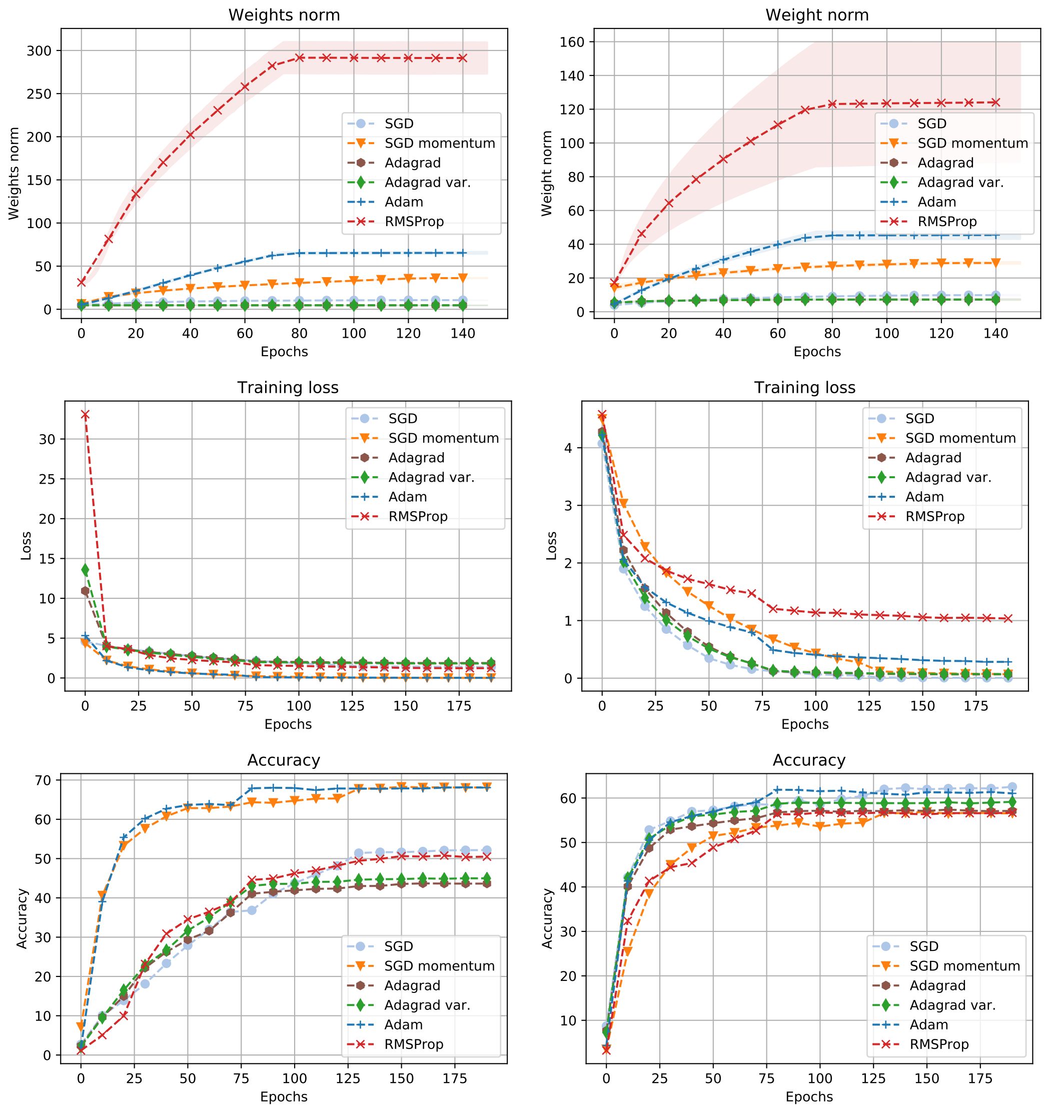


Figure 3.4: Accuracy results on unseen data, for different NN architectures on CIFAR100. *Left panel:* Accuracy and training loss for PreActResNet18 in [5]; *Right panel:* Accuracy and training loss for MobileNet in [6] *Top row:* Weight vectors of the last layer, *Middle row:* Training Loss, *Last row:* Test Accuracy.

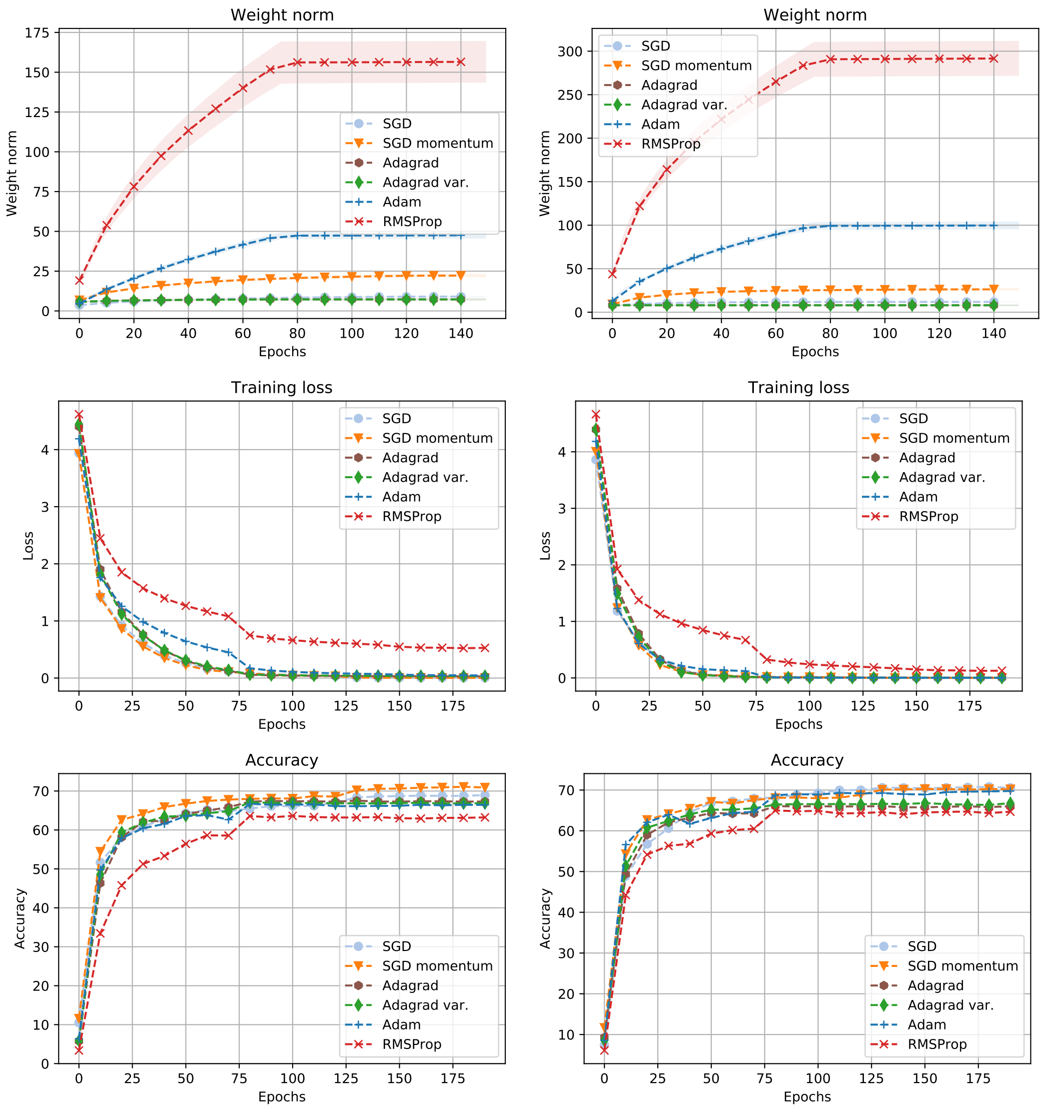


Figure 3.5: Accuracy results on unseen data, for different NN architectures on CIFAR100. *Left:* Accuracy and training loss for MobileNetV2 in [7], *Right panel:* Accuracy and training loss for GoogleNet in [7]. *Top row:* Weight vectors of the last layer, *Middle row:* Training Loss, *Last row:* Test Accuracy.

lected the best choices among the parameters tested. The results show no clear winner once again, which overall support our claims: *the superiority depends on the problem/data at hand; also, all algorithms require fine tuning to achieve their best performance.* We note that a more comprehensive reasoning requires multiple runs for each network, as other hyper-parameters (such as initialization) might play significant role in closing the gap between different algorithms.

An important observation of Figure 3.4 comes from the bottom row of the panel. There, we plot the Euclidean norm $\|\cdot\|_2$ of all the trainable parameters of the corresponding neural network. While such a norm could be considered arbitrary (e.g., someone could argue other types of norms to make more sense, like the spectral norm of layer), we use the Euclidean norm as ***i)*** it follows the narrative of algorithms in linear regression, where plain gradient descent algorithms choose minimum ℓ_2-norm solutions, and ***ii)*** there is recent work that purposely regularizes training algorithms towards minimum norm solutions [83].

Our findings support our claims: in particular, for the case of MobileNet and MobileNetV2, Adam, an adaptive method, converges to a solution that has at least as good generalization as plain gradient methods, while having $2\times$ larger ℓ_2-norm weights. However, this may not always be the trend: in Figure 3.4, left panel, the plain gradient descent models for the PreActResNet18 architecture [5] show slightly better performance, while preserving low weight norm. The same holds also for the case of GoogleNet; see Figure 3.5, right panel.

3.5.2.1 Hyperparameter tuning

Both for adaptive and non-adaptive methods, the step size and momentum parameters are key for favorable performance, as also concluded in [71]. Default values were used for the remaining parameters. The step size was tuned over an exponentially-spaced set $\{0.0001, 0.001, 0.01, 0.1, 1\}$, while the momentum parameter was tuned over the values of $\{0, 0.1, 0.25, 0.5, 0.75, 0.9\}$. We observed that step sizes and momentum values smaller/bigger than these sets gave worse results. *Yet, we note that a better step size could be found between the values of the exponentially-spaced set.* The decay models were similar to the ones used in [71]: no decay and fixed decay. We used fixed decay in the over-parameterized cases, using the `StepLR` implementation in `pytorch`. We experimented with both the decay rate and the decay step in order to ensure fair comparisons with results in [71].

3.5.2.2 Results

Our main observation is that in over-parameterized cases adaptive and non-adaptive methods converge to solutions with similar testing accuracy: the superiority of simple or adaptive methods depends on the problem/data at hand. Further, as already pointed in [71], adaptive methods often require similar parameter tuning. Most of the experiments involve using readily available code from GitHub repositories. Since increasing/decreasing batch-size affects the convergence [84], all the experiments were simulated on identical batch-sizes. Finally, our goal is to show performance results in the purest algorithmic setups: often, our tests did not achieve state of the art performance.

Overall, despite not necessarily converging to the same solution as gradient descent, adaptive methods generalize as well as their non-adaptive counterparts. We compute standard deviations from all Monte Carlo instances, and plot them with the learning curves (shown in shaded colors is the one-apart standard deviation plots; best illustrated in electronic form). For the cases of C{1-5}-OP, we also show the weight norms of the solutions (as in Euclidean distance $\|\cdot\|_2$ of all the trainable weights in the network). Such measure has been in used in practice [83], as a regularization to find minimum Euclidean norm solutions, inspired by the results from support vector machines [45].

We observe that adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants, supporting our conjecture that each algorithm requires a different configuration, but still can converge to a good local point; also that adaptive methods require the same (if not more) tuning. Again, we observe that AdaGrad methods do not perform as well as the rest of the algorithms. Nevertheless, adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants. Further experiments on CIFAR-100 for different architecture are provided in the Appendix.

This result, combined with our experiments, indicate that the minimum norm solution does not guarantee better generalization performance for over-parameterized settings, even in cases of linear regression. Thus, it is unclear why that should be the case for deep neural networks. A detailed analysis about the class of counter-examples is available in Appendix Section B.11.

Details about extensive hyperparameter tuning, dataset descriptions, practical issues in implementation, and more experiments, are available in the Appendix.

3.6 Conclusions and Future Work

In this paper, we consider two class of methods described: non-adaptive methods (Eq. (3.1)) and adaptive methods (Eq. (3.2)). Switching to a spectral domain allows us to divide adaptive methods into two further categories based on if they will have the same generalization as SGD or not (assuming the same initialization point). We obtain that the convergence of adaptive methods completely depends on the structure of pre-conditioner matrices $\boldsymbol{D}(t)$ along with the initialization point and the given data. Our theoretical analysis allows us to obtain useful insights into the convergence of adaptive methods, which can be useful while designing new adaptive methods. If the aim while designing an adaptive method is faster convergence and similar generalization as SGD, then it is important to ensure that the pre-conditioner matrix lies in the span of the data matrix $\boldsymbol{D}(t) = \mathcal{P}_{\boldsymbol{X}}(\cdot)$. Examples of such $\boldsymbol{D}(t)$ include $\{\mathbb{I}, (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\}$. However, if the aim is to hope for a different generalization than SGD (if SGD gets stuck on specific bad minima), then it is essential to ensure that the conditions in Theorem 3.4.5 are satisfied to ensure that $\tilde{\boldsymbol{w}}(t)$ converges to a different solution. Our experimental results on over-parameterized settings for both linear regression and deep learning back our theoretical claims. We also note the small superiority of non-adaptive methods on a few DNN simulations is not fully understood and needs further investigation, beyond the simple linear regression model. What was clear though from our experiments is that adaptive methods may converge to a model that has better generalization properties, where the $\ell 2$-norm of the weights is more substantial, but often require no less fine-tuning than non-adaptive methods.

A preliminary analysis of regularization for over-parameterized linear regression reveals that it can act as an equalizer over the set of adaptive and non-adaptive optimization methods, i.e., force all optimizers to converge to the same solution. However, more work is needed to analyze its effect on the overall generalization guarantees both theoretically and experimentally as compared to the non-regularized versions of these algorithms.

Chapter 4

Towards Improving the Robustness of SGD

This paper focuses on machine learning problems that can be formulated as optimizing the sum of n convex loss functions:

$$\min_{\boldsymbol{w}} \; F(\boldsymbol{w}) \text{ where } F(\boldsymbol{w}) \; = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}) \tag{4.1}$$

Stochastic gradient descent (SGD) is a popular way to solve such problems when n is large; the simplest SGD update is:

$$\text{SGD: } \boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \eta_t \nabla f_{i_t}(\boldsymbol{w_t}) \tag{4.2}$$

where the sample i_t is typically chosen uniformly at random from [n].

However, as is well known, the performance of SGD and most other stochastic optimization methods is highly sensitive to the quality of the available training data. A small fraction of outliers can cause SGD to converge far away from the true optimum. While there has been a significant amount of work on more robust algorithms for special problem classes (e.g. linear regression, PCA etc.) in this paper our objective is to make a modification to the

Parts of this chapter are available at: Shah, V., Wu, X. & Sanghavi, S.. (2020). Choosing the Sample with Lowest Loss makes SGD Robust. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, in PMLR 108:2120-2130 [85]. The author was a part of formulating the problem, analyzing the theory, designing the experiments and writing up the results presented in the paper.

basic SGD method itself; one that can be easily applied to the many settings where vanilla SGD is already used in the training of machine learning models.

We call our method Min-k Loss SGD (MKL-SGD), given below. In each iteration, we first choose a set of k samples and then select the sample with the smallest current loss in that set; this sample is then used for the update step.

Algorithm 2 MKL-SGD

```
1: Initialize $\boldsymbol{w_0}$
2: Given samples $D = (\boldsymbol{x_t}, y_t)_{t=1}^{\infty}$
3: for $t = 1, \ldots$ do
4:     Choose a set $S_t$ of $k$ samples
5:     Select $i_t = \arg\min_{i \in S_t} f_i(\boldsymbol{w_t})$
6:     Update $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \eta \nabla f_{i_t}(\boldsymbol{w_t})$
7: end for
8: Return $\boldsymbol{w_t}$
```

The effectiveness of our algorithm relies on a simple observation: *in a situation where most samples adhere to a model but a few are outliers skewing the output, the outlier points that contribute the most to the skew are often those with high loss.* In this paper, our focus is on the stochastic setting for standard convex functions. We show that it provides a certain degree of robustness against outliers/bad training samples that may otherwise skew the estimate.

Our Contributions

- To keep the analysis simple yet insightful, we define three natural and *deterministic* problem settings - noiseless with no outliers, noiseless with outliers, and noisy with outliers - in which we study the performance of MKL-SGD . In all of these settings the individual losses are assumed to be convex, and the overall loss is additionally strongly convex. We are interested in finding the optimum $\boldsymbol{w}^*$ of the "good" samples, but we do not a-priori know which samples are good and which are outliers.

- The expected MKL-SGD update (over the randomness of sample choice) is *not* the gradient of the original loss function (as would have been the case with vanilla SGD); it is instead the gradient of a different non-convex surrogate loss, even for the simplest and friendliest setting of noiseless with no outliers. Our first result establishes that this non-convexity however does not yield any bad local minima or fixed points for MKL-SGD in this particular setting, ensuring its success.

- We next turn to the setting of noiseless with outliers, where the surrogate loss can now potentially have many spurious local minima. We show that by picking a value of k high enough (depending on a condition number of the loss functions that we define) the local minima of MKL-SGD closest to $\boldsymbol{w}^*$ is better than the (unique) fixed point of SGD.

- We establish the convergence rates of MKL-SGD - with and without outliers - for both the noiseless and noisy settings .

- We back up our theoretical results with both synthetic linear regression experiments that provide insight, as well as encouraging results on the

MNIST and CIFAR-10 datasets.

4.1 Related Work

The related work can be divided into the following four main subparts:

Stochastic optimization and weighted sampling The proposed MKL-SGD algorithm inherently implements a weighted sampling strategy to pick samples. Weighted sampling is one of the popular variants of SGD that can be used for matching one distribution to another (importance sampling), improving the rate of convergence, variance reduction or all of them and has been considered in [86, 87, 88, 89]. Other popular weighted sampling techniques include [90, 91, 92]. Without the assumption of strong convexity for each $f_i(.)$, the weighted sampling techniques often lead to biased estimators which are difficult to analyze. Another idea that is analogous to weighted sampling includes boosting [93] where harder samples are used to train subsequent classifiers. *However, in presence of outliers and label noise, learning the hard samples may often lead to over-fitting the solution to these bad samples.* This serves as a motivation for picking samples with the lowest loss in MKL-SGD .

Robust linear regression Learning with bad training samples is challenging and often intractable even for simple convex optimization problems. For example, OLS is quite susceptible to arbitrary corruptions by even a small fraction of outliers. Least Median Squares (LMS) and least trimmed squares (LTS) estimator proposed in [94, 95, 96] are both sample efficient, have a relatively high break-down point, but require exponential running time to converge. [97] provides a detailed survey on some of these robust estimators for OLS problem. Recently, [98, 99, 100] have proposed robust learning algorithms for linear regression which require the computation of gradient over the entire dataset. In this version, our focus is on stochastic optimization in presence of outliers.

Robust optimization Robust optimization has received a renewed impetus following the works in [101, 102, 103]. In most modern machine learning problems, however, simultaneous access to gradients over the entire dataset is time consuming and often, infeasible. [104, 105] provides robust meta-algorithms for stochastic optimization under adversarial corruptions. However, both these algorithms require the computation of one or many principal components per epoch which requires atleast $O(p^2)$ computation ([106]). In contrast, MKL-SGD algorithm runs in $O(k)$ computations per iteration where k is the number of loss evaluations per epoch. In this paper, we don't consider the adversarial model, our focus is on the simpler corruption model where we consider outliers as defined in the next section.

Label noise in deep learning [107, 108, 109] describe different techniques to learn in presence of label noise and outliers. [110] showed that deep neural networks are robust to random label noise especially for datasets like MNIST and CIFAR10. [111, 112] propose optimization methods based on re-weighting samples that often require significant pre-processing. In this paper, our aim is to propose a computationally inexpensive optimization approach that can also provide a degree of robustness.

4.2 Problem Setup

We make the following assumptions about our problem setting ((4.1)). Let $\mathbb{O}$ be the set of outlier samples; this set is of course unknown to the algorithm. We denote the optimum of the non-outlier samples by $\boldsymbol{w}^*$, i.e.

$$\boldsymbol{w}^* \;:=\; \arg\min_{\boldsymbol{w}} \sum_{i \notin \mathbb{O}} f_i(\boldsymbol{w})$$

In this paper we show that MKL-SGD allows us to estimate $\boldsymbol{w}^*$ without a-priori knowledge of the set $\mathbb{O}$, under certain conditions. We now spell these conditions out.

Assumption 5 (Individual losses). Each $f_i(\boldsymbol{w})$ is convex in $\boldsymbol{w}$, with Lipschitz continuous gradients with constant L_i.

$$\|\nabla f_i(\boldsymbol{w_1}) - \nabla f_i(\boldsymbol{w_2})\| \leq L_i \|\boldsymbol{w_1} - \boldsymbol{w_2}\|$$

and define $L := max_i L_i$

It is common to also assume strong convexity of the overall loss function $F(\cdot)$. Here, since we are dropping samples, we need a very slightly stronger assumption.

Assumption 6 (**Overall loss**)**.** For any $n-k$ size subset S of the samples, we assume the loss function $\sum_{i \in S} f_i(\boldsymbol{w})$ is *strongly convex* in $\boldsymbol{w}$. Recall that here k is the size of the sample set in the MKL-SGD algorithm.

Lastly, we also assume that all the functions share the same minimum value. Assumption 3 is often satisfied by most standard loss functions with a finite unique minima [66] such as squared loss, hinge loss, etc.

Assumption 7 (**Equal minimum values**)**.** Each of the functions $f_i(.)$ shares the same minimum value $\min_{\boldsymbol{w}} f_i(\boldsymbol{w}) = \min_{\boldsymbol{w}} f_j(\boldsymbol{w}) \ \forall \ i, j$.

We are now in a position to formally define three problem settings we will consider in this paper. For each i let $C_i := \{\hat{\boldsymbol{w}} : \hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} f_i(\boldsymbol{w})\}$ denote the set of optimal solutions (there can be more than one because $f_i(\cdot)$ is only convex but not strongly convex). Let $d(a, S)$ denote the shortest distance between point a and set S.

Noiseless setting with no outliers: As a first step and sanity check, we consider what happens in the easiest case: where there are no outliers. There is also no "noise", by which we mean that the optimum $\boldsymbol{w}^*$ we seek is also in the optimal set of every one of the individual sample losses, i.e.

$$\boldsymbol{w}^* \in C_i \text{ for all } i$$

Of course in this case vanilla SGD (and many other methods) will converge to $\boldsymbol{w}^*$ as well; we just study this setting as a first step and also to build insight.

Noiseless setting with outliers: Finally, we consider the case where a subset $\mathbb{O}$ of the samples are outliers. Specifically, we assume that for outlier samples the $\boldsymbol{w}^*$ we seek lies far from their optimal sets, while for the others it is in the optimal sets:

$$d(\boldsymbol{w}^*, C_i) \geq 2\delta \text{ for all } i \in \mathbb{O}$$

$$\boldsymbol{w}^* \in C_i \text{ for all } i \notin \mathbb{O}$$

Note that now vanilla SGD on the entire loss function will *not* converge to $\boldsymbol{w}^*$.

Noisy setting: As a second step, we consider the case when samples are noisy but there are no outliers. In particular, we model noise by allowing $\boldsymbol{w}^*$ to now be outside of individual optimal sets C_i, but not too far; specifically,

No outliers

$$d(\boldsymbol{w}^*, C_i) \leq \delta \text{ for all } i$$

With outliers

$$d(\boldsymbol{w}^*, C_i) \leq \delta \text{ for all } i \in \mathbb{O}$$

$$d(\boldsymbol{w}^*, C_i) > 2\delta \text{ for all } i \notin \mathbb{O}$$

For the noisy setting, we will focus only on the convergence guarantees. We will show that MKL-SGD gets close to $\boldsymbol{w}^*$ in this setting; again in this case vanilla SGD will do so as well for the no outliers setting of course.

4.3 Understanding MKL-SGD

We now build some intuition for MKL-SGD by building some simple notation and looking at some simple settings. Recall MKL-SGD takes k samples and then retains the one with lowest current loss; this means it is sampling non-uniformly. For any $\boldsymbol{w}$, let $m_1(\boldsymbol{w}), m_2(\boldsymbol{w}), m_3(\boldsymbol{w}), \dots m_n(\boldsymbol{w})$ be the sorted order w.r.t. the loss at that $\boldsymbol{w}$, i.e.

$$f_{m_1(\boldsymbol{w})}(\boldsymbol{w}) \leq f_{m_2(\boldsymbol{w})}(\boldsymbol{w}) \leq \cdots \leq f_{m_n(\boldsymbol{w})}(\boldsymbol{w})$$

Recall that for a sample to be the one picked by MKL-SGD for updating $\boldsymbol{w}$, it needs to first be part of the set of k samples, and then have the lowest loss among them. A simple calculation shows that probability that the i^{th} best sample $m_i(\boldsymbol{w})$ is the one picked by MKL-SGD is given by

$$p_{m_i(\boldsymbol{w})}(\boldsymbol{w}) = \begin{cases} \dfrac{\binom{n-i}{k-1}}{\binom{n}{k}} & \text{without replacement} \\ \dfrac{(n-(i-1))^k - (n-i)^k}{n^k} & \text{with replacement} \end{cases} \tag{4.3}$$

In the rest of the paper, we will focus on the "with replacement" scenario for ease of presentation; this choice does not change our main ideas or results. With this notation, we can rewrite the expected update step of MKL-SGD as

$$\mathbb{E}[\boldsymbol{w}_+|\boldsymbol{w}] = \boldsymbol{w} - \eta \sum_i p_{m_i(\boldsymbol{w})} \nabla f_{m_i(\boldsymbol{w})}(\boldsymbol{w})$$

For simplicity of notation in the rest of the paper, we will relabel the update term in the above by defining as follows:

$$\nabla \widetilde{F}(\boldsymbol{w}) \ := \ \sum_i p_{m_i(\boldsymbol{w})} \nabla f_{m_i(\boldsymbol{w})}(\boldsymbol{w})$$

Underlying this notation is the idea that, in expectation, MKL-SGD is akin to gradient descent on a *surrogate* loss function $\tilde{F}(\cdot)$ which is different from the original loss function $F(\cdot)$; indeed if needed this surrogate loss can be found (upto a constant shift) from the above gradient. We will not do that explicitly here, but instead note that even with all our assumptions, indeed even without any outliers or noise, this surrogate loss can be non-convex. It is thus important to see that MKL-SGD does the right thing in all of our settings, which is what we build to now.

4.3.1 Noiseless setting with no outliers

As a first step (and for the purposes of sanity check), we look at MKL-SGD in the simplest setting when there are no outliers and no noise. Recall from above that this means that $\boldsymbol{w}^*$ is in the optimal set of every single individual loss $f_i(\cdot)$. However as mentioned above, even in this case the surrogate loss can be non-convex, as seen e.g. in Figure 4.1 for a simple example. However, in the following lemma we show that even though the overall surrogate loss $\tilde{F}(\cdot)$ is non-convex, in this no-noise no-outlier setting it has a special property with regards to the point $\boldsymbol{w}^*$.

Lemma 4.3.1. *In the noiseless setting, for any $\boldsymbol{w}$ there exists a $\lambda_{\boldsymbol{w}} > 0$ such that*

$$\nabla \widetilde{F}(\boldsymbol{w}) \top (\boldsymbol{w} - \boldsymbol{w}^*) \geq \lambda_{\boldsymbol{w}} \|\boldsymbol{w} - \boldsymbol{w}^*\|^2.$$

In words, what this lemma says is that on the line between any point $\boldsymbol{w}$ and the point $\boldsymbol{w}^*$, the surrogate loss function $\tilde{F}$ is convex from any point –

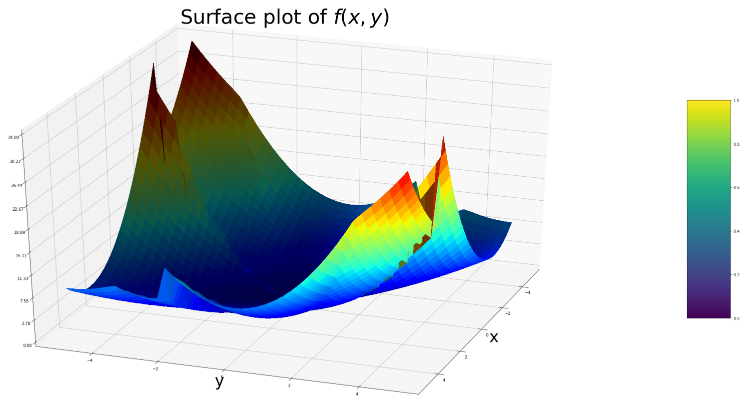


Figure 4.1: Non-convexity of the surface plot with three samples in the two-dimensional noiseless linear regression setting

even thought it is not convex overall. The following uses this to establish our first result: that in the noiseless setting with no outliers, $\boldsymbol{w}^*$ is the only fixed point (in expectation) of MKL-SGD .

Theorem 4.3.2 (**Unique stationary point**). *For the noiseless setting with no outliers, and under assumptions* $1-3$*, the expected MKL-SGD update satisfies* $\nabla\widetilde{F}(\boldsymbol{w}) = 0$ *if and only if* $\boldsymbol{w} = \boldsymbol{w}^*$.

4.3.2 Outlier setting

In presence of outliers, the surrogate loss can have multiple local minima that are far from $\boldsymbol{w}^*$ and indeed potentially even worse than what we could have obtained with vanilla SGD on the original loss function. We now analyze MKL-SGD in the simple setting of symmetric squared loss functions and try to gain useful insights into the landscape of loss function. We would like to point out that the analysis in the next part serves as a clean template and can be extended for many other standard loss functions used in convex optimization.

Squared loss in the vector setting The loss functions are redefined as follows:

$$f_i(\boldsymbol{w}) = \begin{cases} l_i\|\boldsymbol{w} - \boldsymbol{w}^*\|^2 & \forall\ i \notin \mathbb{O} \\ l_i\|\boldsymbol{w} - \boldsymbol{w}_{b_i}\|^2 & \forall\ i \in \mathbb{O}, \end{cases} \tag{4.4}$$

Without loss of generality, assume that $2\delta < \|\boldsymbol{w}_{b_1} - \boldsymbol{w}^*\| \leq \|\boldsymbol{w}_{b_2} - \boldsymbol{w}^*\| \leq \cdots \leq \|\boldsymbol{w}_{b_{|\mathbb{O}|}} - \boldsymbol{w}^*\|$ and $\gamma = \frac{2\delta}{\|\boldsymbol{w}_{b_{|\mathbb{O}|}} - \boldsymbol{w}^*\|}$. Let $\bar{\boldsymbol{w}}$ be any stationary attained by MKL-SGD . Suppose $\theta_{M,\bar{\boldsymbol{w}}}$ be the angle between the line passing through $\boldsymbol{w}_{b_M}$ and $\boldsymbol{w}^*$ and the line connecting $\bar{\boldsymbol{w}}$ and $\boldsymbol{w}^*$ and $\kappa = \frac{\max_{i\in[n]} l_i}{\min_{i\in[n]} l_i}$. Let $\hat{p}(\boldsymbol{w}_0) = \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_0)$ represent the total probability of picking outliers at the starting point $\boldsymbol{w}_0$. The maximum possible value that can be attained $\hat{p}$ is given as:

$$\hat{p}_{max} = \max_{w} \hat{p}(\boldsymbol{w}) = \sum_{i=1}^{|\mathbb{O}|} p_{m_i(\boldsymbol{w})}(\boldsymbol{w}) \tag{4.5}$$

where for any $\boldsymbol{w}$, $p_{m_i(\boldsymbol{w})}(\boldsymbol{w})$ are ordered i.e. $p_{m_1(\boldsymbol{w})}(\boldsymbol{w}) > p_{m_2(\boldsymbol{w})}(\boldsymbol{w}) > \cdots > p_{m_n(\boldsymbol{w})}(\boldsymbol{w})$.

At $\boldsymbol{w}^*$, by definition, we know that $\forall\ i \notin \mathbb{O}$, $f_i(\boldsymbol{w}^*) = 0$ and $\forall\ j \in \mathbb{O}$, $f_j(\boldsymbol{w}^*) > 0$. By continuity arguments, there exists a ball of radius $r > 0$ around $\boldsymbol{w}^*$, $\mathcal{B}_r(\boldsymbol{w}^*)$, defined as follows:

$$\mathcal{B}_r(\boldsymbol{w}^*) = \left\{ \boldsymbol{w} \mid \begin{array}{l} f_i(\boldsymbol{w}) < f_j(\boldsymbol{w})\ \forall\ i \notin \mathbb{O},\ j \in \mathbb{O}, \\ \|\boldsymbol{w} - \boldsymbol{w}^*\| \leq r \end{array} \right\} \tag{4.6}$$

In the subsequent lemma, we show that that it is possible to drift into the ball $\mathcal{B}_r(\boldsymbol{w}^*)$ where the clean samples have the highest probability or the lowest loss.

Lemma 4.3.3. *Consider the loss function and $\mathcal{B}_r(\boldsymbol{w}^*)$ as defined in equations* (4.4) *and* (4.6) *respectively. Suppose $q = \frac{\cos\theta_{M,\bar{\boldsymbol{w}}}}{\gamma} - 1 + \frac{\sqrt{\kappa}\cos\theta_{M,\bar{\boldsymbol{w}}}}{\gamma} > 0$ and $\hat{p}_{max}$ as defined in Equation* (4.5) *satisfies $\hat{p}_{max} \leq \frac{1}{1+\kappa q}$. Starting from any initialization $\boldsymbol{w}_0$, for any stationary point $\bar{\boldsymbol{w}}$ attained by MKL-SGD , we have that $\bar{\boldsymbol{w}} \in \mathcal{B}_r(\boldsymbol{w}^*)$.*

In other words, initializing at any point in the landscape, the final stationary point attained by MKL-SGD will inevitably assign the largest $n - |\mathbb{O}|$ probabilities to the clean samples.

Note that, the above lemma leads to a very strong worst-case guarantee. It states that the farthest optimum will always be within a bowl of distance r from $\boldsymbol{w}^*$ no matter where we initialize. However, when the necessary conditions for its convergence are violated, the guarantees are initialization dependent. Thus, all the discussions in the rest of this section will be with respect to these worst case guarantees. However, as we see in the experimental section for both neural networks and linear regression, random initialization also seems to perform better than SGD.

Effect of κ A direct result of Lemma 4.3.3 is that higher the condition number of the set of quadratic loss functions, lower is the fraction ϵ of outliers the MKL-SGD can tolerate. This is because large κ results in a small value of $\frac{1}{1+\kappa q}$. This implies that $\hat{p}$ has to be small which in turn requires smaller fractions fo corruptions, ϵ.

Effect of γ: The relative distance of the outliers from $\boldsymbol{w}^*$ plays a critical role in the condition for Lemma 4.3.3. We know that $\gamma \in (0, 1]$. $\gamma = 1$ implies the outliers are equidistant from the optimum $\boldsymbol{w}^*$. Low values of γ lead to a large q leading to the violation of the condition with $\hat{p}$ (since RHS in the condition is very small), which implies that one bad outlier can guarantee that the condition in Lemma 4.3.3 are violated. The guarantees in the above lemma are only when the outliers are not adversarially chosen to lie at very high relative distances from $\boldsymbol{w}^*$. One way to avoid the set of outliers far far away from the optimum is to have a filtering step at the start of the algorithm like the one in [104]. We will refer to this in Experiments.

Effect of $\cos\theta_{j,\bar{\boldsymbol{w}}}$: At first glance, it may seem that $\cos\theta_{j,\bar{\boldsymbol{w}}} = 0$ may cause $1 + \kappa q < 0$ and since $\hat{p}(\boldsymbol{w}) > 0$, the condition in Lemma 4.3.3 may never be satisfied. Since, the term $\cos\theta_{j,\bar{\boldsymbol{w}}}$ shows up in the denominator of the loss associated with outlier centered at $\boldsymbol{w}_{b_j}$. Thus, low values of $\cos\theta_{j,\bar{\boldsymbol{w}}}$ implies high value of loss associated with the function centered at $\boldsymbol{w}_{b_j}$ which in turn implies the maximum probability attained by that sample can never be in the top-$|\mathbb{O}|$ probabilities for that $\bar{\boldsymbol{w}}$.

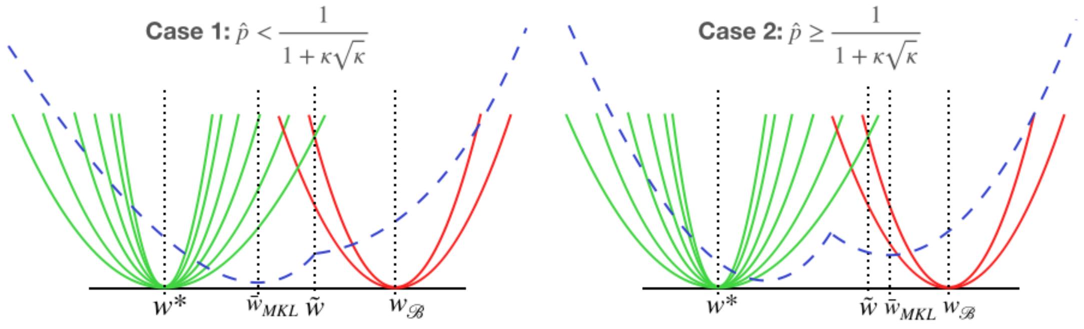


Figure 4.2: Illustration with conditions when bad local minima will or will not exist. Here, we demonstrate that even if we start at an initialization $\boldsymbol{w}_{\mathcal{B}}$ that assigns the highest probabilities to bad samples (red), it is possible to avoid the existence of a bad local minima if Condition 1 is satisfied. Recursively, we show in Lemma 4.3.3 that it is possible to avoid all bad local minima and reach a good local minima (where the good samples have the highest probabilities)

Visualizing Lemma 4.3.3: To illustrate the effect of Lemma 4.3.3 we will go to the scalar setting. For the scalar case, $d = 1$, we have $\theta_{j,\bar{\boldsymbol{w}}} = 0 \,\forall\, j$. Let us also assume for the sake of effective visualization that $\gamma = 1$, which in the scalar setting reduces to all the points being at the same distance from $\boldsymbol{w}^*$. Since we are concerned with the worst case analysis, we assume that all the outliers are centered on the same side of the optimum. The conditions in Lemma 2 are reduced to $q = \sqrt{\kappa} > 0$ (which is trivially true) and $\hat{p}_{max} \leq \dfrac{1}{1+\kappa\sqrt{\kappa}}$. A key takeaway from the above condition is that *for a fixed n as κ increases, we can tolerate smaller $\hat{p}$ and consequently smaller fraction of corruptions ϵ*. For a fixed ϵ and n, increasing the parameter k in MKL-SGD leads to an increase in $\hat{p}$ and thus increasing k can lead to the violation of the above condition. This happens because samples with lower loss will be picked with increasing probability as k increases.

Let $\bar{w}_{MKL}$ be the stationary point of MKL-SGD for this scalar case. If the above condition is satisfied, then the existence of the first bad MKL-SGD stationary point $\bar{w}_{MKL}$ can be avoided as illustrated in Figure 4.2.

To further elaborate on this, let us initialize MKL-SGD at $w_0 = w_B$, a point where the losses of outlier samples are 0 and all the clean samples have non-zero losses. If C1 holds true, then we are in Case 1 (Figure 4.2), the stationary point attained by MKL-SGD will be such that it is possible to avoid the existence of the first bad local minima which occurs when the top-$|\mathbb{O}|$ highest probabilities are assigned to the bad samples.

Not only that all other subsequent local minimas are avoided as well, until we reach the local minima which assigns the largest $(n-|\mathbb{O}|)$ probabilities to the clean samples[1]. This indicates that irrespective of where we initialize in the $1D$ landscape, we are bound to end up at a local minima with the highest probabilities assigned to the clean samples. In the latter part of this section, we will show that MKL-SGD solution attained when Case 1 holds is provably better than the SGD solution. However, if condition 1 is false (Case 2, Figure 4.2), then it is possible that MKL-SGD gets stuck at any one of the many local minimas that exist close to the outlier center w_B and we cannot say anything about the relative distance from $\boldsymbol{w}^*$.

[1]Refer to Appendix section **??** for further details on this discussion

Analysis for the general outlier setting: In this part, we analyze the fixed point equations associated with MKL-SGD and SGD and try to understand the behavior *in a ball $\mathcal{B}_r(\boldsymbol{w}^*)$ around the optimum?* For the sake of simplicity, we will assume that $\|\nabla f_i(\boldsymbol{w})\| \leq G \; \forall \; i \in \mathbb{O}$. Next, we analyze the following two quantities: i) distance of $\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}$ from $\boldsymbol{w}^*$ and distance of the any of the solutions attained by $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}$ from $\boldsymbol{w}^*$.

Lemma 4.3.4. *Let $\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}$ indicate the solution attained SGD. Under assumptions 1-3, there exists an ϵ' such that for all $\epsilon \leq \epsilon'$,*

$$\epsilon G \leq (1-\epsilon) L \|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\|$$

Using Lemma 4.3.1, we will define λ as follows:

$$\lambda := \min_{\boldsymbol{w}} \lambda_{\boldsymbol{w}} \tag{4.7}$$

Assumption 2 ensures that $\lambda > 0$, however the lower bounds for this λ are loss function dependent.

Lemma 4.3.5. *Let $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}$ be any first order stationary point attained by MKL-SGD . Under assumptions 1-3, for a given $\epsilon < 1$ and λ as defined in equation* (4.7)*, there exists a k' such that for all $k \geq k'$,*

$$\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \leq \frac{\epsilon^k G}{\lambda}$$

Finally, we show that any solution attained by MKL-SGD is provably better than the solution attained by SGD. We would like to emphasize that this is a very strong result. The MKL-SGD has numerous local minima and here we show that even the worst[2] solution attained by MKL-SGD is closer to $\boldsymbol{w}^*$ than the solution attained by SGD. Let us define $\alpha(\epsilon, L, k, \lambda) = \dfrac{(1-\epsilon)L\epsilon^{k-1}}{\lambda}$

Theorem 4.3.6. *Let $\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}$ and $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}$ be the the stationary points attained by SGD and MKL-SGD algorithms respectively for the noiseless setting with outliers. Under assumptions 1-3, for any $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} \in \mathcal{B}_r(\boldsymbol{w}^*)$ and λ defined in equation* (4.7)*, there exists an ϵ' and k' such that for all $\epsilon \leq \epsilon'$ and $k \geq k'$, we have $\alpha(\epsilon, L, k, \lambda) < 1$ and,*

$$\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| < \alpha(\epsilon, L, k, \lambda)\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\| \tag{4.8}$$

For squared loss in scalar setting, we claimed that for a fixed n and ϵ, using a large k may not be a good idea. Here, however once we are in the ball, $\mathcal{B}_r(\boldsymbol{w}^*)$, using larger k (any $k < \frac{n}{2}$), reduces $\alpha(\epsilon, L, k, \lambda)$ and allows MKL-SGD to get closer to $\boldsymbol{w}^*$.

The conditions required in Lemma 4.3.3 and Theorem 4.3.6 enable us to provide guarantees for only a subset of relatively well-conditioned problems. We would like to emphasize that the bounds we obtain are worst case bounds and not in expectation. As we will note in the Section 4.5 and the Appendix, however these bounds may not be necessary, for standard convex optimization problems MKL-SGD easily outperforms SGD.

[2]farthest solution from $\boldsymbol{w}^*$

4.4 Convergence Rates

In this section, we go back to the usual in expectation convergence analysis for the stochastic setting. For smooth functions with strong convexity, [91, 90] provided guarantees for linear rate of convergence. We restate the theorem here and show that the theorem still holds for the non-convex landscape obtained by MKL-SGD in noiseless setting.

Lemma 4.4.1 (**Linear Convergence** [90])**.** *Let $F(\boldsymbol{w}) = \mathbb{E}[f_i(\boldsymbol{w})]$ be λ-strongly convex. Set $\sigma^2 = \mathbb{E}[\|\nabla f_i(\boldsymbol{w^*})\|^2]$ with $\boldsymbol{w}^* := argmin F(\boldsymbol{w})$. Suppose $\eta \leq \frac{1}{\sup_i L_i}$. Let $\boldsymbol{\Delta}_t = \boldsymbol{w_t} - \boldsymbol{w^*}$. After T iterations, SGD satisfies:*

$$\mathbb{E}\left[\|\boldsymbol{\Delta}_T\|^2\right] \leq (1 - 2\eta\hat{C})^T \|\boldsymbol{\Delta}_0\|^2 + R_\sigma \tag{4.9}$$

where $\hat{C} = \lambda(1 - \eta \sup_i L_i)$ and $R_\sigma = \frac{\eta\sigma^2}{\hat{C}}$.

In the noiseless setting, we have $\|\nabla f_i(\boldsymbol{w^*})\| = 0$ and so $\sigma := 0$. $\boldsymbol{w}^*$ in (4.9) is the same as $\boldsymbol{w}^*$ stated in Theorem 4.3.2. Even though above theorem is for SGD, it still can be applied to our algorithm 2. At each iteration there exists a parameter $\lambda_{\boldsymbol{w_t}}$ that could be seen as the strong convexity parameter (c.f. Lemma 4.3.1). For MKL-SGD, the parameter λ in (4.9) should be $\lambda = \min_t \lambda_{\boldsymbol{w_t}}$. Thus, MKL-SGD algorithm still guarantees *linear convergence* result but with an implication of slower speed of convergence than standard SGD.

However, Lemma 4.4.1 will not hold for MKL-SGD in noisy setting since there exists no strong convexity parameter. Even for noiseless setting, the rate of convergence for MKL-SGD given in Lemma 4.4.1 is not tight.

The upper bound in (4.9) is loosely set to the constant $\lambda := \min_t \lambda_{\boldsymbol{w}_t}$ for all the iterations. We fix it by concretely looking at each iteration using the following notation. Denote the strong convexity parameter λ_{good} for all the good samples. Let

$$\psi = 2\eta_t \lambda_{good}(1 - \eta_t \sup_i L_i)\frac{n - |\mathbb{O}|}{\psi n}.$$

Next, we give a general bound for the any stochastic algorithm (c.f. Theorem 4.4.2) for both noiseless and noisy setting in absence and presence of outliers.

Theorem 4.4.2 (Distance to $\boldsymbol{w}^*$). *Let $\boldsymbol{\Delta}_t = \boldsymbol{w_t} - \boldsymbol{w}^*$. Suppose at t^{th} iteration, the stepsize is set as η_t, then conditioned on the current parameter $\boldsymbol{w}_t$, the expectation of the distance between the $\boldsymbol{w}_{t+1}$ and $\boldsymbol{w}^*$ can be upper bounded as:*

$$\mathbb{E}_i\left[\|\boldsymbol{\Delta}_{t+1}\|^2 | \boldsymbol{w_t}\right] \leq (1-\psi)\|\boldsymbol{\Delta}_t\|^2 + \eta_t R_t \tag{4.10}$$

where

$$\begin{aligned} R_t = & -\psi \sum_{i \notin \mathbb{O}} \frac{N p_i(\boldsymbol{w_t}) - 1}{N - |\mathbb{O}|} \langle \boldsymbol{\Delta}_t, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\rangle \\ & + 2\sum_{i \notin \mathbb{O}} p_i(\boldsymbol{w_t})\left(\eta_t \|\nabla f_i(\boldsymbol{w}^*)\|^2 - \langle \boldsymbol{\Delta}_t, \nabla f_i(\boldsymbol{w}^*)\rangle\right) \\ & + \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t})\left(\eta_t \|\nabla f_i(\boldsymbol{w_t})\|^2 + (f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t}))\right) \end{aligned}$$

Theorem 4.4.2 implies that for any stochastic algorithm in the both noisy and noiseless setting, outliers can make the upper bound (R_t) much worse as it produces an extra term (the third term in R_t). *The third term in R_t has a lower bound that is an increasing function of* $|\mathbb{O}|$. However, its impact can be reduced by appropriately setting $p_i(\boldsymbol{w_t})$, for instance using a larger k in MKL-SGD . In the appendix, we also provide a sufficient condition (Corollary 1 in the Appendix) when MKL-SGD is always better than standard SGD (in terms of its distance from $\boldsymbol{w}^*$ in expectation).

4.5 Experiments

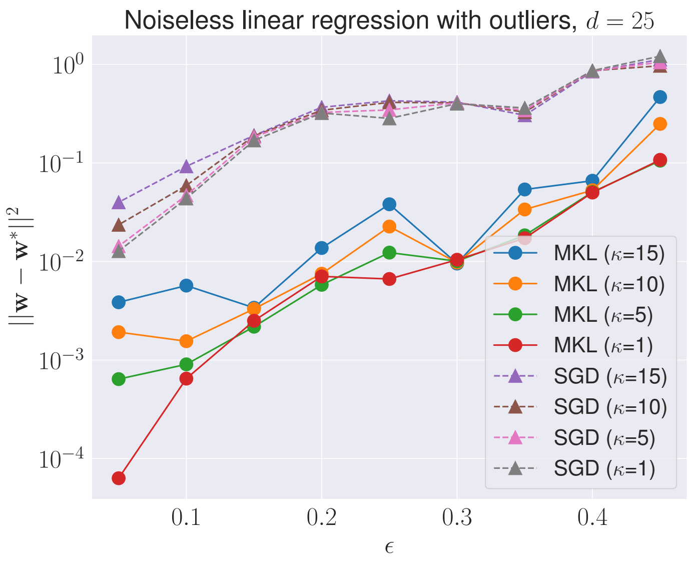
In this section, we compare the performance of MKL-SGD and SGD for synthetic datasets for linear regression and small-scale neural networks.

4.5.1 Linear Regression

For simple linear regression, we assume that X_i are sampled from normal distribution with different condition numbers. $X_i \sim \mathcal{N}(0, \boldsymbol{D})$ where $\boldsymbol{D}$ is a diagonal matrix such that $D_{11} = \kappa$ and $D_{ii} = 1$ for all i). We compare the performance of MKL-SGD and SGD for different values of κ (Fig. 4.3) under noiseless and noisy settings against varying levels of corruption ϵ. It is important to note that different κ values correspond to different rates of convergence. To ensure fair comparison, we run the algorithms till the error values stop decaying and take the distance of $\boldsymbol{w}^*$ from the exponential moving average of the iterates.

4.5.2 Neural Networks

For deep learning experiments, our results are in presence of corruptions via the directed noise model. In this corruption model, all the samples of class



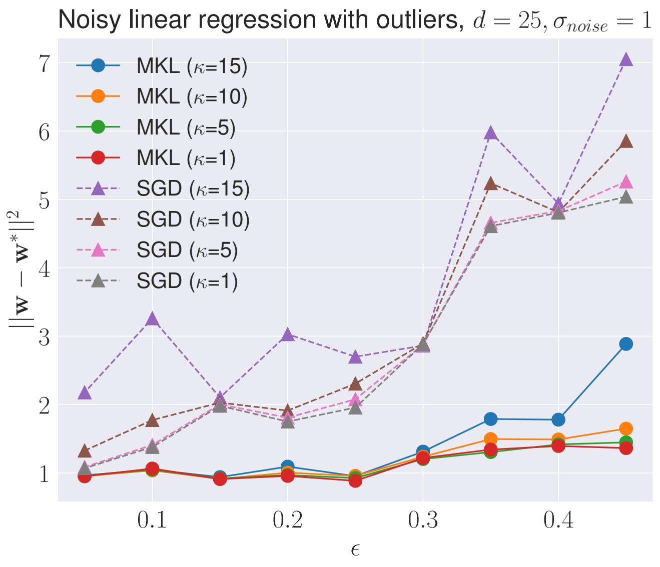


Figure 4.3: Comparing the performance of MKL-SGD ($k = 2$) and SGD for different values of κ in noiseless and noisy linear regression against varying fraction of outliers.

a that are in error are assigned the same wrong label b. This is a stronger corruption model than corruption by random noise (results in Appendix). For the MKL-SGD algorithm, we run a more practical batched (size b) variant such that if $k = 2$ the algorithm picks $b/2$ samples out of b sample loss evaluations. The oracle contains results obtained by running SGD over only non-corrupted samples.

MNIST: We train standard 2 layer convolutional network on subsampled MNIST (5000 samples with labels). We train over 80 epochs using an initial learning rate of 0.05 with the decaying schedule of factor 5 after every 30 epochs. The results of the MNIST dataset are averaged over 5 runs.

Dataset	**MNIST**		
$\epsilon\backslash$ Optimizer	**SGD**	**MKL-SGD**	**Oracle**
0.1	**96.76**	96.49	98.52
0.2	92.54	**95.76**	98.33
0.3	85.77	**95.96**	98.16
0.4	71.95	**94.20**	97.98

Table 4.1: Comparing the test accuracy of SGD and MKL-SGD ($k = 5/3$) over MNIST dataset in presence of corruptions via directed label noise.

CIFAR10: We train Resnet-18 [113] on CIFAR-10 (50000 training samples with labels) for over 200 epochs using an initial learning rate of 0.05 with the decaying schedule of factor 5 after every 90 epochs. The reported accuracy is based on the true validation set. The results of the CIFAR-10 dataset are averaged over 3 runs.

Dataset	**CIFAR10**		
$\epsilon \backslash$ Optimizer	**SGD**	**MKL-SGD**	**Oracle**
0.1	79.1	**81.94**	84.56
0.2	72.29	**77.77**	84.40
0.3	63.96	**66.49**	84.66
0.4	52.4	**53.57**	84.42

Table 4.2: Comparing the test accuracy of SGD and MKL-SGD ($k = 5/3$) over CIFAR-10 dataset in presence of corruptions via directed label noise.

Further experimental results on random noise as well as directed noise are available in the Appendix.

4.6 Discussion and Future Work

To ensure consistency, i.e. $\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \to 0$, we require that $k \geq n\epsilon + 1$. In all other cases, there will be a non-zero contribution from the outliers which keeps the MKL-SGD solution from exactly converging to $\boldsymbol{w}^*$. In this paper, we consider unknown ϵ and thus k should be a hyperparameter. For neural network experiments in the Appendix, we show that tuning k as a hyperparameter can lead to significant improvements in performance in presence of outliers.

The obvious question is if it is possible to provide worst case guarantees for a larger subset of problems using smarter initialization techniques. It will

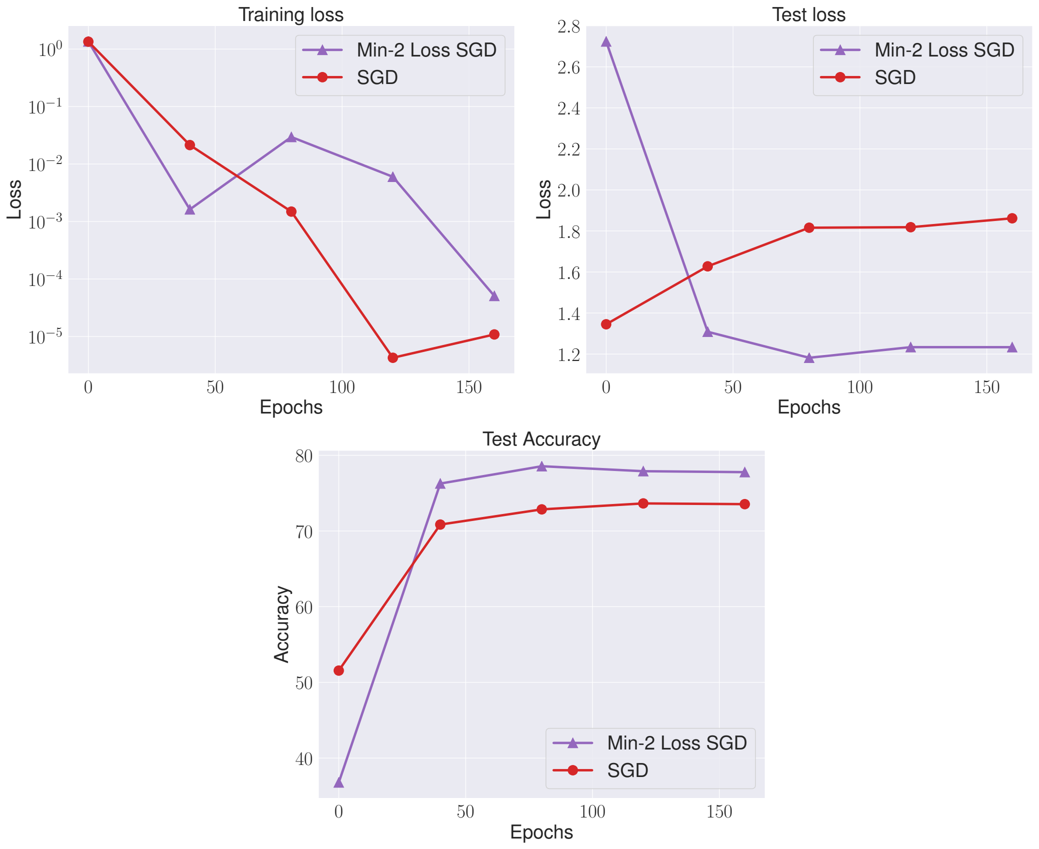


Figure 4.4: Comparing training loss, test loss and test accuracy of MKL-SGD and SGD. Parameters: $\epsilon = 0.2$, $k = 2$, $b = 16$. The training loss is lower for SGD which means that SGD overfits to the noisy data. The lower test loss and higher accuracy demonstrates the robustness MKL-SGD provides for corrupted data.

be interesting to analyze the tradeoff between better generalization guarantees offered by large k and rates of convergence. The worst case analysis in the noisy setting for standard convex optimization losses remains an open problem. As we show in the previous set of experiments, in presence of noise, tuning the hyperparameter k can provide significant boosts to the performance.

4.7 Conclusion

In this paper, we propose MKL-SGD that is computationally inexpensive, has linear convergence (upto a certain neighborhood) and is robust against outliers. We analyze MKL-SGD algorithm under noiseless and noisy settings with and without outliers. MKL-SGD outperforms SGD in terms of generalization for both linear regression and neural network experiments. MKL-SGD opens up a plethora of challenging questions with respect to understanding convex optimization in a non-convex landscape which will be discussed in the Appendix.

Chapter 5

Balancing SGD: Faster Optimization for Imbalanced Classification

A classification dataset is imbalanced if the number of training samples in each class varies widely. In such a setup, vanilla SGD quickly reaches a point where many samples from the majority class have small gradients and low loss, leading to insignificant gradients, which in turn leads to both slow convergence and poor generalization. In this paper, we propose a simple yet efficient variant of SGD that rejects samples from the majority class unless their current loss exceeds a threshold. We prove that this algorithm converges at a faster rate than vanilla SGD and has better generalization performance. Finally, we also illustrate these performance improvements via experiments on synthetic as well as real datasets.

5.1 Introduction

This paper focuses on multi-class classification settings where the training data is imbalanced (i.e., it has many more samples from some classes than from others). For a classification model trained by an algorithm like SGD, imbalanced datasets result in two issues: slow convergence during training, and final classifiers that are biased against minority classes. As we will see, slow convergence arises because, when SGD samples are chosen uniformly at random, most samples from the majority class have relatively small gradients and contribute minimally to the magnitude of the update – effectively wasting computation. On the other hand, bias – which results in poor generalization if the test set is not imbalanced – is a result of the majority class being over-represented in the loss function, if such a function is built naively from the training data.

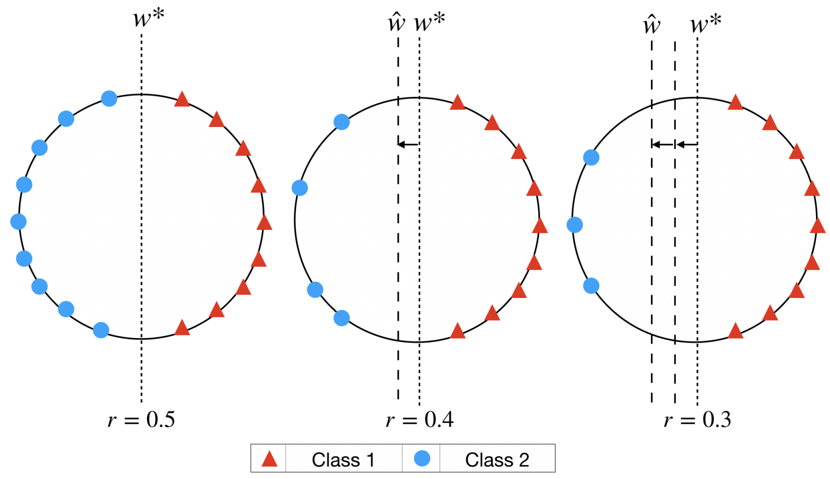


Figure 5.1: Illustration of how a skewed data distribution introduces a bias in classical estimation techniques. w^* determines the direction of the separating hyperplane in presence of balanced data, and $\hat{w}$ is the predicted estimator using SGD in presence of imbalanced data.

Our primary contribution is a simple modification of SGD, which alleviates both issues. Our **main idea** is to have *class-dependent thresholds* on the (current) loss of a sample – such that any randomly drawn sample not meeting the threshold corresponding to its class is not used for the gradient update.

In *Figure 5.1*, we highlight how the presence of imbalancedness leads to a biased estimation using a toy example for binary classification under the separable data regime. As the level of imbalancedness increases, the naive optimization algorithm converges to a point farther from the desired solution [114, 115]. A-priori knowledge of the label/data distribution can reduce the bias and ameliorate the generalization performance [11]. Label-distribution aware variants of SGD are quite popular [116, 12]. However, for massive datasets, knowledge of label distribution knowledge is often unavailable, and estimates for label distribution can be computationally challenging and intractable in the stochastic setting.

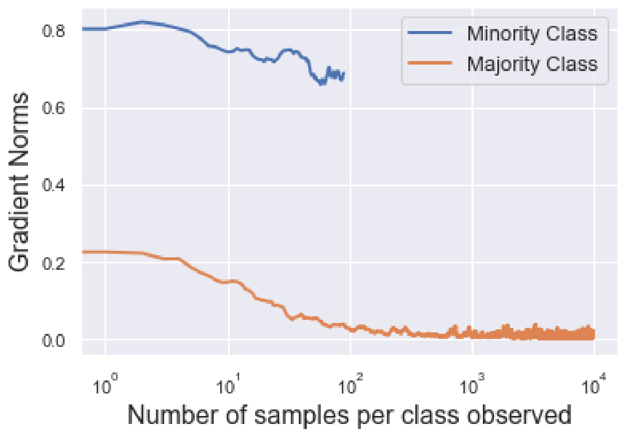


Figure 5.2: Toy example for logistic regression in the separable data setting. We plot the running average of the norm of the gradients vs. the number of samples of that class observed when the imbalance ratio is 0.01. The window size for the running average is 20. Samples from the majority class have insignificant gradient updates using SGD.

In *Figure 5.2*, we use a toy example for logistic regression to draw a comparison between the gradient contributions by the majority class and minority class. The presence of *imbalancedness* (as depicted in Figure 5.1) often biases the estimator $\hat{\boldsymbol{w}}$ away from $\boldsymbol{w}_*$. It is well known that the loss and the norm of the gradient of a data sample i are inversely proportional to its distance from $\hat{\boldsymbol{w}}$. Thus, for any classical method such as SGD, most samples from the majority class have small gradients leading to many insignificant gradient update steps, even though the optimization is far from complete. Thus, the presence of bias not only contributes to poor generalization on the test set but also to slow convergence. For the separable regime, under certain assumptions, [117, 118] explicitly characterize this slow rate of convergence to be $O\left(1/\log t\right)$ for the bias term and $O\left(1/(\log t)^2\right)$ for the direction.

In recent years, some papers have systematically studied the theory behind optimization algorithms, where samples from different classes have different relative importance. Amongst the ones that are closely related to our paper are Perceptron Algorithm with Uneven Margins (PAUM) in [114], DM-SGD in [119] and Instance Shrinking in [120]. However, [114] holds for the deterministic setting, and [119, 120] have to be significantly altered in the presence of *imbalancedness.* We aim to bridge the gap between theory and practice by providing a faster algorithm for stochastic optimization in imbalanced classification.

Main Contributions

i) We propose a simple, efficient, practical variant of SGD, called Balancing SGD (B-SGD) algorithm. This algorithm performs a gradient update step if the loss of an incoming data sample exceeds a class-dependent threshold. This allows B-SGD to handle the trade-off between overfitting to the majority and overfitting to the minority class in a principled manner.

ii) We provide two main theoretical results. In Theorem 1, we provide a threshold dependent upper bound on the number of gradient updates performed by B-SGD. Next, we propose a technique to choose the class-dependent threshold in the absence of the knowledge of label/data distributions.

iii) Lastly, we support our theoretical results with both synthetic logistic regression experiments that provide insight, as well as encouraging results on real imbalanced logistic regression and artificially generated imbalanced CIFAR-10.

5.2 Related Work

In the last few years, imbalanced learning has received considerable interest in the machine learning community [121, 116, 122, 123, 124]. The algorithms designed to cater to imbalanced data have one underlying theme: *design a proxy distribution that is closer in expectation to the test-set distribution.* These algorithms fall into the following categories:

Oversampling. Oversampling is one of the most popular choices within the family of resampling strategies. Here, the main idea is simple: reuse the samples of the minority class by resampling them. Naive over-sampling of the minority class does not lead to improvements in performance [125, 126, 9]. Popular variants of over-sampling include either reweighting the existing examples based on some criteria [10] or creating synthetic examples [9]. However, over-sampling variants often suffer from overfitting to minority class [127, 128] and so require regularization methods [129], which might result in significant memory requirements, poor performance on high-dimensional data [130] and slow convergence. The last point arises from the fact that as the number of samples increases, it lowers the relative importance of each sample [131].

Undersampling. Undersampling relies on the idea of removing the samples of the majority class to balance the distribution. Under-sampling methods [132, 127, 122] can often lead to the loss of crucial information about the estimator [11] if performed arbitrarily. Moreover, without knowing the data-distribution [12], determining the extent of resampling further affects the final performance. For large datasets, it becomes crucial as it is expensive to compute and store these statistics.

Loss-based classification. Loss-based classification (or Cost-sensitive learning) is another widely used approach to *balance* the distributions. Over the years, variants of cost-sensitive algorithms have been proposed such as MFE and MSFE loss [133], Focal Loss [2, 134], CSDNN [135], CoSen CNN [136], CSDBN-DE [137], Threshold moving [138, 116] and many others [139, 140, 141, 131, 142].

Ensemble methods. In ensemble learning, a combination of the above methods can be tailored to improve the performance for specific applications

[138, 143, 144, 145, 146]. [147, 148] provide a detailed comparison of various methods for imbalanced learning. However, most of the techniques require the knowledge of label distribution, limiting their applicability in the stochastic setting.

To alleviate these issues in imbalanced learning, we propose an algorithm called Balancing SGD (B-SGD), which decides on the fly whether to compute a gradient update according to the estimated imbalance ratio, as well as the label and loss value of the incoming data sample. We observe that by combining resampling strategies with cost-sensitive learning, we can improve the convergence rates on imbalanced datasets than SGD and other methods that are agnostic of the label distribution.

Algorithm 3 Balancing SGD (B-SGD)

```
1: Number of classes k; Threshold parameter $c > 0$; Threshold values $\boldsymbol{\tau} = [\tau_i], i \in [k]$.
2: Initialize $\boldsymbol{w}_0, \tau_i(0) = \tau_i, n_{y_i} = 0 \ \forall \ i \in [k]$
3: Given samples $\mathcal{S} = (\boldsymbol{x}_i, y_i)_{i=1}^{\infty}$
4: for $t = 1, 2, 3, \dots T$ do
5:     Choose a sample $i$ uniformly from $\mathcal{S}$
6:     if $f_i(\boldsymbol{w}_t) \geq \tau_{y_i}(t)$ then
7:         $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla f_i(\boldsymbol{w}_t)$
8:         $n_{y_i} = n_{y_i} + 1$
9:         $\tau_{y_i}(t+1) = c\left(\frac{k-1}{k} - \frac{\sum_{j \in \{j \in [k] | y_j \neq y_i\}} n_{y_j}}{\sum_{j \in [k]} n_{y_j}}\right)$
10:    else
11:        $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$, $\tau_{y_i}(t+1) = \tau_{y_i}(t)$
12:    end if
13:    $\tau_{y_j}(t+1) = \tau_{y_j}(t) \ \forall \ j \neq i$
14: end for
15: Return $\boldsymbol{w}_T$
```

5.3 Algorithm

We propose a simple variant of SGD, known as the Balancing-SGD (B-SGD) algorithm to improve both the generalization and convergence behavior of SGD: *in each step, calculate the loss of the incoming sample $f_i(\boldsymbol{w}_t)$, and perform a gradient update step for that sample if it exceeds the label-dependent loss threshold,* $\tau_{y_i}(t)$. In other words, let $\mathcal{T}_t \in \mathcal{S}$ indicate the set of samples in the training set $\mathcal{S}$ that satisfy the label-dependent loss threshold condition, i.e., $\mathcal{T}_t := \{i \in \mathcal{S} | f_i(\boldsymbol{w}_t) \geq \tau_{y_i}(t)\}$.

The effective gradient update for B-SGD is then given as follows:

$$\mathbb{E}\left[\boldsymbol{w}_{t+1}|\boldsymbol{w}_t\right] = \boldsymbol{w}_t - \eta \nabla f_i(\boldsymbol{w}_t) \mathbb{I}_{i \in \mathcal{T}_t}$$

Here, $\mathcal{T}_t$ is a dynamic set that can contain anywhere from 0 data samples to all data samples in S depending on the value of $\boldsymbol{w}_t$ and $\tau_{y_i}(t)$. The rejection of samples that have a low loss with a time-varying label-dependent threshold result in a dynamically changing loss function similar to [149].

Line 9 is the critical part of Algorithm 3, which describes the loss threshold selection process. While we elaborate on how threshold selection happens later in Section 5.4.4, let us motivate why Line 9 is helpful. i) If the number of gradient updates for all the classes is the same, then the loss threshold is zero for all classes. Thus, in the presence of a balanced training set, B-SGD mimics SGD. ii) The loss threshold should be inversely proportional to the number of gradient updates for samples of that class, i.e., larger the number of gradient updates for that class, higher should be the loss threshold.

iii) Let n_m and n_M be the number of gradient update steps taken for classes 1 and 2, respectively. For a simple setting with $k = 2$ classes and $n_1 < n_2$, we have $\tau_1 = c(0.5 - \frac{n_2}{n_1}) > 0, \tau_m = c(0.5 - \frac{n_2}{n_1}) < 0$ which passes the sanity check of selectively choosing samples from the majority class. iv) Note, that the threshold expression acts as a *self-adjusting function*, i.e. if fewer samples from a class i are selected, it leads to a reduction in the value of τ_{y_i} which in turn decreases the probability of rejection for a sample belonging to class i in future updates.

Thus, *B-SGD takes gradient update steps more frugally than SGD.* In practice, this is particularly important as computing gradients are usually more expensive than computing the loss functions for a given sample. For example, in deep learning, loss calculations often involve just forward propagation, while gradient computations involve both forward and backward propagation.

The algorithm exhibits two key features:

i. *Boosting-like updates* [150]: Update samples with higher loss more frequently than corresponding samples with the lower loss

ii. *Passive-aggressive updates* [151, 152, 153]: $\tau_{y_i}(t)$ is inversely proportional to the frequency of the gradient update step taken for each class, i.e., majority class will have a higher loss threshold $\tau_{y_i}(t)$ than the minority class

The B-SGD algorithm requires O(k) extra memory, where k is the

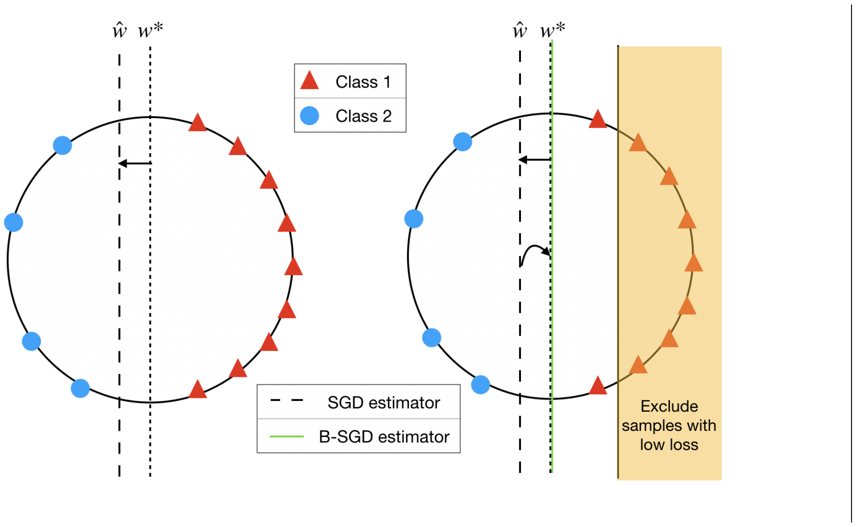


Figure 5.3: Introducing a label-dependent loss-based thresholding approach allows us to alleviate the issue of bias introduced by the skewed label distribution

number of classes. The computational complexity of the B-SGD algorithm is the same as SGD; $O(1)$ gradient/loss computations per incoming sample.

B-SGD belongs to the class of ensemble methods that combine undersampling and loss-based classification. It is important to note that while the undersampling for the majority class can include any subset of points, i.e., the ones with high loss, those with low loss, or those within some range of loss values. Here, the choice of the subset with high loss has twofold advantages: i) fewer gradient computations, ii) the information about support vectors which determine the separating hyper-plane is not lost.

We reiterate that similar approaches have been utilized in literature with limited success. These approaches suffer from many issues, such as catering to specific applications, expensive pre-computations, access to label distribution, and slow convergence in the stochastic setting. With B-SGD, we propose a systematic approach to alleviate these issues in imbalanced classification.

5.4 Theoretical Results for Logistic Regression

Notation: Throughout, $\|\cdot\|$ denotes the Euclidean ℓ_2 norm. $\mathcal{D}$ is the unknown distribution of the data set and $\mathcal{S}$ is the training data such that $|\mathcal{S}| = n$. Bold-faced lowercase letters will be used to denote vectors while plain lowercase letters indicate scalars. Without loss of generality, we also assume that the data is imbalanced with $y = 1$ denoting the minority class. $\mathcal{S}_{-1}$ and $\mathcal{S}_1$ represent the training data for majority class ($y = -1$) and minority class ($y = 1$) respectively. Let the imbalance ratio be denoted by $r = \frac{\#\text{ of }(+1)'s}{n}$. The function $h(s) : \mathbb{R} \to \mathbb{R}$ is an decreasing function of s. $\sigma(q) = \frac{1}{1+\exp(-q)}$ denotes the sigmoid function. $\boldsymbol{w}_*$ is the optimal solution for $L_{\mathcal{D}}(\boldsymbol{w})$.

5.4.1 System Model

For any dataset $\mathcal{S} := \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ with features $\boldsymbol{x}_i \in \mathbb{R}^d$ and labels $y_i \in \mathbb{R}$, the corresponding empirical risk $\mathcal{S}$ minimization problem can be formulated as:

$$\min_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w}) \qquad \text{where} \quad L_{\mathcal{S}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i \in \mathcal{S}}^{n} f_i(\boldsymbol{w}) \tag{5.1}$$

Here, $f_i(\boldsymbol{w}) := f(\boldsymbol{w}; (\boldsymbol{x_i}, y_i))$ represent convex, continuously differentiable loss functions for the sample $(\boldsymbol{x_i}, y_i)$. Similarly, the population risk minimization is expressed as follows:

$$\min_{w} L_{\mathcal{D}}(\boldsymbol{w}) \qquad \text{where} \;\; L_{\mathcal{D}}(\boldsymbol{w}) := \mathbb{E}_{i\sim\mathcal{D}}[f_i(\boldsymbol{w})] \tag{5.2}$$

Our aim is to use $\mathcal{S}$ to find an estimator that minimizes the population risk in equation (5.2). In the rest of this section, we focus on binary classification of imbalanced datasets using logistic regression.

Definition 5.4.1 (Linearly separable)**.** *For the generative logistic model, the underlying parameter $\boldsymbol{w}_* \in \mathbb{R}^{d+1}$ determines the separating hyperplane where $\boldsymbol{w}_*$ is defined as:*

$$\boldsymbol{w}_* = [(\boldsymbol{w}_*{}^{(d)})^\top \quad b_*]^\top \in \mathbb{R}^{d+1}, \quad \textit{where } \boldsymbol{w}_*{}^{(d)}/\|\boldsymbol{w}_*{}^{(d)}\| \textit{ is direction and } b_* \textit{is bias.} \tag{5.3}$$

The class for the input feature $\boldsymbol{x}^{(d)}$ in $\mathcal{D}$ is determined as follows:

$$\boldsymbol{x} = [(\boldsymbol{x}^{(d)})^\top \quad 1]^\top \in \mathbb{R}^{d+1}, \qquad y = \mathbb{1}\{\sigma(\boldsymbol{w}_*{}^\top \boldsymbol{x}) \geq 0.5\} - \mathbb{1}\{\sigma(\boldsymbol{w}_*{}^\top \boldsymbol{x}) < 0.5\} \tag{5.4}$$

where $(d+1)$th dimension is for the constant term and $\sigma(x) = 1/(1+e^{-x})$.

The deterministic generative process in equation (5.4) ensures that our training samples $\mathcal{S} = \{\boldsymbol{x_i}, y_i\}_{i=1}^n$ are linearly separable.

The logistic loss for data sample $(\boldsymbol{x_i}, y_i)$ at $\boldsymbol{w}$ is given as:

$$f_i(\boldsymbol{w}) = f(\boldsymbol{w}; (\boldsymbol{x_i}, y_i)) = \log\bigl(1 + \exp\bigl(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)\bigr)\bigr)$$

SGD treats all the samples in the training set equally and minimizes the empirical objective function:

$$L_{\mathcal{S}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \log\bigl(1 + \exp\bigl(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)\bigr)\bigr) \tag{5.5}$$

Note that ${\boldsymbol{w}_*}^\top = [{\boldsymbol{w}_*}^{(d)}, b_*]$ determines the unique separating hyperplane of the population loss $L_{\mathcal{D}}(\boldsymbol{w}_*)$ where the uniqueness is limited to a constant scaling of the direction vector ${\boldsymbol{w}_*}^{(d)}/\|{\boldsymbol{w}_*}^{(d)}\|$. For more detailed analysis, we refer to [118, 66].

Assumption 8. The hyperplane determined by $\boldsymbol{w}_*$ divides $\mathcal{D}$ into symmetric regions

Assumption 8 implies that if the data collection process were uniform, then we would have a balanced distribution. However, in this paper, we assume that the skewed distribution of the two classes arises from anomalies in the data-collection process.

5.4.2 Bias in imbalanced datasets

As we discussed previously, in the separable data setting, the first order stationary points i.e. those points $\boldsymbol{w}$ such that $\nabla L_{\mathcal{S}}(\boldsymbol{w}) = 0$ are attained only when $\boldsymbol{w} = \infty$. However, this is infeasible and in most practical scenarios, SGD is stopped when the gradient at point $\boldsymbol{w}$ satisfies the following approximate first order stationary point condition (defined in [154]):

$$\|\nabla L_{\mathcal{S}}(\boldsymbol{w})\| \leq \epsilon \tag{5.6}$$

Note that there are an infinite number of points that satisfy Equation (5.6) in the separable data setting. As we will show in Proposition 5.4.2, running SGD incurs a bias that not only generalizes poorly on the test distribution

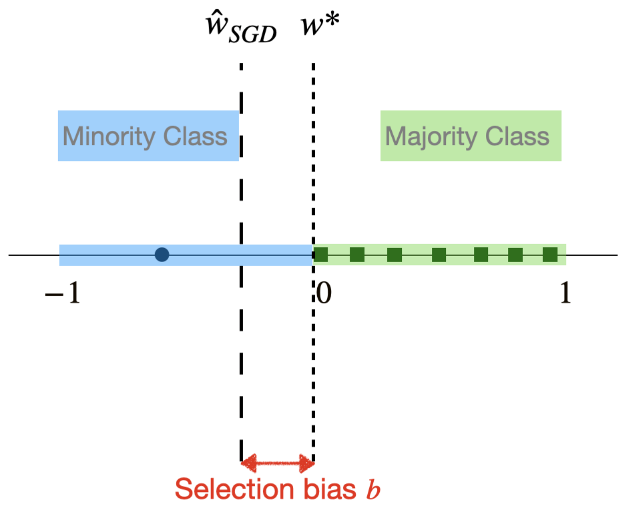


Figure 5.4: Visualizing selection bias in imbalanced datasets using a toy example. $\boldsymbol{w}_*$ is defined in Definition 8 and $\hat{\boldsymbol{w}}_{SGD}$ is the estimator running vanilla SGD safisfies (5.6).

$\mathcal{D}$ but also converges at an extremely slow rate to the maximum margin separating hyperplane of $\mathcal{S}$ as $O\left(1/\log t\right)$ for bias and $O\left(1/(\log t)^2\right)$ for direction [117, 118]. In the rest of this section, we will first establish the relationship between r and b (Proposition 5.4.2), and using a toy example (Fig. 5.4) establish how high bias leads to poor convergence and generalization guarantees.

Proposition 5.4.2. *Consider the loss function defined in equation* (5.5) *and assume* $\hat{\boldsymbol{w}}^\top = [\hat{\boldsymbol{w}}^{(d)}, \hat{b}]$ *is an approximate stationary point satisfying equation* (5.6). *That is, for some* $|\hat{b}| < \infty$ *and* $\varepsilon > 0$ *such that* $\forall i, y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)} \geq \log\left(1/\varepsilon\right)$ *and* $\exp\Big(-y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)} + \hat{b})\Big) < 1$. *Then* $\frac{\partial L(\boldsymbol{w})}{\partial b}|[]\boldsymbol{w} = \hat{\boldsymbol{w}} = 0$ *implies* *that*

$$\hat{b} \propto \log\left(\frac{\sum_{i\in\mathcal{S}_1} e^{-y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}}{\sum_{i\in\mathcal{S}_{-1}} e^{-y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}}\right). \tag{5.7}$$

Note that for Proposition 5.4.2, we do not make any specific assumptions and thus the bias expression holds for any $\hat{\boldsymbol{w}}$ that satisfies equation (5.6). We will utilize the toy example in Figure 5.4 to quantify the bias term. Here, the data lies in 1-D space and satisfies Assumption 8. Let $\boldsymbol{w}_* = [1, 0]$ and $\mathcal{D} = \{(x_i, y_i) | x_i \in U(-1, 1), y_i = sign(x_i)\}$. Consistent with our motivation in Section 5.1, the data collection process is skewed and in the toy example we consider an extreme case where there is only one sample in the minority class, i.e. $rn = 1$, given by $\bar{\boldsymbol{x}}_1 = [\bar{x}_1^{(d)}, 1]^\top$. Then, the number of samples in the majority class is $(1 - r)n = \frac{1-r}{r}$ where we assume that r is small ($1e^{-3}$ or less). $\hat{\boldsymbol{w}}$ satisfies equation (5.6) iff $\hat{b} \in [\bar{x}_1^{(d)}, 0]$ where $\bar{x}_1^{(d)} < 0$. Combining this setup with equation (5.7), the bias for the toy example in Figure 5.4 can be reduced to:

$$\hat{b} \propto \log\left(\frac{e^{-\bar{x}_i^{(d)}\hat{w}^{(d)}}}{\int_0^1 e^{x_i^{(d)}\hat{w}^{(d)}} dx_i^{(d)}}\right) \approx \log\left(\frac{\hat{w}^{(d)} e^{-\bar{x}_i^{(d)}\hat{w}^{(d)}}}{e^{\hat{w}^{(d)}} - 1}\right).$$

For large values of $|\hat{w}^{(d)}|$, the bias tends to 0. However, for small values of $|\hat{w}^{(d)}|$, however, the bias is significant. Since the test set follows the same distribution as $\mathcal{D}$ i.e. is balanced, $0.5|\hat{b}|$ fraction of samples are incorrectly classified. Thus, larger the magnitude of $|\hat{b}|$, higher is the generalization error of the estimator. Further, we also show in appendix section D.2 that the bias has an impact on the rate of convergence as well. This in turn implies that not all samples are equally useful especially at low values of $|\hat{\boldsymbol{w}}|$ and it can be worthwhile omitting certain samples from the majority class $\mathcal{S}_{-1}$ in the gradient update steps to improve both generalization and the rate of convergence.

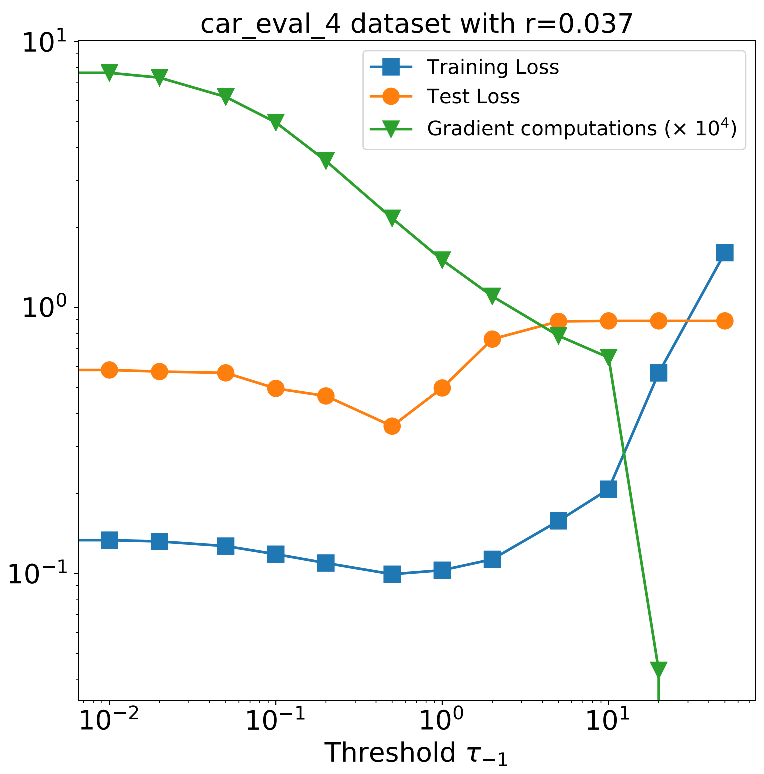


Figure 5.5: Here, we compare the training error, test error and no. of gradient computations for different values of fixed thresholds for the majority class. As threshold, τ_{-1}, increases from 0 to 50, we observe that the no. of gradient computations continues to decrease while both training and test loss initially decrease and then increase. Advantage of fixed thresholding: B-SGD $[\tau_{-1}, \tau_1] = [0.75, 0]$ achieves 37.5% and 71.7% decrease in the test error and gradient computations respectively over SGD ($\tau_{-1} = \tau_1 = 0$).

5.4.3 Finite Iteration Guarantees with Fixed Thresholding

For the sake of simplicity, in this sub-section, we assume that the loss threshold is fixed at its initial value, i.e. $\tau_{y_i}(t) = \tau_{y_i} \ \forall \ t$. For constant loss thresholds, Fig. 5.5 shows that as the loss threshold increases, the number of iterations required to attain an estimation error $\|\boldsymbol{w} - \boldsymbol{w}_*\| < \epsilon$ decreases (Regime A) initially until it increases and saturates at the maximum value (Regime B). The latter behavior is observed because for large thresholds, it is no longer possible to satisfy $\|\boldsymbol{w} - \boldsymbol{w}_*\| < \epsilon$ no matter how may steps we take. Let $\mathcal{S}_y = \{i \in \mathcal{S} | y_i = y\}$,

Theorem 5.4.3. *Consider the B-SGD algorithm with fixed thresholds (over time) τ_{-1} and τ_1 for the majority and minority class, respectively. The expected gradient update step for B-SGD is as follows:*

$$\mathbb{E}\left[\boldsymbol{w}_{t+1} | \boldsymbol{w}_t\right] = \boldsymbol{w}_t - \eta \nabla f_i(\boldsymbol{w}_t) \mathbb{I}_{f_i(\boldsymbol{w}_t) \geq \tau_{y_i}} \tag{5.8}$$

Suppose the data is normalized (i.e., $\|\boldsymbol{x}_i\| = 1$). Define the margin $\gamma(\boldsymbol{w}, \mathcal{S}_{-1}, \mathcal{S}_1) := \min_i \frac{y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{\|\boldsymbol{w}\|}$. Suppose there exists $\boldsymbol{w}_* \in \mathbb{R}^{d+1}$ such that $\|\boldsymbol{w}_*\| = 1$ and $\gamma(\boldsymbol{w}_*, \mathcal{S}_{-1}, \mathcal{S}_1) \geq \Gamma$. Then the number of gradient updates performed by Algorithm 3 is bounded w.h.p. by

$$T_\tau = \frac{1}{\Gamma^2 (1-r)\beta_{-1} + r\beta_1} + \frac{2(1-r)C_{-1} + 2rC_1}{\eta\left(\Gamma^2 (1-r)\beta_{-1} + r\beta_1\right)} \tag{5.9}$$

where $\beta_{y_i} = (\exp(\tau_{y_i}) - 1)\exp(-\tau_{y_i})$ and $C_{y_i} = -\log\left(\exp(\tau_{y_i}) - 1\right)$.

The proof of Theorem 5.4.3 is deferred to the appendix. The proof is similar to the analysis for the perceptron algorithm [114, 155, 156]. We adopt their ideas to our logistic regression in stochastic gradient descent for imbalanced data and significantly improve the bounds in terms of imbalance ratio r with high probability.

One of the main takeaways from Theorem 5.4.3 is that the number of updates T_τ is directly determined by r. If the classes are highly imbalanced satisfying $r = \mathcal{O}\left(1/\log\left(1/\varepsilon\right)\right)$, then for some ε setting $\tau_1 = \varepsilon$ and $\tau_{-1} > \varepsilon$ results in:

$$C_1 = (-\log(\exp(\varepsilon) - 1) = \mathcal{O}\left(\log\left(1/\varepsilon\right)\right) \text{ and } \beta_1 = (\exp(\varepsilon) - 1)\exp(-\varepsilon) = \mathcal{O}\left(\varepsilon\right)$$

which means $rC_1 = \mathcal{O}(1)$ and $r\beta_1 = \mathcal{O}(\varepsilon/(\log(1/\varepsilon))$ have little effect on the bound. In other words, taking all samples from highly-imbalanced minority class will have similar sample complexity as taking a subset of samples from the minority class with a small (constant) loss threshold. *This serves as an inspiration for our proposed Algorithm 3 in Section 5.4.4.* On the other hand, when the thresholds of both majority and minority are small (i.e., $\tau_{y_i} \to \varepsilon$), the update in (D.7) reduces to vanilla SGD, which means the bound T_τ in Theorem 5.4.3 is: $T_\varepsilon = \mathcal{O}\left(1/\varepsilon \log\left(1/\varepsilon\right)\right)$. The complexity T_ε matches the bound in [117]. Comparing with this vanilla SGD, Algorithm 3 with $\tau_1 = \varepsilon$ and constant τ_{-1} clearly achieves a smaller bound than vanilla SGD in terms of number of gradient updates performed. In this sense, we claim *Algorithm 3 has faster optimization.* Figure 5.3 provides further evidence that our algorithm is able to achieve both good generalization as well as faster convergence over SGD.

However, the B-SGD algorithm with fixed thresholds would require a good initial estimate of r which is difficult since we do not have access to any label or data distribution. With that in mind, in the next section, we propose a variable thresholding approach that allows us to apply B-SGD with an estimate R_t for imbalance ratio r based on the samples for which gradient update steps are taken.

5.4.4 Analysis of Variable Thresholding with Unknown r

To describe the variable thresholding approach, we first delve deeper into the analysis of stationary points for the binary classification problem. For the separable data case, we know that there can be many separating hyperplanes for a given set of data points, and all of them perform equally well on the training set. However, in order to guarantee good generalization, we need to understand why a specific solution will have better generalization. This, in turn, motivates how we tune the threshold as we perform an increasing number of updates.

As shown in Section 5.4.2, SGD will lead to biased estimation. However, if we know r in advance, we minimize re-weighted empirical loss leads to an estimator $\hat{\boldsymbol{w}}_{RW}$ that is closer to $\boldsymbol{w}_*$ than $\boldsymbol{w}_{SGD}$.

$$\hat{\boldsymbol{w}}_{RW} = \arg\min_{w} \left(\sum_{i \in \mathcal{S}_1} \log\bigl(1 + \exp\bigl(-\boldsymbol{w}^\top \boldsymbol{x}\bigr)\bigr) + \sum_{i \in \mathcal{S}_{-1}} \frac{r}{1-r} \log\bigl(1 + \exp\bigl(\boldsymbol{w}^\top \boldsymbol{x}\bigr)\bigr) \right) \tag{5.10}$$

The alternative is to under-sample the majority class by removing samples to have $\hat{\boldsymbol{w}}_{US}$ as

$$\hat{\boldsymbol{w}}_{US} = \arg\min_{\boldsymbol{w}} \Bigg(\sum_{i \in \mathcal{S}_1 \cap \mathcal{A}_w} \log\Big(1 + \exp\Big(-\boldsymbol{w}^\top \boldsymbol{x}\Big)\Big) + \sum_{i \in \mathcal{S}_{-1} \cap \mathcal{A}_w} \log\Big(1 + \exp\Big(\boldsymbol{w}^\top \boldsymbol{x}\Big)\Big)\Bigg) \tag{5.11}$$

where $\mathcal{A}_w := \{i \in \mathcal{S} | \log\left(1 + \exp\left(-y_i \boldsymbol{w}^\top \boldsymbol{x}\right)\right) > \tau_{y_i}\}$ and $\tau_{(.)}$ is a function of r. We wish to find the $\hat{\boldsymbol{w}}$ that satisfies equations (5.10) and (5.11), which implies:

$$\sum_{i \in \mathcal{S}_{-1}} \left(\frac{r}{1-r} - \mathbb{1}_{i \in \mathcal{A}_{\hat{\boldsymbol{w}}}} \right) \nabla f_i(\boldsymbol{w}) |[] \boldsymbol{w} = \hat{\boldsymbol{w}} = 0 \tag{5.12}$$

There exists some constant τ_{-1} that satisfies the equality in equation (5.12). However, to write down the closed-form expression of τ_{-1} is not straightforward.

Proposition 5.4.4. *Suppose the data is normalized (i.e., $\|\boldsymbol{x}_i\| = 1$). Suppose $\hat{\boldsymbol{w}}$ minimizes both equations* (5.10) *and* (5.11)*, then without loss of generality $\hat{\boldsymbol{w}}^\top \boldsymbol{x}_i < 0, \forall i \in \mathcal{S}_{-1}$. Set $\tau_{-1} \geq 0$ for the set $\mathcal{A}_{\hat{\boldsymbol{w}}}$. Solving for τ_{-1} given some r in equation* (5.12)*, we observe that $\tau_{-1} > 0$ is monotonically decreasing function with respect to $r \in [0, 0.5]$. In addition,*

$$\textit{if } r \to 0.5 \textit{ or } \tfrac{r}{1-r} \to 1, \textit{ then } \tau_{-1} \to 0; \qquad \textit{if } r \to 0 \textit{ or } \tfrac{r}{1-r} \to 0, \textit{ then } \tau_{-1} \to \log(1 + \exp(\|\hat{\boldsymbol{w}}\|)).$$

The proof is available in the appendix. Thus, based on the Proposition 5.4.4, we propose a criteria to accept a sample from this majority class by linearly interpolating τ_{-1} at $r = 0$ to $r = 0.5$:

$$\tau_{-1} \approx \log(1 + \exp(\|\hat{\boldsymbol{w}}\|))\,(0.5 - r) \tag{5.13}$$

B-SGD Algorithm with unknown r: Let $n_{min}(t)$ and $n_{maj}(t)$ denote the number of gradient updates for the minority and majority class respectively at time t. Since we assume that we are not aware of the label distribution at the start, let the running update of the parameter r denoted by R_t every time a gradient update is performed, i.e., $R_t = \frac{\hat{n}_{min,t}}{\hat{n}_{min,t}+\hat{n}_{maj,t}}$ which represents an empirical estimate of the imbalance ratio observed so far (w.r.t. gradient update steps). The loss threshold vector, $\boldsymbol{\tau}$, in Line 9 of Algorithm 3 becomes:

$$\tau_{-1}(t) = \log(1 + \exp(\|\hat{\boldsymbol{w}}\|))\,(0.5 - R_t)$$
$$\tau_1(t) = \log(1 + \exp(\|\hat{\boldsymbol{w}}\|))\,(R_t - 0.5)$$

where $R_t = \hat{n}_{min,t}/(\hat{n}_{min,t} + \hat{n}_{maj,t})$. This implies that for any binary classification problem, at any point of time exactly one class can have a positive loss threshold.

5.5 Experiments

In this section, we compare B-SGD and SGD for two different scenarios:

5.5.1 Synthetic experiments

For this subsection, we consider a generative model in the separable setting as described in Definition 5.4.1. We compare the performance of SGD and B-SGD, where we plot both estimation error vs. the number of gradient computations and estimation error, $\|\boldsymbol{w} - \boldsymbol{w}_*\|$ vs. time taken. We observe that B-SGD outperforms SGD in the number of gradient computations and the time taken to achieve the same estimation error.

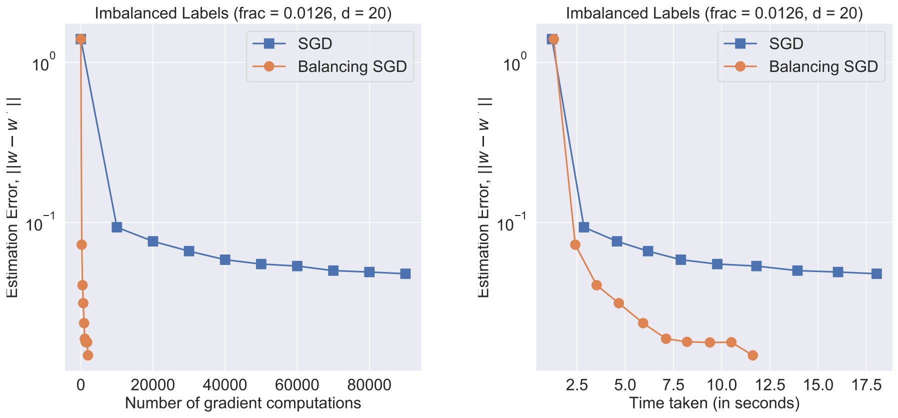


Figure 5.6: Comparing the rate of convergence vs the number of gradient computations and time taken for SGD and Balancing SGD. The reported results in the figure above are over an average of 3 runs.

5.5.2 Real Imbalanced datasets

In this section, we compare the performance of B-SGD and SGD for various *imbalanced datasets defined in the imblearn package* [157]. We observe that using B-SGD allows us to achieve not only better test loss (test accuracy) but also requires significantly fewer gradient computations than SGD. Consistent with our motivation, the training dataset was imbalanced, while the evaluation was on a balanced test set. Table **??** illustrates that B-SGD achieves significant performance gains in terms of both test loss, test accuracy (AUC), and the number of gradient computations over its SGD counterpart as well as the Focal Loss method [2]. The presence of higher training errors but lower test errors provides glaring evidence that B-SGD has better gener-

Algorithm	**SGD**				**B-SGD**				**Focal**
Dataset	TL	TEL	TA	GC	TL	TEL	TA	GC	TA
car_eval_4	**0.031**	**0.247**	88.1	$8.64 \cdot e^4$	0.278	0.249	**89.3**	$\mathbf{9.98 \cdot e^3}$	89.2
optical_digits	**0.054**	0.387	89.0	$1.41 \cdot e^5$	1.479	**0.367**	**89.6**	$\mathbf{8.11 \cdot e^3}$	88.0
isolet	**0.035**	0.649	87.0	$1.94 \cdot e^5$	0.485	**0.585**	**88.5**	$\mathbf{8.80 \cdot e^3}$	88.4
letter_img	**0.061**	0.66	83.8	$2.00 \cdot e^5$	1.81	**0.612**	83.8	$\mathbf{3.34 \cdot e^3}$	**84.3**
pen_digits	**0.097**	0.388	83.8	$2.74 \cdot e^5$	0.711	**0.364**	**85.1**	$\mathbf{3.33 \cdot e^4}$	**85.1**
mammography	**0.051**	0.901	70.9	$5.59 \cdot e^4$	0.144	**0.873**	70.5	$\mathbf{2.12 \cdot e^3}$	**71.1**
	TL	TEL	TE1	Epochs	TL	TEL	TE1	Epochs	TE1
CIFAR-10	**0.021**	1.43	28.5	200	0.163	**0.90**	**26.2**	**88**	28.7

Table 5.1: Comparing training loss (TL), Test Loss (TEL), Test AUC (TA), Top-1 Test Error (TE1), and Number of gradient computations (GC) for SGD and B-SGD over different Imbalanced datasets. The reported results for the first 6 datasets are an average of 5 runs, and for the last 3 datasets are an average of 3 runs. Focal loss (Focal) is the state-of-the-art method proposed in [2], which changes the loss function and so it is not fair to compare the training and the test errors. Focal has the same number of gradient computations as SGD. Hence, we only report test accuracy for Focal.

alization than SGD.

Lastly, we also ran experiments on *CIFAR-10 with imbalanced classes with long-tailed imbalance ratio* $r = 0.01$ using Resnet-32 architecture the codebase provided in [12] and reported the results in Table **??**. Details about the experimental setup, as well as more experiments for both synthetic and real datasets, along with a discussion on practical aspects of implementation including the effect of early stopping and parameter sensitivity, can be found in the appendix.

Appendices

Appendix A

Cheap-SVRG

A.1 Proof of Theorem 2.4.2

By assumptions of the theorem, we have:

$$\eta < \frac{1}{4L\left((1+\theta)+\frac{1}{s}\right)} \text{ and } K > \frac{1}{(1-\theta)\eta\left(1-4L\eta\right)\gamma},$$

As mentioned in the remarks of Section 2.4, the above conditions are sufficient to guarantee $\rho < 1$, for some $\theta \in (0,1)$. Further, for given accuracy parameter ϵ, we assume $\kappa \leq \frac{\epsilon}{2}$.

Let us define

$$\varphi_t := \mathbb{E}\big[F(\widetilde{\boldsymbol{w}}_t) - F(\boldsymbol{w}^*)\big],$$

as in the proof of Theorem 2.4.1. In order to satisfy $varphi_T \leq \epsilon$, it is sufficient to find the required number of iterations such that $\rho^T \varphi_0 \leq \frac{\epsilon}{2}$. In particular:

$$\begin{aligned}\rho^T\varphi_0 \leq \frac{\epsilon}{2} &\Rightarrow -\left(T\log\rho + \log\varphi_0\right) \geq -\log\frac{\epsilon}{2}\\ &\Rightarrow T\cdot\log\left(\rho^{-1}\right) \geq \log\frac{2}{\epsilon} + \log\varphi_0\\ &\Rightarrow T \geq \left(\log\frac{1}{\rho}\right)^{-1}\cdot\log\left(\frac{2\left(F(\widetilde{\boldsymbol{w}}_0) - F(\boldsymbol{w}^*)\right)}{\epsilon}\right)\end{aligned}$$

Moreover, each epoch involves K iterations in the inner loop. Each inner loop iteration involves two atomic gradient calculations. Combining the above, we conclude that the total number of gradient computations required to ensure that $\varphi_T \leq \epsilon$ is $O\left((2K + s)\log \frac{1}{\epsilon}\right)$.

A.2 Mini-batches in CheapSVRG

In the sequel, we show how Alg. 1 can also accommodate mini-batches in the inner loops and maintain similar convergence guarantees, under Assumptions 1-4. The resulting algorithm is described in Alg. 4. In particular:

Algorithm 4 CheapSVRG with mini batches

```
1: **Input**: $\widetilde{\boldsymbol{w}}_0, \eta, s, q, K, T$.
2: **Output**: $\widetilde{\boldsymbol{w}}_T$.
3: **for** $t = 1, 2, \dots, T$ **do**
4:     Randomly select $\mathcal{S}_t \subset [n]$ with cardinality $s$.
5:     Set $\widetilde{\boldsymbol{w}} = \widetilde{\boldsymbol{w}}_{t-1}$ and $\mathcal{S} = \mathcal{S}_t$.
6:     $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}} = \frac{1}{s}\sum_{i \in \mathcal{S}} \nabla f_i(\widetilde{\boldsymbol{w}})$.
7:     $\boldsymbol{w}_0 = \widetilde{\boldsymbol{w}}$.
8:     **for** $k = 1, \dots, K-1$ **do**
9:         Randomly select $\mathcal{Q}_k \subset [n]$ with cardinality $q$.
10:        Set $\mathcal{Q} = \mathcal{Q}_k$.
11:        $\boldsymbol{v}_k = \nabla f_{\mathcal{Q}}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}$.
12:        $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \cdot \boldsymbol{v}_k$.
13:    **end for**
14:    $\widetilde{\boldsymbol{w}}_t = \frac{1}{K}\sum_{j=0}^{K-1} \boldsymbol{w}_j$.
15: **end for**
```

Theorem A.2.1 (Iteration invariant)**.** *Let $\boldsymbol{w}^*$ be the optimal solution for minimization* (2.1)*. Further, let s, q, η, T and K be user defined parameters* *such that*

$$\rho \stackrel{def}{=} \frac{q}{\eta \cdot (q - 4L \cdot \eta) \cdot K \cdot \gamma} + \frac{4L \cdot \eta \cdot (s+q)}{(q - 4L \cdot \eta) \cdot s} < 1.$$

Under Asm. 1-4, CheapSVRG *satisfies the following:*

$$\mathbb{E}\big[F(\widetilde{\boldsymbol{w}}_T) - F(\boldsymbol{w}^*)\big] \leq \rho^T \cdot (F(\widetilde{\boldsymbol{w}}_0) - F(\boldsymbol{w}^*)) + \frac{q}{q - 4L\eta} \cdot \left(\frac{2\eta}{s} + \frac{\zeta}{K}\right) \cdot \max\left\{\xi, \xi^2\right\} \cdot \frac{1}{1-\rho}.$$

A.3 Proof of Theorem A.2.1

To prove A.2.1, we analyze CheapSVRG starting from its core inner loop (Lines 9-14). We consider a fixed subset $\mathcal{S} \subseteq [n]$ and show that in expectation, the steps of the inner loop make progress towards the optimum point. Then, we move outwords to the 'wrapping' loop that defines consecutive epochs to incorporate the randomness in selecting the set $\mathcal{S}$.

We consider the kth iteration of the inner loop, during the tth iteration of the outer loop; we consider a fixed set $\mathcal{S} \subseteq [n]$, starting point $\boldsymbol{w}_0 \in \mathbb{R}^p$ and (partial) gradient information $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}} \in \mathbb{R}^p$ as defined in Steps $6-8$ of Alg. 1. The set $\mathcal{Q}_k$ is randomly selected from $[n]$ with cardinality $|\mathcal{Q}_k| = q$. Similarly to the proof of Thm. 2.4.1, we have:

$$\mathbb{E}_{\mathcal{Q}_k}\left[\|\boldsymbol{w}_k - \boldsymbol{w}^*\|_2^2\right] = \|\boldsymbol{w}_{k-1} - \boldsymbol{w}^*\|_2^2 - 2\eta \cdot (\boldsymbol{w}_{k-1} - \boldsymbol{w}^*)^\top \mathbb{E}_{\mathcal{Q}_k}[\boldsymbol{v}_k] + \eta^2 \mathbb{E}_{\mathcal{Q}_k}\left[\|\boldsymbol{v}_k\|_2^2\right]. \tag{A.1}$$

where the expectation is with respect to the random variable $\mathcal{Q}_k$. By the definition of $\boldsymbol{v}_k$ in Line 12,

$$\begin{aligned}\mathbb{E}_{\mathcal{Q}_k}[\boldsymbol{v}_k] &= \mathbb{E}_{\mathcal{Q}_k}[\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}] \\ &= \nabla F(\boldsymbol{w}_{k-1}) - \nabla F(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}, \qquad \text{(A.2)}\end{aligned}$$

where the second step follows from the fact that $\mathcal{Q}_k$ is selected uniformly at random from $[n]$. Similarly,

$$\begin{aligned}\mathbb{E}_{\mathcal{Q}_k}\left[\|\boldsymbol{v}_k\|_2^2\right] &= \mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}\|_2^2\right] \\ &\overset{(i)}{\leq} 4 \cdot \mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\boldsymbol{w}^*)\|_2^2\right] \\ &\quad + 4 \cdot \mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\widetilde{\boldsymbol{w}}) - \nabla f_{\mathcal{Q}_k}(\boldsymbol{w}^*)\|_2^2\right] + 2 \cdot \|\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}\|_2^2 \\ &\overset{(ii)}{\leq} \frac{8L}{q} \cdot (F(\boldsymbol{w}_{k-1}) - F(\boldsymbol{w}^*) + F(\widetilde{\boldsymbol{w}}) - F(\boldsymbol{w}^*)) + 2 \cdot \|\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}\|_2^2. \qquad \text{(A.3)}\end{aligned}$$

Inequality (i) is follows by applying $\|\mathbf{x} - \mathbf{y}\|_2^2 \leq 2\|\mathbf{x}\|_2^2 + 2\|\mathbf{y}\|_2^2$ on all atomic gradients indexed by $\mathcal{Q}_k$, while (ii) is due to the following lemma.

Lemma A.3.1. *Given putative solution $\boldsymbol{w}_{k-1}$ and mini-batch $\mathcal{Q}_k$ with cardinality $|\mathcal{Q}_k| = q$, the following holds true on expectation:*

$$\mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\boldsymbol{w}^*)\|_2^2\right] \leq q^{-1} \cdot 2L \cdot (F(\boldsymbol{w}_{k-1}) - F(\boldsymbol{w}^*)).$$

Proof. let

$$\mathfrak{Q} = \left\{\mathcal{Q}^i \ : \ |\mathcal{Q}^i| = |\mathcal{Q}|, \ \mathcal{Q}^i \subset [n], \ \mathcal{Q}^i \neq \mathcal{Q}^j, \forall i \neq j\right\},$$

i.e., $\mathfrak{Q}$ contains all different index sets of cardinality $|\mathcal{Q}| = q$. Observe that $|\mathfrak{Q}| = \binom{n}{|\mathcal{Q}|}$.Note that the set $\mathcal{Q}_k$ randomly selected in the inner loop of Alg. 1

is a member of $\mathfrak{Q}$. Then:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\boldsymbol{w}^*)\|_2^2\right] \\
&= \mathbb{E}\Big[\Big\||\mathcal{Q}_k|^{-1} \cdot \sum_{i \in \mathcal{Q}_k} (\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*))\Big\|_2^2\Big] \\
&= \sum_{\mathcal{Q}^j \in \mathfrak{Q}} \mathbb{P}[\mathcal{Q}^j] \cdot \Big\|q^{-1} \cdot \sum_{i \in \mathcal{Q}^j} (\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*))\Big\|_2^2 \\
&\overset{(i)}{=} \sum_{\mathcal{Q}^j \in \mathfrak{Q}} \binom{n}{q}^{-1} \cdot \Big\|q^{-1} \cdot \sum_{i \in \mathcal{Q}^j} (\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*))\Big\|_2^2 \\
&\overset{(ii)}{=} \binom{n}{q}^{-1} \cdot q^{-2} \cdot \sum_{\mathcal{Q}^j \in \mathfrak{Q}} \Big\|\sum_{i \in \mathcal{Q}^j} (\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*))\Big\|_2^2 \\
&\leq \binom{n}{q}^{-1} \cdot q^{-2} \cdot \sum_{\mathcal{Q}^j \in \mathfrak{Q}} \sum_{i \in \mathcal{Q}^j} \big\|(\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*))\big\|_2^2.
\end{aligned}$$

Here, equality (i) is follows from the fact that each $\mathcal{Q}^j$ is selected equiprobably from the set $\mathfrak{Q}$. Equality (ii) is due to the fact that $|\mathcal{Q}^j| = q, \ \forall i$.

Since each set in $\mathfrak{Q}$ has cardinality q, each data sample $i \in [n]$ contributes exactly $\binom{n-1}{q-1}$ summands in the double summation above. This further leads to:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\boldsymbol{w}^*)\|_2^2\right] \\
&\leq \binom{n}{q}^{-1} \cdot \frac{1}{q^2} \cdot \binom{n-1}{q-1} \sum_{i=1}^{n} \|\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*)\|_2^2 \\
&\leq q^{-1} \cdot n^{-1} \cdot \sum_{i=1}^{n} \|\nabla f_i(\boldsymbol{w}_{k-1}) - \nabla f_i(\boldsymbol{w}^*)\|_2^2 \\
&\overset{(i)}{\leq} q^{-1} \cdot 2L \cdot (F(\boldsymbol{w}_{k-1}) - F(\boldsymbol{w}^*)).
\end{aligned}$$

Inequality (i) follows from the the fact that for any $\boldsymbol{w} \in \mathbb{R}^p$ [3],

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \|\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^*)\|_2^2 \leq 2L \cdot (F(\boldsymbol{w}) - F(\boldsymbol{w}^*)) \,. \tag{A.4}$$

Similarly, for $\widetilde{\boldsymbol{w}}$,

$$\mathbb{E}_{\mathcal{Q}_k}\left[\|\nabla f_{\mathcal{Q}_k}(\boldsymbol{w}_{k-1}) - \nabla f_{\mathcal{Q}_k}(\boldsymbol{w}^*)\|_2^2\right] \leq q^{-1} \cdot 2L \cdot (F(\widetilde{\boldsymbol{w}}) - F(\boldsymbol{w}^*)) \,. \tag{A.5}$$

□

Using the above lemma in (A.3), the remainder of the proof follows that of Theorem 2.4.1.

A.4 Coordinate updates in CheapSVRG

In large-scale settings, even computing estimates of the gradient, as in stochastic gradient variants where $\nabla f_i(\boldsymbol{w}) \in \mathbb{R}^d$, requires a lot of computations when d is large. Under such circumstances, it might be desirable to compute partially the estimate $\nabla f_i(\boldsymbol{w})$ by focusing only on a selected subset of its entries. This idea leads to *Coordinate Descent* (CD) algorithms, where even only a single component of $\nabla f_i(\boldsymbol{w})$ is updated per iteration. We refer the reader to [] and the excellent recent survey by Stephen Wright [] for more information about the history of CD variants.

In this section, we describe CheaperSVRG, a coordinate-descent variant of CheapSVRG. The description of CheaperSVRG is provided in Algorithm 5. The only difference with CheapSVRG is that, in every inner iteration, CheaperSVRG updates exclusively a randomly selected coordinate of the gradient of the selected component $\nabla f_i(\cdot)$.

In this section, we introduce the following notation. Given a set $\mathcal{B} \subset [d]$, $\nabla_{\mathcal{B}} f_i(\boldsymbol{w}) \in \mathbb{R}^d$ denotes the gradient of the i-th component of f, supported on the set $\mathcal{B}$: *i.e.*, $(\nabla_{\mathcal{B}} f_i(\boldsymbol{w}))_{\mathcal{B}^c} = 0$ for $\mathcal{B}^c := [d] \setminus \mathcal{B}$. With a slight abuse of notation, we will use the same notation $\nabla_{\mathcal{B}} f_i(\boldsymbol{w})$ to denote the information of $\nabla f_i(\boldsymbol{w})$ in $\mathbb{R}^d$ and $\mathbb{R}^{|\mathcal{B}|}$; the distinction will be apparent from the context. We also use $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}^{\mathcal{B}_k} \in \mathbb{R}^d$ to denote the restriction of $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}$ only on indices from $\mathcal{B}_k$.

Given the above, anti-gradient direction $-\boldsymbol{v}_k$ is only supported on $\mathcal{B}_k$. In the special case where $\mathcal{S} \equiv [n]$, we have:

$$\begin{aligned}
\mathbb{E}_{i_k, \mathcal{B}_k}\left[\boldsymbol{v}_k\right] &= \tfrac{d}{b} \cdot \mathbb{E}_{i_k}\left[\mathbb{E}_{\mathcal{B}_k}\left[\nabla_{\mathcal{B}_k} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\mathcal{B}_k} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}^{\mathcal{B}_k} \mid i_k\right]\right] \\
&\overset{(i)}{=} \tfrac{d}{b} \cdot \mathbb{E}_{i_k}\left[\sum_{\mathcal{B}^j \in \mathfrak{B}} \mathbb{P}[\mathcal{B}^j] \cdot \left(\nabla_{\mathcal{B}_k} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\mathcal{B}_k} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}^{\mathcal{B}_k}\right)\right] \\
&= \tfrac{d}{b} \cdot \mathbb{E}_{i_k}\left[\binom{d}{b}^{-1} \cdot \sum_{\mathcal{B}^j \in \mathfrak{B}} \sum_{\ell \in \mathcal{B}^j} \left(\nabla_{\ell} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\ell} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}^{\ell}\right)\right] \\
&\overset{(ii)}{=} \tfrac{d}{b} \cdot \mathbb{E}_{i_k}\left[\binom{d}{b}^{-1} \cdot \binom{d-1}{b-1} \cdot \sum_{\ell \in [d]} \left(\nabla_{\ell} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\ell} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}^{\ell}\right)\right] \\
&= \mathbb{E}_{i_k}\left[\nabla f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}\right] = \nabla f(\boldsymbol{w}_{k-1})
\end{aligned}$$

where in (i) we assume fixed i_k in the expression of the expectation and $\mathfrak{B}$ denotes the set of all possible $\mathcal{B}_k$ selections and (ii) follows from similar arguments in Section A.2. This justifies the weighting factor $\frac{d}{b}$ in front of $\boldsymbol{v}_k$.

The next theorem contains the iteration invariant for CheaperSVRG.

Algorithm 5 CheaperSVRG

```
1: Input: $\widetilde{\boldsymbol{w}}_0, \eta, s, b, K, T$.
2: Output: $\widetilde{\boldsymbol{w}}_T$.
3: for $t = 1, 2, \dots, T$ do
4:     Randomly select $\mathcal{S}_t \subset [n]$ with cardinality $s$.
5:     Set $\widetilde{\boldsymbol{w}} = \widetilde{\boldsymbol{w}}_{t-1}$ and $\mathcal{S} = \mathcal{S}_t$.
6:     $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}} = \frac{1}{s}\sum_{i\in\mathcal{S}} \nabla f_i(\widetilde{\boldsymbol{w}})$.
7:     $\boldsymbol{w}_0 = \widetilde{\boldsymbol{w}}$.
8:     for $k = 1, \dots, K-1$ do
9:         Randomly select $i_k \subset [n]$.
10:        Randomly select $\mathcal{B}_k \subset [d]$ with cardinality b.
11:        $\boldsymbol{v}_k = \frac{d}{b} \cdot \left(\nabla_{\mathcal{B}_k} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\mathcal{B}_k} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}^{\mathcal{B}_k}\right)$.
12:        $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \cdot \boldsymbol{v}_k$.
13:    end for
14:    $\widetilde{\boldsymbol{w}}_t = \frac{1}{K}\sum_{j=0}^{K-1} \boldsymbol{w}_j$.
15: end for
```

Theorem A.4.1 (Iteration invariant)**.** *Let $\boldsymbol{w}^*$ be the optimal solution for minimization* (2.1)*. Further, let s, b, η, T and K be user defined parameters such that*

$$\rho \overset{def}{=} \frac{1}{\eta \cdot \left(q - 4L \cdot \eta \cdot \frac{p}{b}\right) \cdot K \cdot \gamma} + \frac{4L \cdot \eta \cdot \left(\frac{p}{b} + \frac{1}{s}\right)}{\left(1 - 4L \cdot \eta \frac{p}{b}\right)} < 1.$$

Under Asm. 1-4, CheapSVRG *satisfies the following:*

$$\mathbb{E}\big[F(\widetilde{\boldsymbol{w}}_T) - F(\boldsymbol{w}^*)\big] \leq \rho^T \cdot (F(\widetilde{\boldsymbol{w}}_0) - F(\boldsymbol{w}^*)) + \frac{1}{1 - 4L\eta\frac{p}{b}} \cdot \left(\frac{2\eta}{s} + \frac{\zeta}{K}\right) \cdot \max\left\{\xi, \xi^2\right\} \cdot \frac{1}{1-\rho}.$$

A.4.1 Proof of Theorem A.4.1

In our case, where we use $\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}$, one can easily derive:

$$\mathbb{E}_{i_k, \mathcal{B}_k}\left[\boldsymbol{v}_k\right] = \nabla F(\boldsymbol{w}_{k-1}) - \nabla F(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}$$

and

$$
\begin{aligned}
\mathbb{E}_{i_k,\mathcal{B}_k}\left[\|\boldsymbol{v}_k\|_2^2\right] &= \mathbb{E}_{i_k}\left[\mathbb{E}_{\mathcal{B}_k}\left[\left\|\tfrac{p}{b}\cdot\left(\nabla_{\mathcal{B}_k} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\mathcal{B}_k} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}^{\mathcal{B}_k}\right)\right\|_2^2 \mid i_k\right]\right] \\
&\left(\tfrac{d}{b}\right)^2\cdot\mathbb{E}_{i_k}\left[\sum_{\mathcal{B}^j\in\mathfrak{B}}\mathbb{P}[\mathcal{B}^j]\cdot\left\|\nabla_{\mathcal{B}_k} f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla_{\mathcal{B}_k} f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}^{\mathcal{B}_k}\right\|_2^2\right] \\
&= \tfrac{p}{b}\cdot\mathbb{E}_{i_k}\left[\|\nabla f_{i_k}(\boldsymbol{w}_{k-1}) - \nabla f_{i_k}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}_{\mathcal{S}}\|_2^2\right] \\
&\leq \tfrac{p}{b}\cdot\left(8L\left(F(\boldsymbol{w}_{k-1}) - F(\boldsymbol{w}^*) + F(\widetilde{\boldsymbol{w}}) - F(\boldsymbol{w}^*)\right) + 2\|\widetilde{\boldsymbol{\mu}}_{\mathcal{S}}\|_2^2\right)
\end{aligned}
$$

Using the above quantities in CheapSVRG proof and making changes accordingly, we get the desired result.

Appendix B

On the Generalization of Adaptive Methods

B.1 Folklore theorem on convergence of matrices

We will first present a folklore theorem

Theorem B.1.1 (Behavior of square matrix $\|M^K\|_2$). *[74, 75] Let M is a $d \times d$ matrix. Let $\tau(M) = \max_i |\lambda_i(M)|$ denote the spectral radius of the matrix M. Then, there exists a sequence $\varepsilon_K \geq 0$ such that: $\|M^K\|_2 \leq (\tau(M) + \varepsilon_K)^K$, and $\lim_{K\to\infty} \varepsilon_K = 0$.*

Using the above theorem, H has $\tau(H) < 1$. Further, for sufficiently large $k < K$, ε_K has a small value such that $\tau(H) + \varepsilon_K < 1$; *i.e.*, after some $k_1 < K$, $(\tau(H)+\varepsilon_{k_1})^{k_1}$, will be less than zero, converging to zero for increasing k_1. As K is going towards infinity, this concludes the proof, and leads to the *left inverse* solution: $w_\infty = (-X^\top X)^{-1} \cdot (-I)\, X^\top y = (X^\top X)^{-1} X^\top y \equiv w^\star$, as $K \to \infty$.

B.2 Proof of Proposition 3.3.1

Proposition 3.3.1 implies that adaptive methods with full-rank positive definite preconditioners perform as well as their pure gradient based counter parts when it comes to fitting their training data. However, this proposition gives no information regarding the converged $\boldsymbol{w}^\star$.

We will prove this proposition using induction.

Base case: Here, we compute the first iteration, $K = 1$:

$$\hat{y}_1 = -\left(\prod_{i=0}^{0}\left(I - \eta X D_i X^\top\right) - I\right) y = -\left(I - \eta X D_0 X^\top - I\right) y = \eta X D_0 X^\top y$$

where, once again, we abuse the notation $\prod_{i=0}^{0} A_i = A_0$. This is the same result as in unfolding the recursion for $k = 0$ above, and assuming $w_0 = 0$.

Inductive case: Assume the following is true

$$\widehat{y}_{K-1} = -\left(\prod_{i=K-2}^{0}\left(I - \eta X D_i X^\top\right) - I\right) y = X w_{K-1}$$

Then,

$$\begin{aligned}
\widehat{y}_K = X w_K &= X w_{K-1} - \eta X D_{K-1} X^\top (X w_{K-1} - y) \\
&= (I - \eta X D_{K-1} X^\top) X w_{K-1} + \eta X D_{K-1} X^\top y \\
&= -(I - \eta X D_{K-1} X^\top)\left(\prod_{i=K-2}^{0}\left(I - \eta X D_i X^\top\right) - I\right) y \\
&\quad + \eta X D_{K-1} X^\top y \\
&= -\left(\prod_{i=K-1}^{0}\left(I - \eta X D_i X^\top\right) - I\right) y \\
&= y
\end{aligned}$$

Using Theorem 3.2.1, we observe that, for sufficiently large K and for sufficiently small step size $\eta < \max_i \frac{1}{\lambda_1(X D_i X^\top)}$, $\|I - \eta X D_i X^\top\| < 1 \ \forall \ i$. Thus, $\prod_{i=K-1}^{0}\left(I - \eta X D_i X^\top\right) \ \rightarrow \ 0$.

B.3 Proof of Proposition 3.3.2

We will prove this proposition by induction.

Base case: Here, we compute the first iteration, $K = 1$:

$$\begin{aligned} w_1 &= (-X^\top X)^{-1} \cdot \left(\prod_{i=0}^{0} \left(I - \eta X^\top X D_i \right) - I \right) X^\top y \\ &= (-X^\top X)^{-1} \cdot \left(\left(I - \eta X^\top X D_0 \right) - I \right) X^\top y = \eta D_0 X^\top y \end{aligned}$$

where we abuse the notation $\prod_{i=0}^{0} A_i = A_0$. This is the same result as in unfolding the recursion for $k = 0$, and assuming $w_0 = 0$.

Inductive case: Now, assume that, the above statement holds for $K - 1$,

$$w_{K-1} = \left(-X^\top X\right)^{-1} \cdot \left(\prod_{i=K-2}^{0} \left(I - \eta X^\top X D_i \right) - I \right) X^\top y.$$

Here, we use the convention $\prod_{i=\alpha}^{\beta} A_i = A_\alpha \cdot A_{\alpha-1} \cdots A_{\beta+1} \cdot A_\beta$, for integers $\alpha > \beta$. Then, the expression at the K-iteration satisfies:

$$\begin{aligned} w_K &= w_{K-1} - \eta D_{K-1} \nabla f(w_{K-1}) = w_{K-1} - \eta D_{K-1} X^\top (X w_{K-1} - y) \\ &= \left(I - \eta D_{K-1} X^\top X \right) w_{K-1} + \eta D_{K-1} X^\top y \\ &\overset{(i)}{=} \left(I - \eta D_{K-1} X^\top X \right) \left(-X^\top X \right)^{-1} \cdot \left(\prod_{i=K-2}^{0} \left(I - \eta X^\top X D_i \right) - I \right) X^\top y \\ &\quad + \eta D_{K-1} X^\top y \\ &= \left(\left(-X^\top X \right)^{-1} + \eta D_{K-1} \right) \cdot \left(\prod_{i=K-2}^{0} \left(I - \eta X^\top X D_i \right) - I \right) X^\top y \\ &\quad + \eta D_{K-1} X^\top y \end{aligned}$$

$$
\begin{aligned}
&= \left(\left(-X^\top X\right)^{-1} + \eta D_{K-1}\right)\left(\prod_{i=K-2}^{0}\left(I-\eta X^\top X D_i\right)\right)X^\top y \\
&\qquad\qquad - \left(\left(-X^\top X\right)^{-1} + \eta D_{K-1}\right)X^\top y + \eta D_{K-1}X^\top y \\
&= \left(\left(-X^\top X\right)^{-1} + \eta D_{K-1}\right)\left(\prod_{i=K-2}^{0}\left(I-\eta X^\top X D_i\right)\right)X^\top y - \left(-X^\top X\right)^{-1}X^\top y \\
&= \left(-X^\top X\right)^{-1}\left(I-\eta X^\top X D_{K-1}\right)\left(\prod_{i=K-2}^{0}\left(I-\eta X^\top X D_i\right)\right)X^\top y \\
&\quad - \left(-X^\top X\right)^{-1}X^\top y \\
&= \left(-X^\top X\right)^{-1}\left(\prod_{i=K-1}^{0}\left(I-\eta X^\top X D_i\right)\right)X^\top y - \left(-X^\top X\right)^{-1}X^\top y \\
&= \left(-X^\top X\right)^{-1}\left(\prod_{i=K-1}^{0}\left(I-\eta X^\top X D_i\right) - I\right)X^\top y
\end{aligned}
$$

where ***(i)*** is due to the inductive assumption. This completes the proof.

B.4 Proof of Proposition 3.4.1

Proposition B.4.1. *The set of eigenvalues for $\tilde{\boldsymbol{D}}(t)$ is identical to the set of eigenvalues of $\boldsymbol{D}(t)$.*

Proof. We know for each $t \geq 0$ that $\tilde{\boldsymbol{D}}(t) = \boldsymbol{V}^T\boldsymbol{D}(t)\boldsymbol{V}$. As the matrix $\boldsymbol{D}(t)$ is full rank for any $t \geq 0$ we can write its SVD as $\boldsymbol{D}(t) = \boldsymbol{V}_{D(t)}\boldsymbol{\Lambda}_{D(t)}\boldsymbol{V}_{D(t)}^T$. Substituting the SVD form in the expression we obtain $\tilde{\boldsymbol{D}}(t) = \boldsymbol{V}^T\boldsymbol{V}_{D(t)}\boldsymbol{\Lambda}_{D(t)}\boldsymbol{V}_{D(t)}^T\boldsymbol{V}$. But as both $\boldsymbol{V}_{D(t)}$ and $\boldsymbol{V}$ forms basis of $\mathbb{R}^d$ we have $\boldsymbol{V}_{D(t)}\boldsymbol{V}$ as another basis. Therefore, $\boldsymbol{V}_{D(t)}^T\boldsymbol{V}$is the matrix of eigenvectors and $\boldsymbol{\Lambda}_{D(t)}$ the eigenvalue matrix for $\tilde{\boldsymbol{D}}(t)$. This proves our claim. □

B.5 Proof of Proposition 3.4.2

The adaptive updates in (3.5) can be expressed in the spectral bases as

$$\begin{aligned}
&(\tilde{\boldsymbol{w}}(t+1) - \tilde{\boldsymbol{w}}(t)) - \eta\lambda\tilde{\boldsymbol{D}}(t)\tilde{\boldsymbol{w}}(t) \\
&= -\eta\boldsymbol{V}^T\boldsymbol{V}\tilde{\boldsymbol{D}}(t)\boldsymbol{V}^T\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{V}(\tilde{\boldsymbol{w}}(t) - \tilde{\boldsymbol{w}}^*) + \boldsymbol{U}\tilde{\boldsymbol{\zeta}}) \\
&= -\eta\tilde{\boldsymbol{D}}(t)\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{U}^T(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{V}(\tilde{\boldsymbol{w}}(t) - \tilde{\boldsymbol{w}}^*) + \boldsymbol{U}\tilde{\boldsymbol{\zeta}}) \\
&= -\eta\tilde{\boldsymbol{D}}(t)(\boldsymbol{\Lambda}^2(\tilde{\boldsymbol{w}}(t) - \tilde{\boldsymbol{w}}^*) + \boldsymbol{\Lambda}\tilde{\boldsymbol{\zeta}})
\end{aligned}$$

Therefore, the update in the spectral domain is represented as

$$\tilde{\boldsymbol{w}}(t+1) = \left(I - \eta\tilde{\boldsymbol{D}}(t)(\boldsymbol{\Lambda}^2 + \lambda I)\right)\tilde{\boldsymbol{w}}(t) + \eta\tilde{\boldsymbol{D}}(t)(\boldsymbol{\Lambda}^2\tilde{\boldsymbol{w}}^* + \boldsymbol{\Lambda}\tilde{\boldsymbol{\zeta}})$$

Using induction the closed form of the update in spectral domain can be obtained as

$$\begin{aligned}
\tilde{\boldsymbol{w}}(T) &= \prod_{i=0}^{T-1}\left(I - \eta\tilde{\boldsymbol{D}}(i)(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I})\right)\tilde{\boldsymbol{w}}(0) \\
&\quad + \sum_{i=0}^{T-1}\prod_{j=(i+1)}^{T-1}\left(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}(j)(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I})\right)\eta\tilde{\boldsymbol{D}}(i)(\boldsymbol{\Lambda}^2\tilde{\boldsymbol{w}}^* + \boldsymbol{\Lambda}\boldsymbol{\zeta}) \qquad \text{(B.1)}
\end{aligned}$$

The base case is true trivially. Given the expression is true for all the iterations upto $(T-1)$ we have

$$\begin{aligned}
&\tilde{\boldsymbol{w}}(T) = \left(I - \eta\tilde{\boldsymbol{D}}(T-1)(\boldsymbol{\Lambda}^2 + \lambda I)\right)\tilde{\boldsymbol{w}}(T-1) + \eta\tilde{\boldsymbol{D}}(T-1)(\boldsymbol{\Lambda}^2\tilde{\boldsymbol{w}}^* + \boldsymbol{\Lambda}\tilde{\boldsymbol{\zeta}}) \\
&= \left(I - \eta\tilde{\boldsymbol{D}}(T-1)(\boldsymbol{\Lambda}^2 + \lambda I)\right)\prod_{i=0}^{T-1}\left(I - \eta\tilde{\boldsymbol{D}}(i)(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I})\right)\tilde{\boldsymbol{w}}(0) \\
&+ \sum_{i=0}^{T-2}\left(I - \eta\tilde{\boldsymbol{D}}(T-1)(\boldsymbol{\Lambda}^2 + \lambda I)\right)\prod_{j=(i+1)}^{T-2}\left(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}(j)(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I})\right)\eta\tilde{\boldsymbol{D}}(i)(\boldsymbol{\Lambda}^2\tilde{\boldsymbol{w}}^* + \boldsymbol{\Lambda}\boldsymbol{\zeta}) \\
&+ \eta\tilde{\boldsymbol{D}}(T-1)(\boldsymbol{\Lambda}^2\tilde{\boldsymbol{w}}^* + \boldsymbol{\Lambda}\tilde{\boldsymbol{\zeta}}).
\end{aligned}$$

This completes the proof.

B.6 Proof of Proposition 3.4.3

From Theorem 3.2.1, we know that a sufficient condition for the convergence under update (3.5) is

$$\sup_{t\geq 1} |\lambda \left(I - \eta \tilde{\boldsymbol{D}}(t)(\boldsymbol{\Lambda}^2 + \lambda \boldsymbol{I}) \right) | < 1.$$

Thus, for $\lambda > 0$, the dynamics converges to a bounded weight vector for any $\eta \in \left(0, 2\left(\lambda_{max}(\boldsymbol{D}(t))(\lambda^2_{max}(\boldsymbol{X}) + \lambda)\right)^{-1}\right)$.

We now characterize the fixed point of the dynamics in (3.5). When the convergence happens, for any fixed point $\hat{\boldsymbol{w}}$ of the updates in (3.5)

$$\boldsymbol{D}(t)\left(\lambda\hat{\boldsymbol{w}} + \boldsymbol{X}^T\boldsymbol{X}(\hat{\boldsymbol{w}} - \boldsymbol{w}^*) - \boldsymbol{X}^T\boldsymbol{w}\right) = \boldsymbol{0}.$$

Because, $\inf_t rank(\boldsymbol{D}(t)) = d$ (full rank) we must have

$$\lambda\hat{\boldsymbol{w}} + \boldsymbol{X}^T\boldsymbol{X}(\hat{\boldsymbol{w}} - \boldsymbol{w}^*) - \boldsymbol{X}^T\boldsymbol{w} = \boldsymbol{0}$$

. Expanding the L.H.S. in terms of the SVD of the data matrix we obtain,

$$\sum_{r=1}^{d} \lambda\tilde{\boldsymbol{w}}_r\boldsymbol{v_r} + \sum_{r=1}^{R}\left(\lambda_r^2\tilde{\boldsymbol{w}}_r - \lambda_r^2\tilde{\boldsymbol{w}}_r^* - \lambda_r\tilde{\boldsymbol{\zeta}}_r\right)\boldsymbol{v}_r = \boldsymbol{0}.$$

Therefore, for $\lambda \geq 0$ (holds for both regularized and unregularized) we have $\boldsymbol{v_r}^T\hat{\boldsymbol{w}} = \frac{\lambda_r^2\tilde{\boldsymbol{w}}_r^* + \lambda_r\tilde{\boldsymbol{\zeta}}_r}{\lambda + \lambda_r^2}$ for $r \leq R$. Further, for $\lambda > 0$, $\boldsymbol{v_r}^T\hat{\boldsymbol{w}} = 0$ for $r \geq (R+1)$.

B.7 Proof of Lemma B.7.1

Using the above structure we obtain the following lemma concerning the closed form expression of the iterates. Let us define for any matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$

and any vector $\boldsymbol{b} \in \mathbb{R}^d$:

$$\begin{aligned}
&\boldsymbol{A}_{(1)} = \{\boldsymbol{A}_{ij} : 1 \leq i, j \leq R\},\\
&\boldsymbol{A}_{(2)} = \{\boldsymbol{A}_{ij} : R+1 \leq i \leq d, 1 \leq j \leq R\},\\
&\boldsymbol{b}_{(1)} = \{\boldsymbol{b}_i : 1 \leq i \leq R\},\\
&\boldsymbol{b}_{(2)} = \{\boldsymbol{b}_i : R+1 \leq i \leq d\},
\end{aligned}$$

where R is the rank of the data matrix $\boldsymbol{D}$ and d is the dimension of the data.

Lemma B.7.1. *If $\boldsymbol{D}(t)$ is full rank for all $t \geq 0$ and regularizer $\lambda = 0$, then for any $T \geq 0$, the closed form of the iterate $\tilde{\boldsymbol{w}}(T)$ admits the following expression:*

$$\begin{aligned}
&\tilde{\boldsymbol{w}}_{(1)}(T) = \boldsymbol{A}(T-1, 0)\tilde{\boldsymbol{w}}_1(0)\\
&+\sum_{i=0}^{T-1} \boldsymbol{A}(T-1, i+1)\eta\tilde{\boldsymbol{D}}_{(1)}(i)\boldsymbol{\Lambda}^2_{(1)}(\boldsymbol{w}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}),\\
&\tilde{\boldsymbol{w}}_{(2)}(T) = \boldsymbol{B}(T-1, 0)\tilde{\boldsymbol{w}}_{(1)}(0) + \tilde{\boldsymbol{w}}_{(2)}(0)\\
&+\sum_{i=0}^{T-1} \eta\left(\boldsymbol{B}(T-1, i+1)\tilde{\boldsymbol{D}}_{(1)}(i)+\tilde{\boldsymbol{D}}_{(2)}(i)\right) \times\\
&\times \boldsymbol{\Lambda}^2_{(1)}(\boldsymbol{w}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}),
\end{aligned}$$

where for all $t_2 \geq t_1 \geq 0$,

$$\begin{aligned}
\boldsymbol{A}(t_2, t_1) &= \prod_{i=t_1}^{t_2}\left(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}_{(1)}(i)\boldsymbol{\Lambda}^2_{(1)}\right)\\
\boldsymbol{B}(t_2, t_1) &= -\eta\tilde{\boldsymbol{D}}_{(2)}(t_1)\boldsymbol{\Lambda}^2_{(1)}\\
&\quad -\eta\sum_{i=t_1+1}^{t_2}\tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}^2_{(1)}\boldsymbol{A}(i-1, t_1)
\end{aligned}$$

In the above lemma, the vector $\tilde{\boldsymbol{w}}_{(1)}(T)$ represents the in-span component of the iterate, where as $\tilde{\boldsymbol{w}}_{(2)}(T)$ represents the out-of-span component of the iterate. We make an important observation in the complex expression in Lemma B.7.1 that for appropriate choice of η, we have $\max|\lambda(\boldsymbol{A}(t_2, t_1))| < 1$ for all $t_2 \geq t_1 \geq 0$. This is true because, even though $\lambda_{min}(\boldsymbol{\Lambda}^2) = 0$, when only the $R \times R$ submatrix $\boldsymbol{\Lambda}_{(1)}$ is considered, we have $\lambda_{min}(\boldsymbol{\Lambda}^2_{(1)}) > 0$. Using this result we prove the convergence of in-span component. We will use the following equations regarding the product two specific block matrices.

$$\begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{B}_1 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_2 & \boldsymbol{0} \\ \boldsymbol{B}_2 & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_1\boldsymbol{A}_2 & \boldsymbol{0} \\ \boldsymbol{B}_1\boldsymbol{A}_2 + \boldsymbol{B}_2 & \boldsymbol{I} \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{B}_1 & \boldsymbol{C}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_1\boldsymbol{A}_2 & \boldsymbol{0} \\ \boldsymbol{B}_1\boldsymbol{A}_2 & \boldsymbol{0} \end{bmatrix} \tag{B.2}$$

Firstly, we obtain the block structure shown in the paper.

$$\begin{aligned} \left(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}(i)\boldsymbol{\Lambda}^2\right) &= \boldsymbol{I} - \eta \begin{bmatrix} \tilde{\boldsymbol{D}}_{(1)}(i) & \tilde{\boldsymbol{D}}_{(2)}(i) \\ \tilde{\boldsymbol{D}}_{(2)}(i) & \tilde{\boldsymbol{D}}_{(3)}(i) \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}^2_{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \\ &= \begin{bmatrix} \left(I - \eta\tilde{\boldsymbol{D}}_{(1)}(i)\boldsymbol{\Lambda}^2[1]\right) & \boldsymbol{0}_{R\times(d-R)} \\ -\eta\tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}^2[1] & \boldsymbol{I}_{(d-R)\times(d-R)} \end{bmatrix} \end{aligned}$$

The block structure is maintained for the product of these matrices, i.e. for all $t_2 \geq t_1 \geq 0$,

$$\prod_{i=t_1}^{t_2} \left(I - \eta\tilde{\boldsymbol{D}}(i)\boldsymbol{\Lambda}^2\right) = \begin{bmatrix} \boldsymbol{A}(t_2, t_1) & \boldsymbol{0}_{R\times(d-R)} \\ \boldsymbol{B}(t_2, t_1) & \boldsymbol{I}_{(d-R)\times(d-R)} \end{bmatrix},$$

$$\begin{aligned} \boldsymbol{A}(t_2, t_1) = \prod_{i=t_1}^{t_2} \left(I - \eta\tilde{\boldsymbol{D}}_{(1)}(i)\boldsymbol{\Lambda}^2_{(1)}\right), &\, \boldsymbol{B}(t_2, t_1) \\ &= -\eta\tilde{\boldsymbol{D}}_{(2)}(t_1)\boldsymbol{\Lambda}^2_{(1)} - \eta \sum_{i=t_1+1}^{t_2} \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}^2_{(1)}\boldsymbol{A}(i-1, t_1) \end{aligned}$$

This can be shown easily using induction and using Equation (B.2).

Substituting these results in the closed form of the iterates in proposition 3.4.2 we obtain

$$\tilde{\boldsymbol{w}}(T) = \begin{bmatrix} \boldsymbol{A}(T-1,0)\tilde{\boldsymbol{w}}_1(0) \\ \boldsymbol{B}(T-1,0)\tilde{\boldsymbol{w}}_1(0) + \tilde{\boldsymbol{w}}_{(2)}(0) \end{bmatrix} + \sum_{i=0}^{T-1} \begin{bmatrix} \boldsymbol{A}(T-1,i+1) & \boldsymbol{0} \\ \boldsymbol{B}(T-1,i+1) & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \eta\tilde{\boldsymbol{D}}_1(i)\boldsymbol{\Lambda}_1^2(\boldsymbol{w}_1^* + \boldsymbol{\Lambda}_1^{-1}\boldsymbol{\zeta}_1) \\ \eta\tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_1^2(\boldsymbol{w}_1^* + \boldsymbol{\Lambda}_1^{-1}\boldsymbol{\zeta}_1) \end{bmatrix}$$

In-span Component: Therefore, the component of in the span of data is $\tilde{\boldsymbol{w}}(T)$

$$\tilde{\boldsymbol{w}}_{(1)}(T) = \boldsymbol{A}(T-1,0)\tilde{\boldsymbol{w}}_1(0) + \sum_{i=0}^{T-1} \boldsymbol{A}(T-1,i+1)\eta\tilde{\boldsymbol{D}}_1(i)\boldsymbol{\Lambda}_{(1)}^2(\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}),$$

Similar to the regularized case, we have for any

$$\eta \in \left(0, 2\left(\lambda_{max}(\boldsymbol{D}(t))(\lambda_{max}^2(\boldsymbol{X}))\right)^{-1}\right)$$

the in-span component converges. Further, from the fixed point argument we know that $\boldsymbol{v}_r^T\hat{\boldsymbol{w}} = \tilde{\boldsymbol{w}}_r^* + \lambda_r^{-1}\tilde{\boldsymbol{\zeta}}_r$.

Out-of-span Component: The component outside the span of the data is

$$\tilde{\boldsymbol{w}}_{(2)}(T) = \boldsymbol{B}(T-1,0)\tilde{\boldsymbol{w}}_{(1)}(0) + \tilde{\boldsymbol{w}}_{(2)}(0) + \sum_{i=0}^{T-1} \eta\left(\boldsymbol{B}(T-1,i+1)\tilde{\boldsymbol{D}}_{(1)}(i) + \tilde{\boldsymbol{D}}_{(2)}(i)\right)\boldsymbol{\Lambda}_{(1)}^2(\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}).$$

B.8 Proof of Theorem 3.4.5

The convergence of the in-span component follows similar to the regularized case. In particular, we observe

$$\lambda_{max}(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}_{(1)}(t)\boldsymbol{\Lambda}_{(1)}^2) \leq 1 - \eta\lambda_{min}(\tilde{\boldsymbol{D}}_{(1)}(t))\lambda_{min}(\boldsymbol{\Lambda}_{(1)}^2) < 1.$$

The last inequality is true as 1) $\lambda_{min}(\tilde{\boldsymbol{D}}_{(1)}(t)) > 0$ due to the positive definiteness of the matrix $\tilde{\boldsymbol{D}}(t)$, and 2) $\lambda_{min}(\boldsymbol{\Lambda}_{(1)}^2) > 0$ as it considers only the in-span component (i.e. the top-left $R \times R$ sub-matrix of $\boldsymbol{\Lambda}$). On the other hand, we have $\lambda_{min}(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}_{(1)}(t)\boldsymbol{\Lambda}_{(1)}^2) \geq 1 - \eta\lambda_{max}(\tilde{\boldsymbol{D}}_{(1)}(t))\lambda_{max}(\boldsymbol{\Lambda}_{(1)}^2)$. Therefore, we obtain $\lambda_{min}(\boldsymbol{I} - \eta\tilde{\boldsymbol{D}}_{(1)}(t)\boldsymbol{\Lambda}_{(1)}^2) > -1$ for any $0 < \eta < 2/(\lambda_{max}(\tilde{\boldsymbol{D}}_{(1)}(t))\lambda_{max}(\boldsymbol{\Lambda}_{(1)}^2))$.

As $\tilde{\boldsymbol{D}}_{(1)}(t)$ is a principal sub-matrix of $\tilde{\boldsymbol{D}}(t)$ for each $t \geq 0$, we have from Cauchy Interlacing Theorem $\lambda_{max}(\tilde{\boldsymbol{D}}_{(1)}(t)) \leq \lambda_{max}(\tilde{\boldsymbol{D}}(t)) = \lambda_{max}(\boldsymbol{D}(t))$. The last equality is due to Proposition 3.4.1. The characterization of the fixed point follows the same argument as Proposition 3.4.3.

To prove the second part, we further simplify the out-of-span component using exchange of summation (for finite T). Here, we use the convention

$\boldsymbol{A}(t_1, t_2) = \boldsymbol{I}$ for any $t_1 < t_2$.

$$\begin{aligned}
&\tilde{\boldsymbol{w}}_{(2)}(T) - \big(\boldsymbol{B}(T{-}1,0)\tilde{\boldsymbol{w}}_{(1)}(0) + \tilde{\boldsymbol{w}}_{(2)}(0)\big) \\
&= \sum_{i=0}^{T-1} \eta \Big(\tilde{\boldsymbol{D}}_{(2)}(i) - \eta \tilde{\boldsymbol{D}}_{(2)}(i+1)\boldsymbol{\Lambda}_{(1)}^2 \tilde{\boldsymbol{D}}_{(1)}(i) - \\
&\qquad \eta \sum_{j=i+2}^{T-1} \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_{(1)}^2 \boldsymbol{A}(j{-}1, i+1)\tilde{\boldsymbol{D}}_{(1)}(i)\Big) \boldsymbol{\Lambda}_{(1)}^2(\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}). \\
&= \left(\sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i) - \eta^2 \sum_{i=0}^{T-1}\sum_{j=i+1}^{T-1} \tilde{\boldsymbol{D}}_{(2)}(j)\boldsymbol{\Lambda}_{(1)}^2 \boldsymbol{A}(j{-}1, i+1)\tilde{\boldsymbol{D}}_{(1)}(i)\right) \boldsymbol{\Lambda}_{(1)}^2(\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}). \\
&= \left(\sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i) \left(\boldsymbol{I} - \eta \boldsymbol{\Lambda}_{(1)}^2 \sum_{j=0}^{i-1} \boldsymbol{A}(i{-}1, j+1)\tilde{\boldsymbol{D}}_{(1)}(j)\right)\right) \boldsymbol{\Lambda}_{(1)}^2(\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}), \\
&= \left(\sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i) \Big(\boldsymbol{\Lambda}_{(1)}^2(\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}) - \boldsymbol{\Lambda}_{(1)}^2 \tilde{\boldsymbol{w}}_{(1)}(i) + \boldsymbol{\Lambda}_{(1)}^2 \boldsymbol{A}(i{-}1, 0)\tilde{\boldsymbol{w}}_1(0)\Big)\right), \\
&= \sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_{(1)}^2 \Big((\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i)\Big) + \sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_{(1)}^2 \boldsymbol{A}(i{-}1, 0)\tilde{\boldsymbol{w}}_1(0).
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
&\|\tilde{\boldsymbol{w}}_{(2)}(T) - \tilde{\boldsymbol{w}}_{(2)}(0)\|_2 \\
&\le \sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_{(1)}^2 \Big((\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i)\Big) \\
&\qquad + \left(\boldsymbol{B}(T{-}1, 0) + \sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_{(1)}^2 \boldsymbol{A}(i{-}1, 0)\right) \tilde{\boldsymbol{w}}_1(0) \\
&= \sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i)\boldsymbol{\Lambda}_{(1)}^2 \Big((\boldsymbol{w}_{(1)}^* + \boldsymbol{\Lambda}_{(1)}^{-1}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i)\Big) - \eta \tilde{\boldsymbol{D}}_{(2)}(0)\boldsymbol{\Lambda}_{(1)}^2 \tilde{\boldsymbol{w}}_1(0)
\end{aligned}$$

We have $|\lambda|_{\max} = \sup_t |\lambda_{\max}(I - \eta \tilde{\boldsymbol{D}_1}(t)\boldsymbol{\Lambda}_{(1)}^2)| < 1$ due to appropriate choice of η. Also, by assumption of the theorem we have for some $\alpha \ge 0$,

$\beta \geq 0$, $\alpha + \beta > 1$, for some universal constants $0 < c_{conv}, c_\lambda < \infty$, and for all $t \geq 0$:

(i) the out-of-span pre-conditioner matrix decaying as $O(1/t^\alpha)$ for some $\alpha \geq 0$, i.e. $|\lambda_{\max}(\tilde{\boldsymbol{D}}_{(2)}(i))| = \frac{c_\lambda}{(t+1)^\alpha}$, and

(ii) the convergence rate of the in-span component is $O(1/t^\beta)$ with iteration t for some $\beta > 0$, i.e. $\|(\boldsymbol{w}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i)\|_2 \leq \frac{c_{conv}}{(t+1)^\beta}$.

For the first term we have,

$$\begin{aligned}
&\| \sum_{i=0}^{T-1} \eta \tilde{\boldsymbol{D}}_{(2)}(i) \boldsymbol{\Lambda}^2_{(1)} \left((\boldsymbol{w}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i) \right) \|_2 \\
&\leq c_\lambda \sum_{i=0}^{T-1} \lambda_{\max}(\tilde{\boldsymbol{D}}_{(2)}(i)) \|(\boldsymbol{w}^*_{(1)} + \boldsymbol{\Lambda}^{-1}_{(1)}\boldsymbol{\zeta}_{(1)}) - \tilde{\boldsymbol{w}}_{(1)}(i)\|_2 \\
&\leq c_\lambda c_{conv} \sum_{i=1}^{T-1} \tfrac{1}{(i+1)^{(\alpha+\beta)}} \leq \tfrac{c_\lambda c_{conv}}{\alpha+\beta-1} \left(1 - \tfrac{1}{(i+1)^{(\alpha+\beta-1)}}\right).
\end{aligned}$$

Therefore, the first term saturates to a value at most $\frac{c_\lambda c_{conv}}{\alpha+\beta-1}$.

For the second term we have,

$$\|\eta \tilde{\boldsymbol{D}}_{(2)}(0) \boldsymbol{\Lambda}^2_{(1)} \tilde{\boldsymbol{w}}_1(0)\|_2 \leq \eta \lambda_{\max}(\tilde{\boldsymbol{D}}_{(2)}(0)) \lambda^2_{\max}(\boldsymbol{\Lambda}_{(1)}) \|\tilde{\boldsymbol{w}}_1(0)\|_2$$

B.9 Proof of Proposition B.9.1

Proposition B.9.1. *The following pre-conditioner matrices have $\tilde{\boldsymbol{D}}_{(2)}(t) = \boldsymbol{0}$.*

1. *$\boldsymbol{D}(t) = \boldsymbol{I}$, i.e. gradient descent,*

2. *$\boldsymbol{D}(t) = (\boldsymbol{X}^T\boldsymbol{X} + \epsilon\boldsymbol{I})^{-1}$ for all $t \geq 0$.*

We have $(\boldsymbol{X}^T\boldsymbol{X} + \epsilon \boldsymbol{I})^{-1} = \sum_{r=1}^{R}(\lambda_r^2 + \epsilon)^{-1}\boldsymbol{v}_r\boldsymbol{v}_r^T + \sum_{r=R+1}^{d} \epsilon^{-1}\boldsymbol{v}_r\boldsymbol{v}_r^T$. Further, $\boldsymbol{I} = \sum_{r=1}^{d} bsv_r\boldsymbol{v}_r^T$. So the proposition is true.

B.10 Proof of Lemma 3.5.1

Proposition B.10.1. *Suppose $\boldsymbol{X}^\top\boldsymbol{y}$ has no zero components. Define $\boldsymbol{D} = \boldsymbol{diag}(|\boldsymbol{X}^\top\boldsymbol{y}|^3)$ and assume there exists a scalar c such that $\boldsymbol{X}\boldsymbol{D}^{-1}\boldsymbol{sign}(\boldsymbol{X}^\top\boldsymbol{y}) = c\boldsymbol{y}$. Then, when initialized at 0, the AdaGrad variant in (3.9) converges to the unique solution $\boldsymbol{w} \propto \boldsymbol{D}^{-1}\boldsymbol{sign}(\boldsymbol{X}^\top\boldsymbol{y})$.*

We will prove this using induction. Let $Q = diag(|X^Ty|)$ We will show that

$$w_k = \lambda_k Q^{-1} sign(X^Ty)$$

for some λ_k. $w_0 = 0$ is satsified for $\lambda_0 = 0$ and so the base case is trivially true.

$$\begin{aligned} g_k &= X^T(Xw_k - y) \\ &= \lambda_k X^TXQ^{-1}sign(X^Ty) - X^Ty \\ &= (\lambda_k c - 1)X^Ty \end{aligned}$$

where the last inequality follows from $w_k = \lambda_k Q^{-1} sign(X^Ty)$.3

$$H_k = diag(\sum_{s=1}^{n} g_s \cdot g_s) = \nu_k diag(|X^Ty|^2) = \nu_k Q^2$$

$$\begin{aligned}
w_{k+1} &= w_k - \alpha_k H_k^{-1} X^T (X w_k - y) \\
&= w_k - \alpha_k H_k^{-1} X^T X w_k + \alpha_k H_k^{-1} X^T y \\
&= \lambda_k Q^{-1} sign(X^T y) - \lambda_k \alpha_k H_k^{-1} X^T X Q^{-1} X^T y + \alpha_k H_k^{-1} X^T y \\
&= \lambda_k Q^{-1} sign(X^T y) - \lambda_k \alpha_k c H_k^{-1} X^T y + \alpha_k H_k^{-1} X^T y \\
&= \left(\lambda_k - \frac{\lambda_k \alpha_k c}{\nu_k} + \frac{\alpha_k}{\nu_k} \right) Q^{-1} sign(X^T y) \\
&= \lambda_{k+1} Q^{-1} sign(X^T y)
\end{aligned}$$

B.11 More details and experiments for the counter-example

The simulation is completed as follows: For each setting (n, p, J), we generate 100 different instances for (X, y), and for each instance we compute the solutions from gradient descent, AdaGrad variant and Adam (RMSprop is included in the Appendix) and the minimum norm solution w_{mn}. In the appendix, we have the above table with the Adagrad variant that normalizes the final solution $\widehat{w}$ (Table **??**) before calculating the distance w.r.t. the minimum norm solution: we observed that this step did not improve or worsen the performance, compared to the unnormalized solution. *This further indicates that there is an infinite collection of solutions –with different magnitudes– that lead to better performance than plain gradient descent; thus our findings are not a pathological example where adaptive methods work better.*

We record $\|\widehat{w} - w_{\text{mn}}\|_2$, where $\widehat{w}$ represents the corresponding solutions obtained by the algorithms in the comparison list. For each (X, y) instance, we further generate $\{y_i^{\text{test}}, x_i^{\text{test}}\}_{i=1}^{100}$, and we evaluate the performance of both models on predicting y_i^{test}, $\forall i$.

Table 3.4 shows that gradient descent converges to the minimum norm solution, in contrast to the adaptive methods. This justifies the fact that the adaptive gradient methods (including the proposed adagrad variant) converge to a different solution than the minimum norm solution. Nevertheless, the accuracy on *unseen* data is higher in the adaptive methods (both our proposed AdaGrad variant and in most instances, Adam), than the plain gradient descent, when ℓ is small: the adaptive method successfully identifies the correct class, while gradient descent only predicts one class (the positive class; this is justified by the fact that the accuracy obtained is approximately close to p, as n increases). We first provide the same table in Table 3.4 but with unnormalized values for distances with respect to Adagrad variant.

Here, we provide further results on the counterexample in Subsubsection 3.5.1.1. Tables **??** and B.1 contains results for $J = 10$: the purpose of these tables is to show that even if we change the memory use footprint of the AdaGrad variant—by storing fewer or more gradients to compute D_k in (3.9)—the results are the same: the AdaGrad variant consistently converges to a solution different than the minimum norm solution, while being more accurate than the latter for small values of ℓ (*i.e.*, smaller margin between the two classes).

Plain gradient descent methods provably need to rely on the first ele-

			GD	ADAVAR	ADAM
$n = 10$	$\ell = 1/32$	Acc. (%)	63	**100**	91
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$1.015 \cdot 10^{-16}$	0.9911	0.1007
	$\ell = 1/16$	Acc. (%)	53	**100**	87
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$1.7401 \cdot 10^{-16}$	0.9263	0.0864
	$\ell = 1/8$	Acc. (%)	58	**99**	84
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$4.08 \cdot 10^{-16}$	0.8179	0.0764
$n = 50$	$\ell = 1/32$	Acc. (%)	77	**100**	88
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$4.729 \cdot 10^{-15}$	0.8893	0.0271
	$\ell = 1/16$	Acc. (%)	80	**100**	89
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$6.9197 \cdot 10^{-15}$	0.7929	0.06281
	$\ell = 1/8$	Acc. (%)	91	**100**	89
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$9.7170 \cdot 10^{-15}$	0.6639	0.1767
$n = 100$	$\ell = 1/32$	Acc. (%)	85	**100**	95
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$4.975 \cdot 10^{-9}$	0.8463	0.0344
	$\ell = 1/16$	Acc. (%)	83	**100**	76
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$2.5420 \cdot 10^{-9}$	0.7217	0.1020
	$\ell = 1/8$	Acc. (%)	**100**	**100**	90
		$\|\widehat{w} - w_{\mathrm{mn}}\|_2$	$1.5572 \cdot 10^{-11}$	0.6289	0.3306

Table B.1: Prediction accuracy and distances from the minimum norm solution for plain gradient descent and adaptive gradient descent methods. We set $p = 7/8$ and $J = 10$, as in the main text. The adaptive method uses D_k according to (3.9). The distances shown are median values out of 100 different realizations for each setting; the accuracies are obtained by testing 10^4 predictions on unseen data.

ments to decide; using the same rule for adaptive methods[1]. The remaining subsection considers the case where we decide based on the $y = \texttt{sign}(x^\top w)$ rule, where w is the complete learned model. As we show empirically, more often than not adaptive methods outperform plain gradient methods.

Observing the performance of various optimization techniques for different values of n, p and ℓ, we observed that the best performances are obtained when the dataset is highly imbalanced irrespective of the optimization algorithm chosen. When the data is (almost) balanced, it is difficult to comment on how the performance of these algorithms is affected by variations in the levels ℓ and probability p.

B.12 Deep Learning

In this section, we will extend the experiments to over-parameterized and under-parameterized neural networks without regularization. We begin with a detailed description of the datasets and the architectures we use along with comprehensive set of experiments with hyperparameter tuning.

MNIST dataset and the M1 architecture. Each experiment for M1 is simulated over 50 epochs and 10 runs for both under- and over-parameterized settings. Both the MNIST architectures consisted of two convolutional layers

[1]We note that using only the three elements in adaptive methods is not backed up by theory since it assumes that the training and test datasets have no overlap. We include this in comparison for completeness.

Name	Network type	Dataset
M1-UP	Shallow CNN + FFN	MNIST
M1-OP	Shallow CNN + FFN	MNIST
C1-UP	Shallow CNN + FFN	CIFAR-10
C1-OP	ResNet18	CIFAR-10
C2-OP	PreActResNet18	CIFAR-100
C3-OP	MobileNet	CIFAR-100
C4-OP	MobileNetV2	CIFAR-100
C5-OP	GoogleNet	CIFAR-100

Table B.2: Summary of the datasets and the architectures used for experiments. CNN stands for convolutional neural network, FF stands for feed forward network. More details are given in the main text.

(the second one with dropouts [79]) followed by two fully connected layers. The primary difference between the M1-OP ($\sim$ 73K parameters) and M1-UP ($\sim$ 21K parameters) architectures was the number of channels in the convolutional networks and # of nodes in the last fully connected hidden layer.

Figure 3.3, left two columns, reports the results over 10 Monte-Carlo realizations. Top row corresponds to the M1-UP case; bottom row to the M1-OP case. We plot both training errors and the accuracy results on unseen data. For the M1-UP case, despite the grid search, observe that AdaGrad (and its variant) do not perform as well as the rest of the algorithms. Nevertheless, adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants, supporting our conjecture that each algorithm requires a different configuration, but still can converge to a good local point; also that adaptive methods require the same (if not more) tuning. For the M1-OP case, SGD momentum performs less favorably compared to plain SGD, and we conjecture that this is due to non-optimal tuning. In this case, all adaptive methods perform similarly to SGD.

CIFAR10 dataset and the C1 architecture. For C1, C1-UP is trained over 350 epochs, while C1-OP was trained over 200 epochs. The under-parameterized setting is on-purpose tweaked to ensure that we have fewer parameters than examples ($\sim$ 43K parameters), and slightly deviates from [158]; our generalization guarantees ($\sim$ 76%) are in conjunction with the attained test accuracy levels. Similarly, for the C1-OP case, we implement a Resnet [80] + dropout architecture ($\sim$ 0.25 million parameters) Adam and RMSProp achieves the best performance than their non-adaptive counterparts for both the under-parameterized and over-parameterized settings.

Figure 3.3, right panel, follows the same pattern with the MNIST data; it reports the results over 10 Monte-Carlo realizations. Again, we observe that AdaGrad methods do not perform as well as the rest of the algorithms. Nevertheless, adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants. Further experiments on CIFAR-100 for different architecture are provided in the Appendix.

CIFAR100 and other deep architectures (C{2-5}-OP). In this experiment, we focus only on the over-parameterized case: DNNs are usually designed over-parameterized in practice, with ever growing number of layers, and, eventually, a larger number of parameters [81]. We again completed

10 runs for each of the set up we considered. C2-OP corresponds to PreActResNet18 from [5], C3-OP corresponds to MobileNet from [6], C4-OP is MobileNetV2 from [7], and C5-OP is GoogleNet from [82]. The results are depicted in Figure 3.4. After a similar hyper-parameter tuning phase, we selected the best choices among the parameters tested. The results show no clear winner once again, which overall support our claims: *the superiority depends on the problem/data at hand; also, all algorithms require fine tuning to achieve their best performance.* We note that a more comprehensive reasoning requires multiple runs for each network, as other hyper-parameters (such as initialization) might play significant role in closing the gap between different algorithms.

An important observation of Figure 3.4 comes from the bottom row of the panel. There, we plot the Euclidean norm $\|\cdot\|_2$ of all the trainable parameters of the corresponding neural network. While such a norm could be considered arbitrary (e.g., someone could argue other types of norms to make more sense, like the spectral norm of layer), we use the Euclidean norm as ***i)*** it follows the narrative of algorithms in linear regression, where plain gradient descent algorithms choose minimum ℓ_2-norm solutions, and ***ii)*** there is recent work that purposely regularizes training algorithms towards minimum norm solutions [83].

Our findings support our claims: in particular, for the case of MobileNet and MobileNetV2, Adam, an adaptive method, converges to a solution that has at least as good generalization as plain gradient methods, while having $2\times$

larger ℓ_2-norm weights. However, this may not always be the trend: in Figure 3.4, left panel, the plain gradient descent models for the PreActResNet18 architecture [5] show slightly better performance, while preserving low weight norm. The same holds also for the case of GoogleNet; see Figure 3.4, right panel.

B.12.0.1 Hyperparameter tuning

Both for adaptive and non-adaptive methods, the step size and momentum parameters are key for favorable performance, as also concluded in [71]. Default values were used for the remaining parameters. The step size was tuned over an exponentially-spaced set $\{0.0001, 0.001, 0.01, 0.1, 1\}$, while the momentum parameter was tuned over the values of $\{0, 0.1, 0.25, 0.5, 0.75, 0.9\}$. We observed that step sizes and momentum values smaller/bigger than these sets gave worse results. *Yet, we note that a better step size could be found between the values of the exponentially-spaced set.* The decay models were similar to the ones used in [71]: no decay and fixed decay. We used fixed decay in the over-parameterized cases, using the `StepLR` implementation in `pytorch`. We experimented with both the decay rate and the decay step in order to ensure fair comparisons with results in [71].

B.12.0.2 Results

Our main observation is that, both in under- or over-parameterized cases, adaptive and non-adaptive methods converge to solutions with similar testing accuracy: the superiority of simple or adaptive methods depends on the problem/data at hand. Further, as already pointed in [71], adaptive methods often require similar parameter tuning. Most of the experiments involve using readily available code from GitHub repositories. Since increasing/decreasing batch-size affects the convergence [84], all the experiments were simulated on identical batch-sizes. Finally, our goal is to show performance results in the purest algorithmic setups: often, our tests did not achieve state of the art performance.

Overall, despite not necessarily converging to the same solution as gradient descent, adaptive methods generalize as well as their non-adaptive counterparts. In M1 and C1-UP settings, we compute standard deviations from all Monte Carlo instances, and plot them with the learning curves (shown in shaded colors is the one-apart standard deviation plots; best illustrated in electronic form). For the cases of C{1-5}-OP, we also show the weight norms of the solutions (as in Euclidean distance $\|\cdot\|_2$ of all the trainable weights in the network). Such measure has been in used in practice [83], as a regularization to find minimum Euclidean norm solutions, inspired by the results from support vector machines [45].

We observe that adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants, supporting our conjecture that each algorithm requires a different configuration, but still can converge to a good local point; also that adaptive methods require the same (if not more) tuning. Again, we observe that AdaGrad methods do not perform as well as the rest of the algorithms. Nevertheless, adaptive methods (such as Adam and RMSProp) perform similarly to simple SGD variants. Further experiments on CIFAR-100 for different architecture are provided in the Appendix.

Appendix C

Appendix

C.1 Additional Results for Section 3

The following lemma provides upper bounds on the expected gradient of the worst-possible MKL-SGD solution that lies in a ball around $\boldsymbol{w}^*$. Simultaneously satisfying the following bound with the one in Lemma 4.3.4 may lead to an infeasible set of ϵ and N'. And thus we use Lemma 4.3.5 in conjunction with 4.3.4.

Lemma C.1.1. *Let us assume that MKL-SGD converges to $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}$. For any $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} \in \mathcal{B}_r(\boldsymbol{w}^*)$ that satisfies assumptions N1, N2, A4 and A5, there exists $N' \geq N$ and $\epsilon' \leq \epsilon$ such that,*

$$\begin{aligned}&\| \sum_{i \notin \mathbb{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) \| \\ &\qquad \leq \min \left\{ (1-\epsilon^k) L \| \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* \|, \epsilon^k G(\boldsymbol{w}) \right\}\end{aligned}$$

The proof for lemma 2 can be found in the Appendix Section C.2.7

C.2 Proofs and supporting lemmas

C.2.1 Proof of Lemma 4.3.1

Proof. $\widetilde{F}(\boldsymbol{w}) = \sum_i p_{m_i(\boldsymbol{w})}(\boldsymbol{w})$. Let us fix a $\boldsymbol{w}$ such that $p_i = p_i(\boldsymbol{w})$. We know that for any p_i, $\sum_i p_i f_i(\boldsymbol{w})$ is strongly convex in $\boldsymbol{w}$ with parameter $\lambda_{\boldsymbol{w}}$. This implies

$$\nabla \widetilde{F}(\boldsymbol{w})^\top (\boldsymbol{w} - \boldsymbol{w}^*) \geq \lambda \|\boldsymbol{w} - \boldsymbol{w}^*\|^2$$

□

C.2.2 Proof of Theorem 4.3.2

Proof. By the definition of the noiseless framework, $\boldsymbol{w}^*$ is the unique optimum of $F(\boldsymbol{w})$ and lies in the optimal set of each $f_i(.)$. We will prove this theorem by contradiction. Assume there exists some $\hat{\boldsymbol{w}} \neq \boldsymbol{w}^*$ that also satisfies optimum of $\nabla \widetilde{F}(\hat{\boldsymbol{w}}) = \boldsymbol{0}$. At $\hat{\boldsymbol{w}}$, we have $0 =< \nabla \widetilde{F}(\hat{\boldsymbol{w}}), \hat{\boldsymbol{w}} - \boldsymbol{w}^* >= \lambda \|\hat{\boldsymbol{w}} - \boldsymbol{w}^*\|^2$. This implies $\hat{\boldsymbol{w}} = \boldsymbol{w}^*$. □

C.2.3 Proof of Lemma 4.3.3

Let $\bar{\boldsymbol{w}}$ be a stationary point of MKL-SGD . Now, we analyze the loss landscape on the line joining $\boldsymbol{w}^*$ and $\boldsymbol{w}_C$ where $\boldsymbol{w}_C = C\bar{\boldsymbol{w}}$ is any arbitrary point [1] in the landscape at a distance as far as the farthest outlier from $\boldsymbol{w}^*$. Let C be a very large number.

[1]Note that we just need $\boldsymbol{w}_C$ for the purpose of landscape analysis and it is not a parameter of the algorithm

The loss functions and $\widetilde{\boldsymbol{w}}$ are redefined as follows:

$$f_i(\boldsymbol{w}) = \begin{cases} l_i\|\boldsymbol{w} - \boldsymbol{w}^*\|^2 & \forall\ i \in \mathbb{O} \\ l_i\|\boldsymbol{w} - \boldsymbol{w}_{b_i}\|^2 & \forall\ i \notin \mathbb{O}, \end{cases}$$

$$\widetilde{\boldsymbol{w}} := \left\{ \boldsymbol{w} \middle| \begin{array}{l} \boldsymbol{w} = \min\limits_{\alpha \in (0,1)} \alpha \boldsymbol{w}^* + (1-\alpha)\boldsymbol{w}_C, \\ f_{l_m}(\boldsymbol{w}) = f_{l_M}(\boldsymbol{w}) \end{array} \right\}$$

where $|\mathbb{O}| = b$ such that $n = g + b$. Let $l_m = \min_{i \notin \mathbb{O}} l_i$ and Let $l_M = \max_{i \in \mathbb{O}} l_i$ and $l_{max} = \min_{i \in [n]} l_i$, $l_{min} = \min_{i \in [n]} l_i$. Let us define $\kappa = \dfrac{l_{max}}{l_{min}} \geq \dfrac{l_M}{l_m}$.

Now at $\bar{w}$, we have $\nabla \widetilde{F}(\bar{\boldsymbol{w}}) = 0$. Let us assume that the outliers are chosen in such a way that at $\boldsymbol{w}_C$, all the outliers have the lowest loss. As stated in the previous lemma, the results hold irrespective of that. This implies:

$$\begin{aligned} \sum_{i \notin \mathbb{O}} p_i(\boldsymbol{w}_C)\nabla f_i(\bar{\boldsymbol{w}}) &= -\sum_{j \in \mathbb{O}} p_j(\boldsymbol{w}_C)\nabla f_j(\bar{\boldsymbol{w}}) \\ \sum_{i \notin \mathbb{O}} p_i(\boldsymbol{w}_C)l_i(\bar{\boldsymbol{w}} - \boldsymbol{w}^*) &= -\sum_{j \in \mathbb{O}} p_j(\boldsymbol{w}_C)l_j(\bar{\boldsymbol{w}} - \boldsymbol{w}_{b_j}) \\ \bar{\boldsymbol{w}} &= \frac{\sum_{i \notin \mathbb{O}} p_i(\boldsymbol{w}_C)l_i\boldsymbol{w}^* + \sum_{j \in \mathbb{O}} p_j(\boldsymbol{w}_C)l_j\boldsymbol{w}_{b_j}}{\sum_{i \notin \mathbb{O}} p_i(\boldsymbol{w}_C)l_i + \sum_{j \in \mathbb{O}} p_j(\boldsymbol{w}_C)l_j} \\ \text{By triangle inequality, } \|\bar{\boldsymbol{w}} - \boldsymbol{w}^*\| &\leq \frac{\sum_{j \in \mathbb{O}} p_j(\boldsymbol{w}_C)l_j\|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\sum_{i \notin \mathbb{O}} p_i(\boldsymbol{w}_C)l_i + \sum_{j \in \mathbb{O}} p_j(\boldsymbol{w}_C)l_j} \end{aligned}$$

Without loss of generality assume that the outliers are ordered as follows: $\|\boldsymbol{w}_{b_1} - \boldsymbol{w}^*\| \leq \|\boldsymbol{w}_{b_2} - \boldsymbol{w}^*\| \leq \cdots \leq \|\boldsymbol{w}_{b_{|\mathbb{O}|}} - \boldsymbol{w}^*\|$.

Now $\widetilde{\boldsymbol{w}}$ be some point of intersection of function in the set of clean samples and a function in the set of outliers to $\boldsymbol{w}^*$. Let θ_j be the angle between the line connecting $\boldsymbol{w}_{b_j}$ and $\boldsymbol{w}^*$ to the line connecting $\boldsymbol{w}_C$ to $\boldsymbol{w}^*$. For any two curves with Lipschitz constants l_i and l_j, the halfspaces passing through the weighted mean are also the region where both functions have equal values.

Thus,

$$\widetilde{\boldsymbol{w}} = \frac{\sqrt{l_i}\boldsymbol{w}^* + \sqrt{l_j}\boldsymbol{w}_{b_j}}{\sqrt{l_i} + \sqrt{l_j}}$$

.

$$\|\widetilde{\boldsymbol{w}} - \boldsymbol{w}^*\| = \frac{\sqrt{l_j}\|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\sqrt{l_j} + \sqrt{l_i}}$$

Let γ denote the following ratio:

$$\gamma = \frac{\min_{j\in\mathbb{O}} \|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\max_{j\in\mathbb{O}} \|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|} = \frac{2\delta}{\delta_{max}}$$

Now, we want:

$$\begin{aligned}
\frac{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j \|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C) l_i + \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j} &\leq \frac{\sqrt{l_{t_j}}}{\sqrt{l_{t_j}} + \sqrt{l_g}} \frac{\|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\cos\theta_j} = \frac{\|\widetilde{\boldsymbol{w}} - \boldsymbol{w}^*\|}{\cos\theta_j} \\
\frac{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j \|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C) l_i + \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j} &\leq \frac{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j \|\boldsymbol{w}_{b_{|\mathbb{O}|}} - \boldsymbol{w}^*\|}{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C) l_i + \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j} \\
&\leq \frac{\sqrt{l_{t_j}}}{\sqrt{l_{t_j}} + \sqrt{l_g}} \frac{\|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\cos\theta_j} \\
\Rightarrow \frac{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j}{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C) l_i + \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j} &\leq \frac{\sqrt{l_{t_j}}}{\sqrt{l_{t_j}} + \sqrt{l_g}} \frac{\|\boldsymbol{w}_{b_j} - \boldsymbol{w}^*\|}{\cos\theta_j \|\boldsymbol{w}_{b_{|\mathbb{O}|}} - \boldsymbol{w}^*\|} \\
\frac{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j}{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C) l_i + \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C) l_j} &\leq \frac{\sqrt{l_{t_j}}}{\sqrt{l_{t_j}} + \sqrt{l_g}} \frac{\gamma}{\cos\theta_j}
\end{aligned}$$

For simplicity, $\Gamma = \dfrac{\gamma}{\cos\theta_j}$, then we have:

$$\begin{aligned}
\frac{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C)l_j}{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C)l_i + \sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C)l_j} &\leq \frac{\sqrt{l_{t_j}}}{\sqrt{l_{t_j}} + \sqrt{l_g}}\Gamma \\
\frac{1}{\Gamma}\left(\frac{\sqrt{l_g}}{\sqrt{l_{t_j}}} + 1\right) - 1 \leq \frac{(1-\hat{p})l_m}{\hat{p}l_M} &\leq \frac{\sum_{i\notin\mathbb{O}} p_i(\boldsymbol{w}_C)l_i}{\sum_{j\in\mathbb{O}} p_j(\boldsymbol{w}_C)l_j} \\
\frac{\hat{p}}{1-\hat{p}} &\leq \frac{\frac{l_m}{l_M}}{\frac{1}{\Gamma} - 1 + \frac{1}{\Gamma}\frac{\sqrt{l_g}}{\sqrt{l_{t_j}}}} \\
\hat{p} &\leq \frac{1}{1 + \kappa\left(\frac{1}{\Gamma} - 1 + \frac{\sqrt{\kappa}}{\Gamma}\right)} \\
&\leq \frac{1}{1 + \frac{l_M}{l_m}\left(\frac{1}{\Gamma} - 1 + \frac{1}{\Gamma}\frac{\sqrt{l_g}}{\sqrt{l_{t_j}}}\right)}
\end{aligned}$$

Replacing $\Gamma = \dfrac{\gamma}{\cos\theta_j}$, and let $q = \frac{\cos\theta_j}{\gamma} - 1 + \frac{\cos\theta_j\sqrt{\kappa}}{\gamma}$ the condition to guarantee that bad local minima do no exist is $\hat{p} \leq \dfrac{1}{1+\kappa q}$ and $q > 0$.

Note: In the vector case, for example there exists a fine tradeoff between how large θ_j can be and if for large θ_j, the loss corresponding to the outlier will be one of the lowest. Understanding that tradeoff is beyond the scope of this paper.

C.2.4 Proof of Lemma 4.3.4

Proof. At $\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}$, $\nabla \widetilde{F}(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}) = 0$. Then,

$$\begin{aligned}
\sum_{i \notin \mathbb{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}) &= -\sum_{i \in \mathcal{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}) \\
\|\sum_{i \notin \mathbb{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| &= \|\sum_{i \in \mathcal{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| \\
\|\sum_{i \notin \mathbb{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| &\leq \sum_{i} \|\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| \\
&\leq \sum_{i} L\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\| \\
&= (1-\epsilon)nL\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\| \qquad \text{(C.1)} \\
\|\sum_{i \in \mathcal{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| &\leq \sum_{i \in \mathcal{O}} \|\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| \\
&\leq \sum_{i \in \mathcal{O}} G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}) \\
&\leq \epsilon G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}) \qquad \text{(C.2)}
\end{aligned}$$

$$\|\sum_{i \in \mathcal{O}} \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})\| = \min\left(\epsilon n G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}), (1-\epsilon)nL\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\|\right)$$

□

C.2.5 Proof of Lemma 4.3.5

Proof. At $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}$, $\nabla\widetilde{F}(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) = 0$. This implies

$$\sum_{i \notin \mathbb{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) = -\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})$$

Multiplying both sides by $(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*)$

$$\sum_{i \notin \mathbb{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) < \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* >$$

$$= -\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) < \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* >$$

$$< \nabla\widetilde{F}_{\mathcal{G}}(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* >$$

$$= -\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) < \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* >$$

Lower bounding the LHS using Lemma 4.3.1 and $m = m(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})^2$,

$$m\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|^2 \leq \| < \nabla\widetilde{F}_{\mathcal{G}}(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* > \| = LHS$$

$$RHS \leq \| -\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) < \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* > \|$$

$$m\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|^2 \leq \sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| < \nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}), \bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^* > \|$$

$$m\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|^2 \leq \sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\|\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\|\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|$$

$$m\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|^2 \leq \sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})$$

$$m\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \leq \epsilon^k G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})$$

□

C.2.6 Proof of Theorem 4.3.6

Proof. There exists an $\epsilon' \leq \epsilon$ such that in Lemma 4.3.4, we have

$$(1-\epsilon)L\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\| \geq \epsilon G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}})$$

Combining above equation with Lemma 4.3.5, we get

$$\begin{aligned} (1-\epsilon)L\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\| \geq \epsilon G(\bar{\boldsymbol{w}}_{\boldsymbol{SGD}}) \geq \epsilon\frac{\lambda}{\epsilon^2}\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \\ \Rightarrow \|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \leq \frac{(1-\epsilon)L\epsilon^{k-1}}{\lambda}\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\| \end{aligned}$$

Picking a large enough k, we can guarantee that $\dfrac{(1-\epsilon)L\epsilon^{k-1}}{\lambda} < 1$ □

C.2.7 Proof of Lemma C.1.1

Proof. From the definition of good samples in the noiseless setting, we know that $f_i(\boldsymbol{w}^*) = 0 \ \forall \ i \notin \mathbb{O}$. Similarly, for samples belonging to the outlier set, $f_i(\boldsymbol{w}^*) > 0 \ \forall \ i \in \mathcal{O}$. There exists a ball around the optimum of radius r such that $f_i(\boldsymbol{w}) \leq f_j(\boldsymbol{w}) \ \forall i \notin \mathbb{O}, j \in \mathcal{O}, \boldsymbol{w} \in \mathbb{O}_r(\boldsymbol{w}^*)$. Assume that $N' \geq N$ and $\epsilon' \leq \epsilon$, such that $\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \leq r$.

At $\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}$, $\nabla\widetilde{F}(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) = 0$. This implies

$$\begin{aligned}
\sum_{i \notin \mathbb{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) &= -\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) \\
\|\sum_{i \notin \mathbb{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| &= \|\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| \\
\|\sum_{i \notin \mathbb{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| &\le \sum_{i} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\|\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| \\
&\le \sum_{i} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})L\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \\
&= (1-\epsilon^k)L\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\| \qquad \text{(C.3)} \\
& \qquad \text{(C.4)}
\end{aligned}$$

$$\begin{aligned}
\|\sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| &\le \sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\|\nabla f_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})\| \\
&\le \sum_{i \in \mathcal{O}} p_i(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}})G(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) \\
&\le \epsilon^k G(\bar{\boldsymbol{w}}_{\boldsymbol{MKL}}) \qquad \text{(C.5)}
\end{aligned}$$

□

C.3 Additional results and proofs for Section 4.4

Consider the sample size n with bad set(outlier) $\mathbb{O}$ and good set $\mathcal{G}$ such that $|\mathcal{G}| = n - |\mathbb{O}|$. Define

$$F_{good}(\boldsymbol{w}) = \frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} f_i(\boldsymbol{w}).$$

We assume:

(1) (Stationary point) Assume $\boldsymbol{w}^*$ is the solution for the average loss function

of good sample such that

$$\nabla F_{good}(\boldsymbol{w}^*) = 0 \quad \text{but } \nabla f_i(\boldsymbol{w}^*) \neq 0, \forall i$$

(2) (Strong Convexity) $F_{good}(\boldsymbol{w})$ is strongly convex with parameters λ_{good} i.e.,

$$\langle \nabla F_{good}(\boldsymbol{w}) - \nabla F_{good}(\boldsymbol{w}^*), \boldsymbol{w} - \boldsymbol{w}^* \rangle \ \geq \lambda_{good} \|\boldsymbol{w} - \boldsymbol{w}^*\|^2$$

(3) (Gradient Lipschitz) $f_i(\boldsymbol{w})$ has L_i Liptchitz gradient i.e.,

$$\|\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^*)\| \ \leq L_i \|\boldsymbol{w} - \boldsymbol{w}^*\|$$

Theorem C.3.1. ***(Distance to w*)***

$$\mathbb{E}_i \left[\|\boldsymbol{w_{t+1}} - \boldsymbol{w}^*\|^2 | \boldsymbol{w_t} \right] \leq \left(1 - 2\eta_t \lambda_{good} (1 - \eta_t \sup_i L_i) \frac{|\mathcal{G}|}{n} \right) \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 + R_t \tag{C.6}$$

where

$$\begin{aligned} R_t = & - 2\eta_t (1 - \eta_t \sup_i L_i)) \sum_{i \in \mathcal{G}} \left(p_i(\boldsymbol{w_t}) - \frac{1}{n} \right) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*) \rangle \\ & - 2\eta_t \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w}^*) \rangle + 2\eta_t^2 \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \|\nabla f_i(\boldsymbol{w}^*)\|^2 \\ & + \eta_t^2 \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t}) \|\nabla f_i(\boldsymbol{w_t})\|^2 + 2\eta_t \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t}) \left(f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t}) \right) \end{aligned}$$

Proof. Observe first that for each component function i.e. ,

$$\langle \boldsymbol{w} - \boldsymbol{v}, \nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{v}) \rangle \geq \frac{1}{L_i} \|f_i(\boldsymbol{w}) - f_i(\boldsymbol{v})\|^2$$

For detailed proof, see Lemma A.1 in [90].

For each individual component function $f_i(\boldsymbol{w})$, we have

$$\begin{aligned}
\|\boldsymbol{w_{t+1}} - \boldsymbol{w}^*\|^2 =& \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 + \eta_t^2 \|\nabla f_i(\boldsymbol{w_t})\|^2 - 2\eta_t \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t})\rangle \\
\leq& \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 + 2\eta_t^2 \|\nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\|^2 + 2\eta_t^2 \|\nabla f_i(\boldsymbol{w}^*)\|^2 \\
\leq& \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 + 2\eta_t^2 L_i \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\rangle \\
&+ 2\eta_t^2 \|\nabla f_i(\boldsymbol{w}^*)\|^2 - 2\eta_t \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t})\rangle \\
=& \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 \\
&- 2\eta_t (1 - \eta_t \sup_i L_i) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\rangle \\
&+ 2\eta_t^2 \|\nabla f_i(\boldsymbol{w}^*)\|^2 - 2\eta_t \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w}^*)\rangle
\end{aligned}$$

We next take an expectation with respect to the choice of i conditional on $\boldsymbol{w_t}$

$$\begin{aligned}
&\mathbb{E}_i \left[\|\boldsymbol{w_{t+1}} - \boldsymbol{w}^*\|^2 | \boldsymbol{w_t}\right] \\
&\leq \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 - 2\eta_t (1 - \eta_t \sup_i L_i) \underbrace{\left\langle \boldsymbol{w_t} - \boldsymbol{w}^*, \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \left(\nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\right) \right\rangle}_{Term1} \\
&- 2\eta_t \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \nabla f_i(\boldsymbol{w}^*)\rangle + 2\eta_t^2 \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \|\nabla f_i(\boldsymbol{w}^*)\|^2 \\
&+ \eta_t^2 \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t}) \|\nabla f_i(\boldsymbol{w_t})\|^2 + 2\eta_t \underbrace{\langle \boldsymbol{w}^* - \boldsymbol{w_t}, \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t}) \nabla f_i(\boldsymbol{w_t})\rangle}_{Term2} \qquad \text{(C.7)}
\end{aligned}$$

We have $Term1$ as follows:

$$\begin{aligned}
Term1 =& \frac{|\mathcal{G}|}{n} \left\langle \boldsymbol{w_t} - \boldsymbol{w}^*, \sum_{i \in \mathcal{G}} \frac{p_i(\boldsymbol{w_t})}{|\mathcal{G}|/n} \left(\nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\right) \right\rangle \\
=& \frac{|\mathcal{G}|}{n} \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla F_{good}(\boldsymbol{w_t}) - \nabla F_{good}(\boldsymbol{w}^*)\rangle \\
&+ \frac{|\mathcal{G}|}{n} \sum_{i \in \mathcal{G}} \left(\frac{p_i(\boldsymbol{w_t})}{|\mathcal{G}|/n} - \frac{1}{|\mathcal{G}|}\right) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\rangle \\
\geq& \lambda_{good} \frac{|\mathcal{G}|}{n} \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 + \sum_{i \in \mathcal{G}} \left(p_i(\boldsymbol{w_t}) - \frac{1}{n}\right) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*)\rangle
\end{aligned}$$

For $Term2$ we apply use the property of the convex function $\langle \nabla f_i(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v} \rangle \leq f_i(\boldsymbol{w}) - f_i(\boldsymbol{v})$

$$\langle \nabla f_i(\boldsymbol{w_t}), \boldsymbol{w}^* - \boldsymbol{w_t} \rangle \leq f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t})$$

Putting the terms back to (C.7), we have for $\eta_t \leq 1/(\sup_i L_i)$

$$\mathbb{E}_i \left[\|\boldsymbol{w_{t+1}} - \boldsymbol{w}^*\|^2 | \boldsymbol{w_t} \right] \leq \left(1 - 2\eta_t \lambda_{good} (1 - \eta_t \sup_i L_i) \frac{|\mathcal{G}|}{n} \right) \|\boldsymbol{w_t} - \boldsymbol{w}^*\|^2 + R_t \tag{C.8}$$

where

$$\begin{aligned} R_t = & - 2\eta_t(1 - \eta_t \sup_i L_i)) \sum_{i \in \mathcal{G}} \left(p_i(\boldsymbol{w_t}) - \frac{1}{n} \right) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w_t}) - \nabla f_i(\boldsymbol{w}^*) \rangle \\ & - 2\eta_t \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \langle \boldsymbol{w_t} - \boldsymbol{w}^*, \nabla f_i(\boldsymbol{w}^*) \rangle \\ & + 2\eta_t^2 \sum_{i \in \mathcal{G}} p_i(\boldsymbol{w_t}) \|\nabla f_i(\boldsymbol{w}^*)\|^2 + \eta_t^2 \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t}) \|\nabla f_i(\boldsymbol{w_t})\|^2 \\ & + 2\eta_t \sum_{i \in \mathbb{O}} p_i(\boldsymbol{w_t}) \left(f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t}) \right) \end{aligned}$$

□

We have the following corollary that for noiseless setting, if we can have some good initialization, MKL-SGD is always better than SGD even the corrupted data is greater than half. For noisy setting, we can also perform better than SGD with one more condition: the noise is not large than the distance $\|\Delta_t\|^2$. This condition is not mild in the sense that $\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2$ is always greater than $\|\bar{\boldsymbol{w}}_{\boldsymbol{SGD}} - \boldsymbol{w}^*\|^2$ for SGD algorithm and $\|\bar{\boldsymbol{w}}_{\boldsymbol{MKL}} - \boldsymbol{w}^*\|^2$ for MKL-SGD.

Corollary C.3.2. *Suppose we have $|\mathcal{G}| \le \frac{n}{2}$. At iteration t for $\eta_t \le \frac{1}{\sup_i L_i}$, the parameter $\boldsymbol{w_t}$ satisfies $\sup_{i\in\mathcal{G}} f_i(\boldsymbol{w_t}) \le \inf_{j\in\mathbb{O}} f_j(\boldsymbol{w_t})$. Moreover, assume the noise level at optimal $\boldsymbol{w}^*$ satisfies*

either

$$\|\nabla f_i(\boldsymbol{w}^*)\| \le \frac{\lambda_{good}(1-\eta_t \sup_i L_i)/n}{1+\sqrt{1+\eta_t(1-\eta_t\sup_i L_i)\lambda_{good}/n}}\|\boldsymbol{w_t}-\boldsymbol{w}^*\|, \text{ for } i\in\mathcal{G} \quad \text{(C.9)}$$

or

$$\sum_{i\in\mathcal{G}}\|\nabla f_i(\boldsymbol{w}^*)\|^2 \le \left(\frac{\lambda_{good}(1-\eta_t\sup_i L_i)|\mathcal{G}|/n}{\sqrt{n}+\sqrt{\sqrt{n}+\eta_t(1-\eta_t\sup_i L_i)\lambda_{good}|\mathcal{G}|/n}}\right)^2\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2. \quad \text{(C.10)}$$

Using the same setup, the vanilla SGD and MKL-SGD (K=2) algorithms yield respectively

SGD

$$\mathbb{E}_i\left[\|\boldsymbol{w_{t+1}}-\boldsymbol{w}^*\|^2|\boldsymbol{w_t}\right] \le \left(1-2\eta_t\lambda_{good}(1-\eta_t\sup_i L_i)\frac{|\mathcal{G}|}{n}\right)\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2 + R_t^{(SGD)}$$

MKL-2

$$\mathbb{E}_i\left[\|\boldsymbol{w_{t+1}}-\boldsymbol{w}^*\|^2|\boldsymbol{w_t}\right] \le \left(1-2\eta_t\lambda_{good}(1-\eta_t\sup_i L_i)\frac{|\mathcal{G}|}{n}\right)\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2 + R_t^{(MKL_2)}$$

where

$$R_t^{(MKL_2)} \le R_t^{(SGD)}.$$

Proof. We use Theorem C.3.1 to analyse the term R_t for vanilla SGD and MKL-SGD($K=2$) respectively.

For vanilla SGD, we have $p_i(\boldsymbol{w_t}) = \frac{1}{n}$ and $\sum_{i\in\mathcal{G}} \nabla f_i(\boldsymbol{w}^*) = 0$, which results in

$$R_t^{(SGD)} = \frac{2\eta_t^2}{n}\sum_{i\in\mathcal{G}} \|\nabla f_i(\boldsymbol{w}^*)\|^2 + \frac{\eta_t^2}{n}\sum_{i\in\mathbb{O}} \|\nabla f_i(\boldsymbol{w_t})\|^2 + \frac{2\eta_t}{n}\sum_{i\in\mathbb{O}} (f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t}))$$

note that MKL-SGD for $K = 2$ have

$$p_{m_i(\boldsymbol{w})}(\boldsymbol{w}) = \frac{2(n-i)}{n(n-1)} \tag{C.11}$$

where $m_1(\boldsymbol{w}), m_2(\boldsymbol{w}), m_3(\boldsymbol{w}), \ldots m_n(\boldsymbol{w})$ are the indices of data samples for some $\boldsymbol{w}$:

$$f_{m_1(\boldsymbol{w})}(\boldsymbol{w}) \le f_{m_2(\boldsymbol{w})}(\boldsymbol{w}) \le \cdots \le f_{m_n(\boldsymbol{w})}(\boldsymbol{w})$$

Suppose the iteration $\boldsymbol{w_t}$ satisfies that $f_i(\boldsymbol{w_t}) < f_j(\boldsymbol{w_t})$ for $i \in \mathcal{G}, j \in \mathbb{O}$. For $|\mathcal{G}| \le \frac{n}{2}$, we have for

$$\begin{aligned}
R_t^{(MKL_2)} =& -2\eta_t(1-\eta_t \sup_i L_i)\sum_{i=1}^{|\mathcal{G}|}\frac{(n-2i+1)}{n(n-1)}\langle \boldsymbol{w_t}-\boldsymbol{w}^*, \nabla f_{m_i}(\boldsymbol{w_t}) - \nabla f_{m_i}(\boldsymbol{w}^*)\rangle \\
&+ 2\eta_t \sum_{i=1}^{|\mathcal{G}|}\frac{2(n-i)}{n(n-1)}\left(\langle \boldsymbol{w}^* - \boldsymbol{w_t}, \nabla f_i(\boldsymbol{w}^*)\rangle + \eta_t\|\nabla f_i(\boldsymbol{w}^*)\|^2\right) \\
&+ \eta_t^2 \sum_{i=|\mathcal{G}|+1}^{n}\frac{2(n-i)}{n(n-1)}\|\nabla f_i(\boldsymbol{w_t})\|^2 + 2\eta_t\sum_{i=|\mathcal{G}|+1}^{n}\frac{2(n-i)}{n(n-1)}(f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t})) \\
\le& -2\eta_t(1-\eta_t\sup_i L_i)\frac{|\mathcal{G}|\lambda_{good}}{n(n-1)}\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2 \\
&+ \frac{4\eta_t}{n}\sum_{i=1}^{|\mathcal{G}|}\left(\|\boldsymbol{w}^*-\boldsymbol{w_t}\|\|\nabla f_i(\boldsymbol{w}^*)\| + \eta_t\|\nabla f_i(\boldsymbol{w}^*)\|^2\right) \\
&+ \sum_{i=|\mathcal{G}|+1}^{n}\frac{\eta_t^2}{n}\|\nabla f_i(\boldsymbol{w_t})\|^2 + \sum_{i=|\mathcal{G}|+1}^{n}\frac{2\eta_t}{n}(f_i(\boldsymbol{w}^*) - f_i(\boldsymbol{w_t}))
\end{aligned}$$

We will have $R_t^{(MKL_2)} \leq R_t^{(SGD)}$ if we can

$$(1-\eta_t \sup_i L_i)\frac{|\mathcal{G}|\lambda_{good}}{(n-1)}\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2 \geq \sum_{i=1}^{|\mathcal{G}|}\left(2\|\boldsymbol{w}^*-\boldsymbol{w_t}\|\|\nabla f_i(\boldsymbol{w}^*)\|+\eta_t\|\nabla f_i(\boldsymbol{w}^*)\|^2\right). \tag{C.12}$$

Indeed, for the noise level $\|\nabla f_i(\boldsymbol{w}^*)\|^2$ satisfying (C.9) we have for $i \in \mathcal{G}$,

$$(1-\eta_t \sup_i L_i)\frac{\lambda_{good}}{(n-1)}\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2 \geq 2\|\boldsymbol{w}^*-\boldsymbol{w_t}\|\|\nabla f_i(\boldsymbol{w}^*)\|+\eta_t\|\nabla f_i(\boldsymbol{w}^*)\|^2.$$

Summing up the terms in $i \in \mathcal{G}$, we get (C.12). For the noise level $\|\nabla f_i(\boldsymbol{w}^*)\|^2$ satisfying (C.10) we have

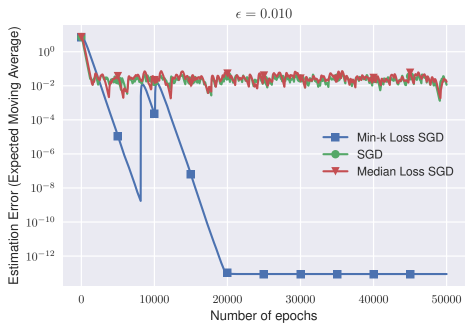
$$\begin{aligned}(1-\eta_t \sup_i L_i)\frac{\lambda_{good}|\mathcal{G}|}{(n-1)}\|\boldsymbol{w_t}-\boldsymbol{w}^*\|^2 &\geq \left(2\|\boldsymbol{w}^*-\boldsymbol{w_t}\|\sqrt{n\sum_{i\in\mathcal{G}}\|\nabla f_i(\boldsymbol{w}^*)\|^2}+\eta_t\sum_{i\in\mathcal{G}}\|\nabla f_i(\boldsymbol{w}^*)\|^2\right)\\ &\geq 2\|\boldsymbol{w}^*-\boldsymbol{w_t}\|\sum_{i\in\mathcal{G}}\|\nabla f_i(\boldsymbol{w}^*)\|+\eta_t\sum_{i\in\mathcal{G}}\|\nabla f_i(\boldsymbol{w}^*)\|^2.\end{aligned}$$

which results in (C.12). □

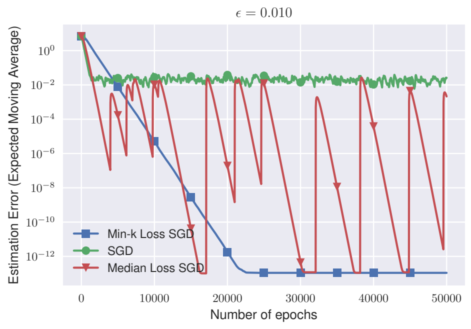
C.4 More experimental results

C.4.1 Linear Regression

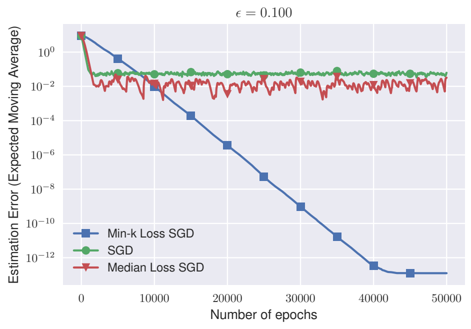
Here, we show that there exists a tradeoff for MKL-SGD between the rate of convergence and robustness it provides against outliers. Larger the k, more robust is the algorithm, but slower is the rate of convergence. The algorithm outperforms median loss SGD and SGD. We also experimentd with other order statistics and observed that for most general settings min-k loss was the best to pick.



(a) k=2

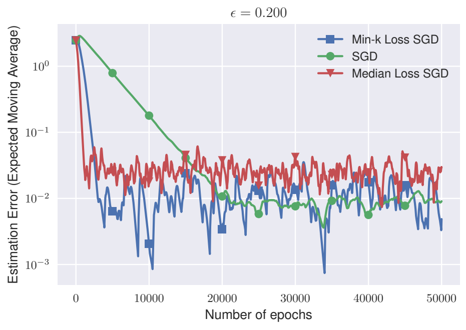


(b) k=3

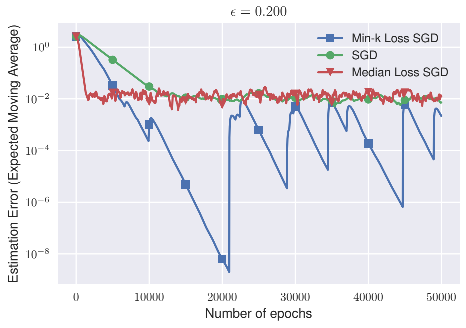


(c) k=5

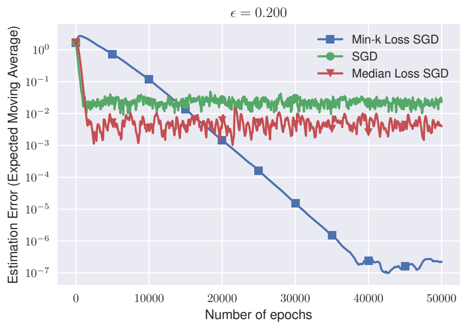
Figure C.1: Comparing the performance of MKL-SGD , SGD and Median loss SGD in the noiseless setting, $d = 50$.



(a) k=3

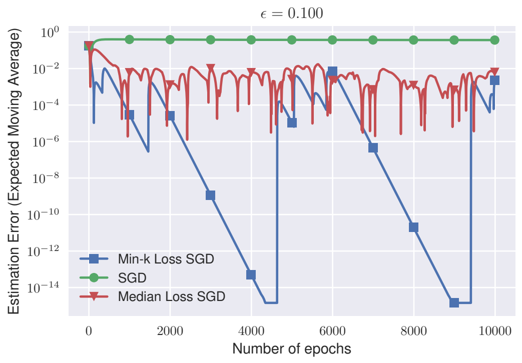


(b) k=5

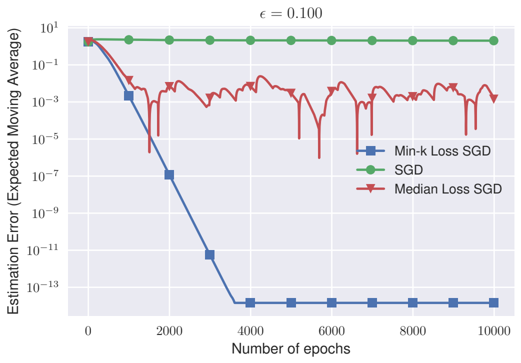


(c) k=9

Figure C.2: Comparing the performance of MKL-SGD , SGD and Median loss SGD in the noisy setting, $d = 10$, Noise variance=0.0001

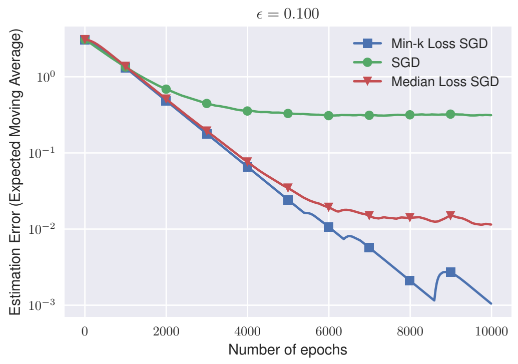


(a) k=3

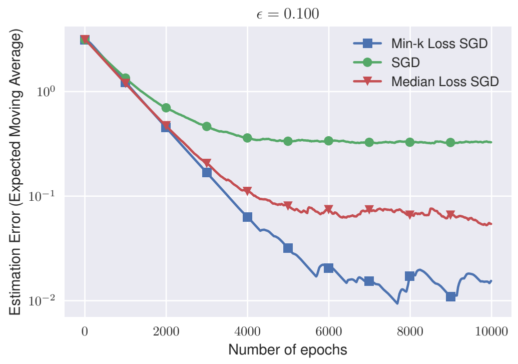


(b) k=5

Figure C.3: Comparing the performance of MKL-SGD , SGD and Median loss SGD in the noiseless setting, $d = 25$, Noise variance=0.01



(a) k=3



(b) k=5

Figure C.4: Comparing the performance of MKL-SGD , SGD and Median loss SGD in the noisy setting, $d = 10$, Noise variance=0.1

C.4.2 Deep Learning

Here, we show that in presence of outliers instead of tuning other hyperparameters like learning rate, tuning over k might lead to significant gains in performances for deep neural networks.

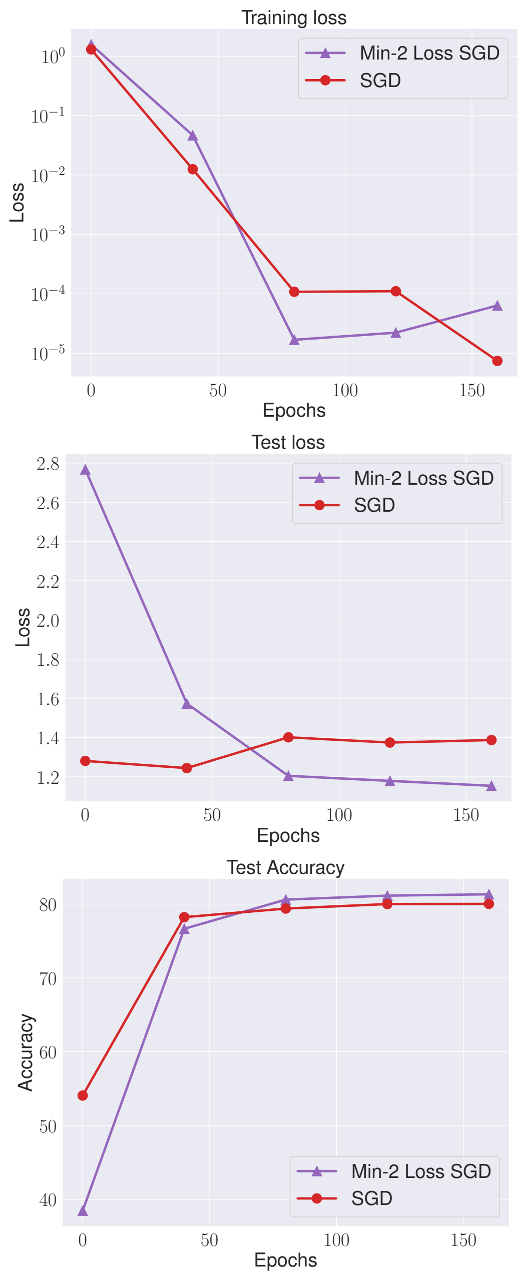


Figure C.5: Comparing training loss, test loss and test accuracy of MKL-SGD and SGD. Parameters: $\epsilon = 0.1$, $k = 2$, $b = 16$. The training loss is lower for SGD which means that SGD overfits to the noisy data. The lower test loss and higher accuracy demonstrates the robustness MKL-SGD provides for corrupted data.

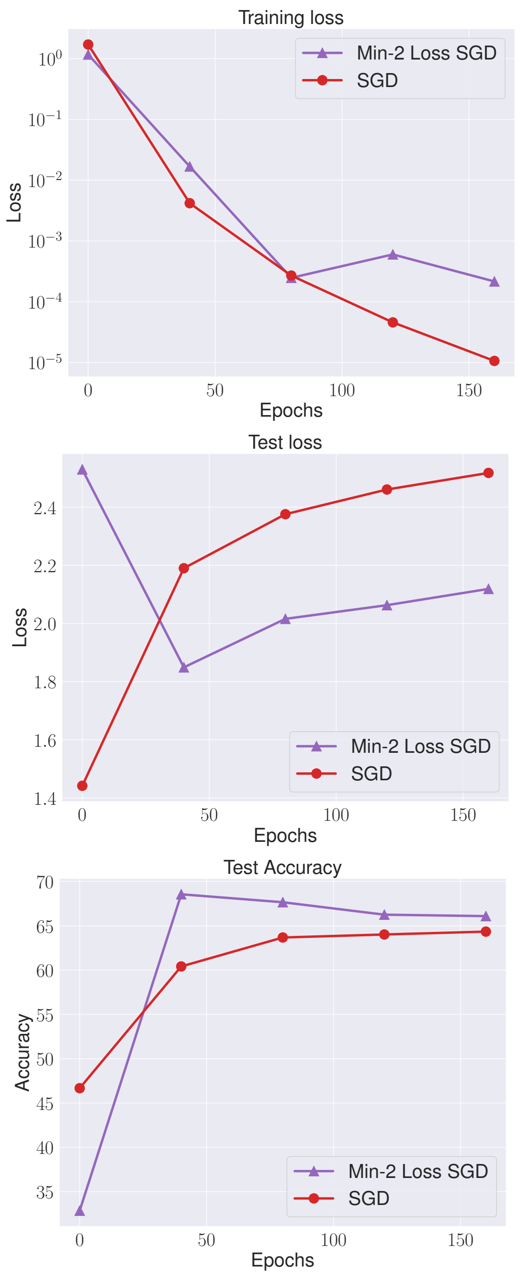


Figure C.6: Comparing training loss, test loss and test accuracy of MKL-SGD and SGD. Parameters: $\epsilon = 0.3$, $k = 2$, $b = 16$. The training loss is lower for SGD which means that SGD overfits to the noisy data. The lower test loss and higher accuracy demonstrates the robustness MKL-SGD provides for corrupted data.

Dataset	MNIST with 2-layer CNN (Directed Noise)						
Optimizer	**SGD**	**MKL-SGD**					**Oracle**
$\epsilon \alpha$	1.0	0.9	0.8	0.7	0.6	0.5	1.0
0.1	96.76	97.23	95.89	**97.47**	96.34	94.54	98.52
0.2	92.54	95.81	95.58	**97.46**	97.03	95.76	98.33
0.3	85.77	91.56	93.59	95.30	**96.54**	95.96	98.16
0.4	71.95	78.68	82.25	85.93	91.29	**94.20**	97.98

Table C.1: In this experiments, we train a standard 2 layer CNN on subsampled MNIST (5000 training samples with labels corrupted using random label noise). We train over 80 epochs using an initial learning rate of 0.05 with the decaying schedule of factor 5 after every 30 epochs. The reported accuracy is based on the true validation set. The results of the MNIST dataset are reported as the mean of 5 runs. For the MKL-SGD algorithm, we introduce a more practical variant that evaluates k sample losses and picks a batch of size αk where $k = 10$.

Dataset	MNIST with 2-layer CNN (Random Noise)						
Optimizer	**SGD**	**MKL-SGD**					**Oracle**
$\epsilon \backslash \alpha$	1.0	0.9	0.8	0.7	0.6	0.5	1.0
0.1	96.91	97.9	**98.06**	97.59	96.49	94.43	98.44
0.2	93.94	95.5	96.16	97.02	**97.04**	96.25	98.18
0.3	87.14	90.71	91.60	92.97	94.54	**95.36**	97.8
0.4	71.83	74.31	76.6	78.30	77.58	**80.86**	97.16

Table C.2: In this experiments, we train a standard 2 layer CNN on subsampled MNIST (5000 training samples with labels corrupted using random label noise). We train over 80 epochs using an initial learning rate of 0.05 with the decaying schedule of factor 5 after every 30 epochs. The reported accuracy is based on the true validation set. The results of the MNIST dataset are reported as the mean of 5 runs. For the MKL-SGD algorithm, we introduce a more practical variant that evaluates k sample losses and picks a batch of size αk where $k = 10$.

Dataset	**CIFAR-10 with Resnet-18 (Directed Noise)**						
Optimizer	**SGD**	**MKL-SGD**					**Oracle**
α \ ϵ	1.0	0.9	0.8	0.7	0.6	0.5	1.0
0.1	79.1	77.52	79.57	81.00	**81.94**	80.53	84.56
0.2	72.29	69.58	70.17	72.76	77.77	**78.93**	84.40
0.3	63.96	61.43	60.46	61.58	66.49	**69.57**	84.66
0.4	**52.4**	51.53	51.04	51.07	53.57	51.2	84.42

Table C.3: In this experiments, we train Resnet 18 on CIFAR-10 (50000 training samples with labels corrupted using directed label noise). We train over 200 epochs using an initial learning rate of 0.05 with the decaying schedule of factor 5 after every 90 epochs. The reported accuracy is based on the true validation set. The results of the CIFAR-10 dataset are reported as the mean of 3 runs. For the MKL-SGD algorithm, we introduce a more practical variant that evaluates k sample losses and picks a batch of size αk where $k = 16$.

Appendix D

Balancing SGD

D.1 Proof of Proposition 5.4.2

Theorem D.1.1. *Suppose the Loss function defined in* (5.5). *Assume that the estimator* $\hat{\boldsymbol{w}}$ *close to stationary points satisfying for some finite* $\hat{b}$ $\forall i, \exists \varepsilon$ *such that*

$$y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)} \geq \log\left(\frac{1}{\varepsilon}\right) \qquad \textit{and} \qquad e^{-y_i\left(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b\right)} < 1.$$

Then $\frac{\partial L(\boldsymbol{w})}{\partial b} |[] \boldsymbol{w} = \hat{\boldsymbol{w}} = 0$ *implies*

$$b \propto \frac{1}{2}\log\left(\frac{\sum_{i\in\mathcal{S}_1} e^{-y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}}{\sum_{i\in\mathcal{S}_{-1}} e^{-y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}}\right) \tag{D.1}$$

Proof. First, we look at the gradient

$$\begin{aligned}\frac{\partial L(\boldsymbol{w})}{\partial b} =& -\frac{1}{n}\sum_{i=1}^{n} \sigma(y_i\left(\boldsymbol{x}_i^\top \boldsymbol{w} + b\right)) e^{-y_i\left(\boldsymbol{x}_i^\top \boldsymbol{w}+b\right)} y_i \\ =& \frac{1}{n}\left(\sum_{i\in\mathcal{S}_0} \frac{-y_i e^{-y_i(\boldsymbol{x}_i^\top \boldsymbol{w}^{(d)}+b)}}{1+e^{-y_i(\boldsymbol{x}_i^\top \boldsymbol{w}^{(d)}+b)}} + \sum_{i\in\mathcal{S}_1} \frac{-y_i e^{-y_i(\boldsymbol{x}_i^\top \boldsymbol{w}^{(d)}+b)}}{1+e^{-y_i(\boldsymbol{x}_i^\top \boldsymbol{w}^{(d)}+b)}}\right)\end{aligned} \tag{D.2}$$

There exists a small ε for all i such that $y_i \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)} \geq \log\left(\frac{1}{\varepsilon}\right)$ and $e^{-y_i\left(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b\right)} <$

1 for some b. Thus, we have the following Taylor expansion:

$$\begin{aligned}\frac{1}{1+e^{-y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b)}} &= 1 - e^{-y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b)} + \mathcal{O}\left(e^{-2y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b)}\right) \\ &= 1 - e^{-y_i b}\mathcal{O}(\varepsilon) \\ &= 1 - \mathcal{O}(\varepsilon) \end{aligned} \tag{D.3}$$

where the last equality we take into the account with the fact that b is a finite constant. Plugging (D.3) into equation (D.2) yields

$$\begin{aligned}\left.\frac{\partial L(\boldsymbol{w})}{\partial b}\right|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} =& \frac{1}{n}\sum_{i\in\mathcal{S}_0} -y_i e^{-y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b)}\left(1 - e^{-y_i b}\mathcal{O}(\varepsilon)\right) \\ &+ \frac{1}{n}\sum_{i\in\mathcal{S}_1} -y_i e^{-y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}+b)}\left(1 - e^{-y_i b}\mathcal{O}(\varepsilon)\right) \\ =& \frac{e^b}{n}\sum_{i\in\mathcal{S}_0} e^{-y_i\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}\left(1-\mathcal{O}(\varepsilon)\right) - \frac{e^{-b}}{n}\sum_{i\in\mathcal{S}_1} e^{-y_i\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}\left(1-\mathcal{O}(\varepsilon)\right)\end{aligned}$$

Solving $\left.\frac{\partial L(\boldsymbol{w})}{\partial b}\right|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} = 0$ gives

$$b \propto \frac{1}{2}\log\left(\frac{\sum_{i\in\mathcal{S}_1} e^{-y_i\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}}{\sum_{i\in\mathcal{S}_{-1}} e^{-y_i\boldsymbol{x}_i^\top \hat{\boldsymbol{w}}^{(d)}}}\right) \tag{D.4}$$

□

D.2 Proof of bias convergence relation for the toy example

Now, let us examine what $\boldsymbol{w}_{t+1}$ looks like given some $\boldsymbol{w}_t$.

$$\mathbb{E}\left[\boldsymbol{w}_{t+1}|\boldsymbol{w}_t\right] = \boldsymbol{w}_t - \eta\mathbb{E}\left[\nabla f_i(w_t)\right] \tag{D.5}$$

The expected gradient for this toy example at $\boldsymbol{w}_t = [a_t, b_t]^\top$ is given as:

$$\begin{aligned}
\mathbb{E}\left[\nabla f_i(\boldsymbol{w}_t)\right] &= -\boldsymbol{p}_1\sigma(-\boldsymbol{w}_t^\top \boldsymbol{p}_1) + \sum_{i\in\mathcal{S}_{-1}} x_i\sigma(\boldsymbol{w}_t^\top \boldsymbol{x_i}) \\
&\approx -\boldsymbol{p}_1\sigma(-\boldsymbol{w}_t^\top \boldsymbol{p}_1) + \int_0^1 \boldsymbol{x}_i\sigma(\boldsymbol{w}_t^\top \boldsymbol{x_i})\partial\boldsymbol{x}_i[0] \quad \text{... assuming small } r \\
&\approx -\boldsymbol{p}_1\sigma(-\boldsymbol{w}_t^\top \boldsymbol{p}_1) + \begin{bmatrix}\delta_t^{(w)} \\ \delta_t^{(b)}\end{bmatrix}
\end{aligned}$$

where $\delta_t^{(w)} = \frac{1}{a_t}\log\frac{\exp(a_t+b_t)+1}{\exp(a_t)+1}$ and
$\delta_t^{(b)} = \frac{1}{a_t^2}\left(\log a_t(\exp(a_t+b_t)+\exp(2b_t)) + Li_2(-\exp(a_t+b_t)) - Li_2(-\exp(b_t))\right)$

This leads us to two phases of running any SGD based algorithm on separable data: *i) Rapid correction in the direction of* $\boldsymbol{w}_t$, *ii) Increase in the magnitude of* $\boldsymbol{w}_t$. In general, equation (5.7) indicates that the frequent occurrence of gradient updates from the majority class than the minority push the separating hyper-plane further away from majority class. The farther the separating hyper-plane from the majority class, the smaller (and less useful) are the corresponding gradient updates from majority class. We know from [117, 118] that the rate of convergence is quite slow in Phase 2. However, if we restrict the subset associated with the denominator in equation (5.7), it is possible to reduce this bias term even further which in turn will lead to faster convergence.

Using the expected gradient from the toy example, we observe that while it is easy to see that the bias can affect generalization, it also leads to slow convergence. High value of bias, b_t ensures that the $\delta_t^{(w)}$ can scale up-to two times $\frac{1}{a_t}$ causing the expected gradient to keep on getting biased until a_t

is not large enough. Thus in the initial phase, when a_t is small, bias can be quite high. Only in the second phase when a_t increases in magnitude does the bias becomes small. In this paper, we want our proposed algorithm to reach Phase 2 with as little bias as possible, similar to the case when it would have received equal distribution of both classes. Similar to

D.3 Convergence

Lemma D.3.1. *(Stochastic PAUM in [114]) Consider the perceptron algorithm (Algorithm 6) with the linear classifier*

$$f_i(w) = sign(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \quad with \quad \|\boldsymbol{x}_i\|^2 = 1$$

and so the $\nabla_w f_i(w) = -y_i \boldsymbol{x}_i$. Suppose there exits $\boldsymbol{w}_* \in \mathbb{R}^d$ such that $\|\boldsymbol{w}_*\| = 1$ and $\gamma(\boldsymbol{w}_*, \mathcal{S}_0, \mathcal{S}_1) \geq \Gamma$. Then the number of updates made by the PAUM is bounded w.h.p. by

$$\frac{1}{\Gamma^2} + \frac{2(1-r)\tau_{-1} + 2r\tau_1}{\eta\Gamma^2} \tag{D.6}$$

Proof. Observe that

$$\begin{aligned}
\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t\|^2 + 2\eta y_i \langle \boldsymbol{w}_t, x_i \rangle \mathbb{1}_{i\in\mathcal{S}_0} + 2\eta y_i \langle \boldsymbol{w}_t, x_i \rangle \mathbb{1}_{i\in\mathcal{S}_1} + \eta^2 \|\boldsymbol{x}_i\|^2 \\
\mathbb{E}\left[\|\boldsymbol{w}_{t+1}\|^2\right] &\leq \mathbb{E}\left[\|\boldsymbol{w}_t\|^2\right] + 2\eta\tau_{-1}\mathbb{E}[\mathbb{1}_{i\in\mathcal{S}_0}] + 2\eta\tau_1\mathbb{E}[\mathbb{1}_{i\in\mathcal{S}_1}] + \eta^2\|\boldsymbol{x}_i\|^2 \\
&= \mathbb{E}\left[\|\boldsymbol{w}_t\|^2\right] + 2\eta(1-r)\tau_{-1} + 2\eta r\tau_1 + \eta^2 \qquad \text{take } \|\boldsymbol{x}_i\|^2 = 1 \\
\left(\mathbb{E}[\|\boldsymbol{w}_{t+1}\|]\right)^2 &\leq \mathbb{E}\left[\|\boldsymbol{w}_{t+1}\|^2\right] \leq 2t\eta(1-r)\tau_{-1} + 2\eta r\tau_1 + \eta^2 t
\end{aligned}$$

Algorithm 6 Stochastic PAUM (the minority class $y = 1$)

```
1: Initialize $\boldsymbol{w}$. Two margin parameters $\tau_{-1}$ and $\tau_1$
2: Given samples $D = (\boldsymbol{x}_i, y_i)_{i=1}^{\infty}$
3: for $t = 1$ to $m$ do
4:     Chose $i$ uniformly from $\{1, \dots, n\}$.
5:     if $y_i \boldsymbol{w}^\top \boldsymbol{x}_i \leq \tau_{y_i}$ then
6:         $\boldsymbol{w}^+ = \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$
7:     end if
8: end for
9: Return $\boldsymbol{w}$
```

On the other hand, we have

$$\begin{aligned}\langle \mathbb{E}[\boldsymbol{w}_{t+1}], \boldsymbol{w}_* \rangle &= \langle \mathbb{E}[\boldsymbol{w}_t], \boldsymbol{w}_* \rangle + \eta \mathbb{E}[y_i \langle x_i, \boldsymbol{w}_* \rangle] \\ &\geq \langle \mathbb{E}[\boldsymbol{w}_t], \boldsymbol{w}_* \rangle + \eta \Gamma \\ &\geq t \eta \Gamma\end{aligned}$$

Thus, we have

$$\begin{aligned}(t\eta\Gamma)^2 &\leq (\langle \mathbb{E}[\boldsymbol{w}_{t+1}], \boldsymbol{w}_* \rangle)^2 \\ &\leq \|\mathbb{E}[\boldsymbol{w}_{t+1}]\|^2 \|\boldsymbol{w}_*\|^2 \\ &\leq 2\eta t(1-r)\tau_{-1} + 2\eta r t \tau_1 + \eta^2 t\end{aligned}$$

which is upper bound (D.6) □

Theorem D.3.2 (Restatement of Theorem 5.4.3)**.** *Consider the B-SGD algorithm with fixed thresholds (over time) τ_{-1} and τ_1 for the majority and minority class, respectively. The expected gradient update step for B-SGD is as follows:*

$$\mathbb{E}\left[\boldsymbol{w}_{t+1} | \boldsymbol{w}_t\right] = \boldsymbol{w}_t - \eta \nabla f_i(\boldsymbol{w}_t) \mathbb{I}_{f_i(\boldsymbol{w}_t) \geq C_{y_i}} \tag{D.7}$$

Suppose the data is normalized (i.e., $\|\boldsymbol{x}_i\| = 1$). Define the margin

$$\gamma(\boldsymbol{w}, \mathcal{S}_{-1}, \mathcal{S}_1) := \min_i \frac{y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle\right)}{\|\boldsymbol{w}\|}.$$

Suppose there exists $\boldsymbol{w}_* \in \mathbb{R}^{d+1}$ such that $\|\boldsymbol{w}_*\| = 1$ and $\gamma(\boldsymbol{w}_*, \mathcal{S}_{-1}, \mathcal{S}_1) \geq \Gamma$. Then the number of gradient updates performed by Algorithm 3 is bounded w.h.p. by

$$\frac{1}{\Gamma^2\left((1-r)\beta_{-1} + r\beta_1\right)^2} + \frac{2(1-r)B_{-1} + 2rB_1}{\eta\Gamma^2\left((1-r)\beta_{-1} + r\beta_1\right)^2} \tag{D.8}$$

where $\beta_{y_i} = (\exp(C_{y_i}) - 1)\exp(-C_{y_i})$ and $B_{y_i} = -\log\left(\exp(C_{y_i}) - 1\right)$.

Note that here C_{y_i} is τ_{y_i} in Theorem 5.4.3.

Proof. Let $p_i(\boldsymbol{w}_t) = \frac{\exp(-y_i\langle w, \boldsymbol{x}_i\rangle)}{1+\exp(-y_i\langle w, \boldsymbol{x}_i\rangle)}$. Note that

$$\begin{aligned}\log(1 + \exp\left(-y_i\langle w, \boldsymbol{x}_i\rangle\right)) \geq C_{y_i} \quad &\Leftrightarrow \quad y_i\langle w, \boldsymbol{x}_i\rangle \leq \underbrace{-\log\left(\exp(C_{y_i}) - 1\right)}_{\tau_{y_i}} \\ &\Rightarrow \quad \underbrace{(\exp(C_{y_i}) - 1)\exp(-C_{y_i})}_{\beta_{y_i}} \leq p_i(\boldsymbol{w}_t) \leq 1\end{aligned}$$

Observe that

$$\begin{aligned}\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t\|^2 + 2\eta y_i\langle \boldsymbol{w}_t, x_i\rangle p_i(\boldsymbol{w}_t)\mathbb{1}_{i\in\mathcal{S}_0} \\ &\quad + 2\eta y_i\langle \boldsymbol{w}_t, x_i\rangle p_i(\boldsymbol{w}_t)\mathbb{1}_{i\in\mathcal{S}_1} + \eta^2\|p_i(\boldsymbol{w}_t)\boldsymbol{x}_i\|^2 \\ \mathbb{E}[\|\boldsymbol{w}_{t+1}\|^2] &\leq \mathbb{E}[\|\boldsymbol{w}_t\|^2] + 2\eta(1-r)B_{-1} + 2r\eta B_1 + \eta^2\|\boldsymbol{x}_i\|^2\end{aligned}$$

On the other hand, we have

$$\begin{aligned}\langle \mathbb{E}[\boldsymbol{w}_{t+1}], \boldsymbol{w}_*\rangle &= \langle \mathbb{E}[\boldsymbol{w}_t], \boldsymbol{w}_*\rangle + \eta \mathbb{E}[p_i(\boldsymbol{w}_t) y_i \langle x_i, \boldsymbol{w}_*\rangle] \\ &\geq \langle \mathbb{E}[\boldsymbol{w}_t], \boldsymbol{w}_*\rangle + \eta\beta_{-1}\mathbb{E}[\mathbb{1}_{i\in\mathcal{S}_{-1}}]\Gamma + \eta\beta_1\mathbb{E}[\mathbb{1}_{i\in\mathcal{S}_1}]\Gamma \\ &\geq \langle \mathbb{E}[\boldsymbol{w}_t], \boldsymbol{w}_*\rangle + ((1-r)\beta_{-1} + r\beta_1)\,\eta\Gamma \\ &\geq t\eta\,((1-r)\beta_{-1} + r\beta_1)\,\Gamma\end{aligned}$$

Thus, we have

$$\begin{aligned}(t\eta\Gamma\,((1-r)\beta_{-1} + r\beta_1))^2 &\leq (\langle \mathbb{E}[\boldsymbol{w}_{t+1}], \boldsymbol{w}_*\rangle)^2 \\ &\leq \|\mathbb{E}[\boldsymbol{w}_{t+1}]\|^2 \|\boldsymbol{w}_*\|^2 \\ &\leq 2\eta t(1-r)B_{-1} + 2\eta r t B_1 + \eta^2 t\end{aligned}$$

which yields our upper bound of t □

D.4 Proof of Proposition 5.4.4

Proposition D.4.1. *Suppose the data is normalized (i.e., $\|\boldsymbol{x}_i\| = 1$). Suppose $\hat{\boldsymbol{w}}$ minimizes both equations (5.10) and (5.11), then without loss of generality $\hat{\boldsymbol{w}}^\top \boldsymbol{x}_i < 0, \forall i \in \mathcal{S}_{-1}$. Set $\tau_{-1} \geq 0$ for the set $\mathcal{A}_{\hat{\boldsymbol{w}}}$. Solving for τ_{-1} given some r in equation (5.12), we observe that $\tau_{-1} > 0$ is monotonically decreasing function with respect to $r \in [0, 0.5]$. In addition, If $r \to 0.5$ or $\frac{r}{1-r} \to 1$, then $\tau_{-1} \to 0$. If $r \to 0$ or $\frac{r}{1-r} \to 0$, then we could let $\tau_{-1} \to \log(1+\exp(\|\hat{\boldsymbol{w}}\|))$ so that we have $\mathbb{1}_{\{\log\left(1+e^{-z_i}\right) > \tau_{-1}\}} = 0$.* [1]

[1]Note the the term of $\tau_{-1} \to \log(1+\exp(\|\hat{\boldsymbol{w}}\|))$ is a sufficient condition but not a necessary condition. As long as it is greater than $c\log(1+\exp(\langle \hat{\boldsymbol{w}}, \mathbf{x}_i\rangle))$ for each i with some $c > 1$.

Proof. The expression that establishes the relation of τ_{-1} and r is

$$\sum_{i \in \mathcal{S}_{-1}} \left(\frac{r}{1-r} - \mathbb{1}_{i \in \mathcal{A}_{\hat{\boldsymbol{w}}}} \right) \nabla f_i(\boldsymbol{w}) |[] \boldsymbol{w} = \hat{\boldsymbol{w}} = 0 \tag{D.9}$$

Let us now explicitly write down down the function f, ∇f and $\mathcal{A}_{\hat{\boldsymbol{w}}}$ for the majority class $y = -1$:

$$\begin{aligned} f(y_i = -1, \hat{y}_i) &= \log\left(1 + e^{\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle}\right) \\ \nabla_w f(y_i = -1, \hat{y}_i) &= \frac{\boldsymbol{x}_i}{1 + \exp\left(-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle\right)} \\ \mathcal{A}_{\hat{\boldsymbol{w}}} &= \left\{ i \in \mathcal{S}_{-1} \middle| \log\left(1 + e^{\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle}\right) > \tau_{-1} \right\} \\ &= \left\{ i \in \mathcal{S}_{-1} \middle| -\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle \leq -\log\left(e^{\tau_{-1}} - 1\right) \right\} \end{aligned}$$

Substituting above equations back to (D.9) gives

$$\frac{r}{1-r} \sum_{i \in \mathcal{S}_{-1}} \frac{\boldsymbol{x}_i}{1 + \exp\left(-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle\right)} = \sum_{i \in \mathcal{S}_{-1}} \frac{\boldsymbol{x}_i \mathbb{1}_{i \in \mathcal{A}_{\hat{\boldsymbol{w}}}}}{1 + \exp\left(-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle\right)}$$

For the class $i \in \mathcal{S}_{-1}$, we have $\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle < 0$, multiplying above with $\hat{\boldsymbol{w}}$, we get

$$\begin{aligned} & \frac{r}{1-r} \sum_{i \in \mathcal{S}_{-1}} \frac{-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle}{1 + \exp\left(-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle\right)} = \sum_{i \in \mathcal{S}_{-1}} \frac{-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle \mathbb{1}\{\log\left(1 + e^{\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle}\right) > \tau_{-1}\}}{1 + \exp\left(-\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle\right)} \\ \Rightarrow \quad & \frac{r}{1-r} \sum_{i \in \mathcal{S}_{-1}} \frac{z_i}{1 + \exp(z_i)} = \sum_{i \in \mathcal{S}_{-1}} \frac{z_i \mathbb{1}\{\log(1 + e^{-z_i}) > \tau_{-1}\}}{1 + \exp(z_i)} \end{aligned} \tag{D.10}$$

where let $z_i = -\langle \hat{\boldsymbol{w}}, \boldsymbol{x}_i \rangle > 0$. Now observe that when r increases, the left hand side of the above equation will increases since the term $\sum_{i \in \mathcal{S}_{-1}} \frac{z_i}{1+\exp(z_i)}$ is fixed. For the right hand side, when τ_{-1} decreases with other parameters fixed, the term $\sum_{i \in \mathcal{S}_{-1}} \frac{z_i \mathbb{1}\{\log\left(1+e^{-z_i}\right) > \tau_{-1}\}}{1+\exp(z_i)}$ increases. Thus when r increase, τ_{-1} needs to decrease for the equality (D.10) to hold. Finally, it is easy to check that

- If $r \to 0.5$ or $\frac{r}{1-r} \to 1$, $\tau_{-1} \to 0$;
- If $r \to 0$ or $\frac{r}{1-r} \to 0$, then $\mathbb{1}\{\log\left(1+e^{-z_i}\right) > \tau_{-1}\} = 0$. A sufficient condition for $\mathbb{1}\{\log\left(1+e^{-z_i}\right) > \tau_{-1}\} = 0$ is $\tau_{-1} = \log\bigl(1+e^{\|\hat{\boldsymbol{w}}\|}\bigr)$ for normalized data (i.e., $\|\boldsymbol{x}_i\| = 1$).

$\square$

D.5 Experiments

We performed a grid-based hyper-parameter search for initial learning rates for SGD and picked the step-size that achieves the lowest training loss for SGD (Note that B-SGD had no learning rate based hyper-parameter tuning). While the training dataset was imbalanced, the test loss was evaluated over a balanced test set. The following table illustrates the training and test loss performance of SGD and B-SGD vs. the number of gradient computations. It is quite evident that B-SGD achieves significant performance gains in terms of both loss and number of gradient computations over its SGD counterpart.

D.5.1 Synthetic experiments

In this section, we run synthetic experiments on logistic regression on separable data. We assume that the training data is imbalanced however the test distribution is balanced. In this section, we compare the performance of SGD and B-SGD in terms of training error, test error and estimation error vs gradient computations (and time taken) for different values of d. We observe that SGD has high estimation errors demonstrating that it over-fits to the majority class. On the other hand, B-SGD has low estimation error, thus demonstrating its effectiveness in imbalanced classification.

Algorithm	**SGD**				**B-SGD**				**Focal**
Dataset	TL	TEL	TA	GC	TL	TEL	TA	GC	TA
car_eval_4	**0.031**	**0.247**	88.1	$8.64 \cdot e^4$	0.278	0.249	**89.3**	$\mathbf{9.98 \cdot e^3}$	89.2
optical_digits	**0.054**	0.387	89.0	$1.41 \cdot e^5$	1.479	**0.367**	**89.6**	$\mathbf{8.11 \cdot e^3}$	88.0
isolet	**0.035**	0.649	87.0	$1.94 \cdot e^5$	0.485	**0.585**	**88.5**	$\mathbf{8.80 \cdot e^3}$	88.4
letter_img	**0.061**	0.66	83.8	$2.00 \cdot e^5$	1.81	**0.612**	83.8	$\mathbf{3.34 \cdot e^3}$	**84.3**
pen_digits	**0.097**	0.388	83.8	$2.74 \cdot e^5$	0.711	**0.364**	**85.1**	$\mathbf{3.33 \cdot e^4}$	**85.1**
mammography	**0.051**	0.901	70.9	$5.59 \cdot e^4$	0.144	**0.873**	70.5	$\mathbf{2.12 \cdot e^3}$	**71.1**

Table D.1: Comparing training loss (TL), Test Loss (TEL), Test AUC (TA), Top-1 Test Error (TE1), and Number of gradient computations (GC) for SGD and B-SGD over different Imbalanced datasets. The reported results for the first 6 datasets are an average of 5 runs, and for the last 3 datasets are an average of 3 runs. Focal loss (Focal) is the state-of-the-art method proposed in [2], which changes the loss function and so it is not fair to compare the training and the test errors. Focal has the same number of gradient computations as SGD. Hence, we only report test accuracy for Focal.

D.5.2 Real data from imblearn package

We ran some of the experiments described in Table **??** on more datasets and for more iterations and tabulated the results in Table 5.1. Using the same setup as described in the main paper, we run experiments on additional datasets in the imblearn package and further illustrate that using B-SGD reduces the number of gradient computations required as well as improves the generalization performance. To generate the training test split, we use stratified sampling and ensure that the ratio of the number of samples for the

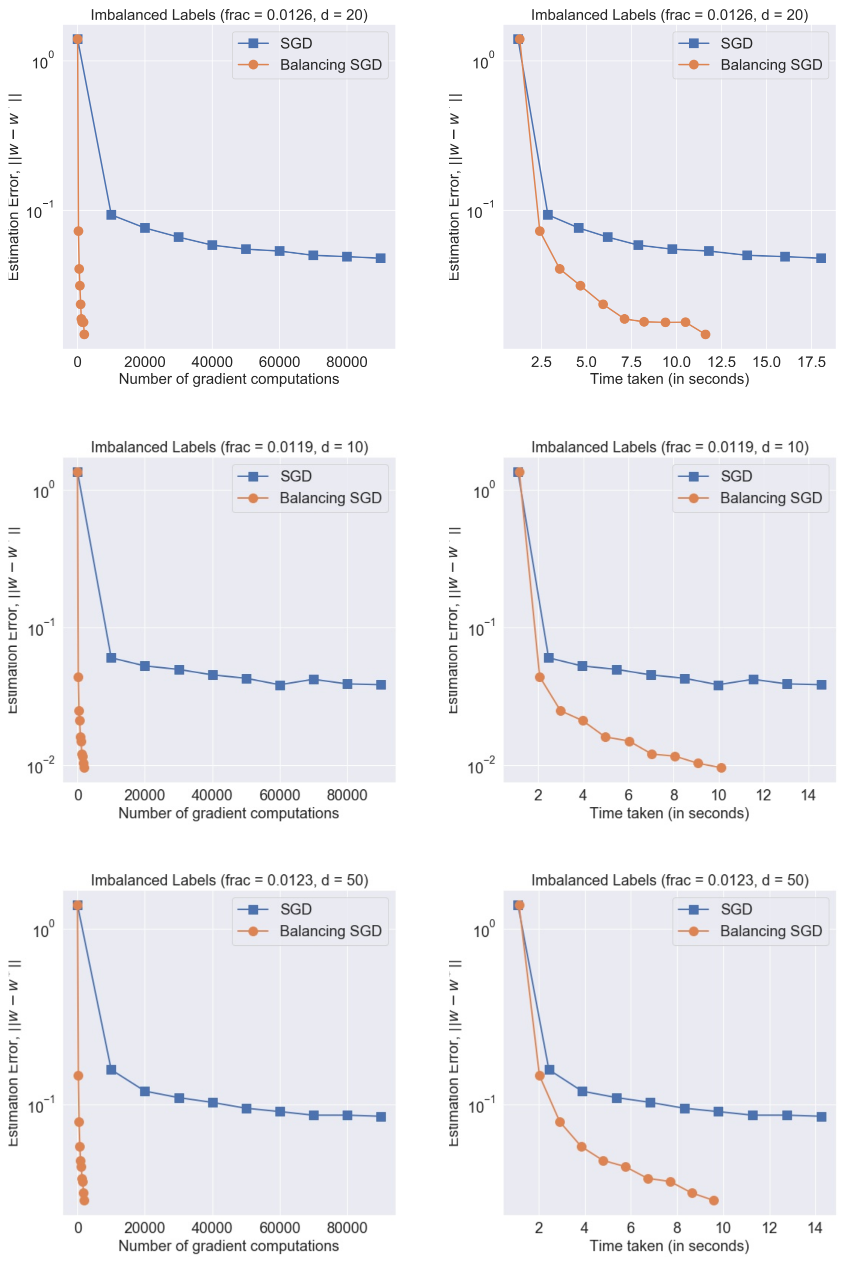


Figure D.1: Comparing the training loss and test loss vs the number of gradient computations for SGD and Balancing SGD for synthetic datasets across $d = \{10, 20, 50\}$

minority class in the training and test set is 3:1. We then randomly sample data from the majority class such that the number of training and test points are identical. Thus, the test set often comprises of around $3 - 5\%$ of the training set depending on the imbalance ratio r. Further information about the datasets including the number of samples, number of feature per samples and the true imbalance ratio is available in the sklearn documentation of the imblearn package [157].

For a given run, we pass the same samples through SGD, B-SGD and Focal Loss. B-SGD evaluates the loss of the incoming samples and determines whether to take an update step while the other two algorithms always take an update step. For evaluation purposes we take the average of 3 runs and calculate the expected moving average of training loss, test loss and test AUC with parameter value of 0.01 for the most recent evaluation. However, label distribution aware methods often show better generalization than B-SGD, however, over significantly more gradient computations.

Lastly, we observe that the variance in training data is high as we train over only one sample at a time. This variance term can be reduced by taking a mini-batch, which is also a practical alternative for large datasets.

D.5.3 Artificially generated imbalanced datasets from real data

D.5.3.1 Hyperparameter Tuning:

For hyper-parameter tuning, we vary the step size as ($\eta \in \{0.1, 0.01, 0.001\}$). Similarly, we use the following step-size decay rule $\eta_t = \frac{\eta}{1+q\cdot t}$ where $q \in$

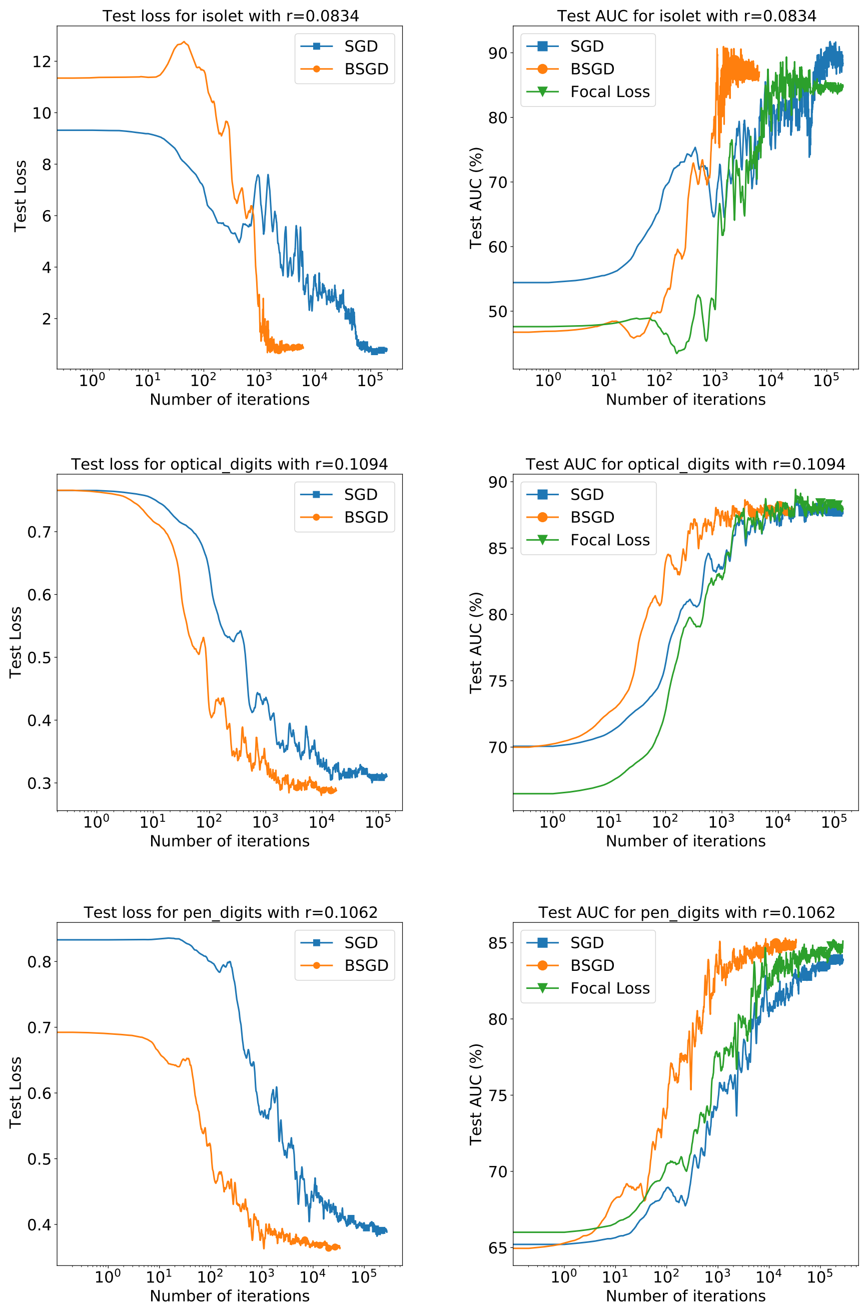


Figure D.2: Comparing the training loss and test loss vs the number of gradient computations for SGD and Balancing SGD for isolet, optical_digits and pen_digits dataset. Each experiment is an average of 5 runs

Algorithm	**SGD**				**B-SGD**				**Focal**
	TL	TEL	TE1	Epochs	TL	TEL	TE1	Epochs	TE1
CIFAR-10	**0.021**	1.43	28.5	200	0.163	**0.90**	**26.2**	**88**	28.7

Table D.2: Comparing training loss (TL), Test Loss (TEL), Test AUC (TA), Top-1 Test Error (TE1), and Number of gradient computations (GC) for SGD and B-SGD over different Imbalanced datasets. The reported results artificially generated imbalanced dataset for CIFAR-10. Focal loss (Focal) is the state-of-the-art method proposed in [2], which changes the loss function and so it is not fair to compare the training and the test errors. Focal has the same number of gradient computations as SGD. Hence, we only report test accuracy for Focal.

$\{0.1, 0.01, 0.001\}$ and select the hyperparameter tuple that optimizes SGD and Focal individually. For B-SGD, the threshold parameter was tuned over the following set, $c \in \{0.1, 0.2, 0.5, 1\}$.

D.5.4 CIFAR-10

We use the experimental setup in [12] and evaluated and compared the performance of B-SGD with respect to SGD and Focal Loss. We observe that in this case also, B-SGD outperforms SGD and Focal Loss in terms of both gradient evaluations and generalization performance. Note, that our method typically outperforms or shows comparable performance to many state of the art methods which are not label distribution aware.

D.5.5 Early Stopping:

Early stopping is a beneficial way to achieve a solution that generalizes well. Initially, the bias reduction caused by B-SGD allows $\boldsymbol{w}$ to get closer to $\boldsymbol{w}_*$. However, if we run the algorithm for too long, then the algorithm starts getting closer and closer to $\hat{\boldsymbol{w}}$ that determines the separating hyper-plane of the training data and away from $\boldsymbol{w}_*$, which in turn decreases the magnitude of improvements achieved by B-SGD over SGD.

D.5.6 Parameter Sensitivity of Threshold Parameter, c

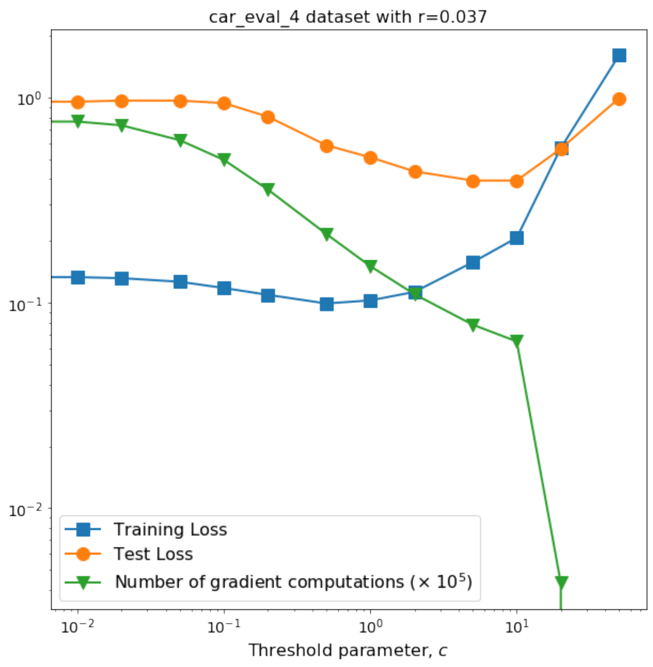


Figure D.3: In this figure, we evaluate the parameter sensitivity of threshold parameter c in Algorithm 3 with respect to the training error, test error and number of gradient computations. We observe that the number of gradient computations is inversely proportional to threshold, while both training and test loss first decrease and then increase as c increases from 0 to 50

The dependence of the generalization behavior (and the number of gradient steps taken) follows a linear increasing behavior up to a particular value beyond which it becomes constant. For $c = 0$, the loss threshold function accepts all samples, and B-SGD is identical to SGD. As c increases, B-SGD starts selectively choosing samples from the majority class leading to gradual improvements in generalization and convergence performance with respect to SGD. However, as $c > c'$ for some c', it is observed that fewer and fewer samples are accepted from majority class until they are not enough to determine the supporting hyper-plane, in turn adversely impacting both generalization and convergence performances. For larger values of c, the generalization will often be worse than SGD. As a result, low values of c, i.e., $c = 1$,are recommended in practice. We also argue that for B-SGD hyper-parameter tuning over c is more important than over learning rate η.

Appendix E

Conclusions

In this dissertation, we propose practical variants of SGD to address its shortcomings with respect to convergence, generalization and robustness.

Bibliography

[1] Causality workbench team. A phamacology dataset, 06 2008.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[3] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[4] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[6] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018.

[8] 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.

[9] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[10] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[11] Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications.* John Wiley & Sons, 2013.

[12] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *arXiv preprint arXiv:1906.07413*, 2019.

[13] Vatsal Shah, Megasthenis Asteris, Anastasios Kyrillidis, and Sujay Sanghavi. Trading-off variance and complexity in stochastic gradient descent. *arXiv preprint arXiv:1603.06861*, 2016.

[14] Ruiliang Zhang, Shuai Zheng, and James T Kwok. Fast distributed asynchronous sgd with variance reduction. *arXiv preprint arXiv:1508.01633*, 2015.

[15] Mark Schmidt and Nicolas Le Roux. Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*, 2013.

[16] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[17] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[18] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, and Quoc V Le. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

[19] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[20] Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Accelerating stochastic gradient descent via online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

[21] Christopher D Sa, Christopher Re, and Kunle Olukotun. Global convergence of stochastic gradient descent for some non-convex matrix problems. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2332–2341, 2015.

[22] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

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

[24] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[25] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[26] Jakub Konecny and Peter Richtarik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[27] Jakub Konevcny, Jie Liu, Peter Richtarik, and Martin Takac. ms2gd: Mini-batch semi-stochastic gradient descent in the proximal setting. *arXiv preprint arXiv:1410.4744*, 2014.

[28] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.

[29] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

[30] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[31] Mark Schmidt. Convergence rate of stochastic gradient with constant step size, Sep 2014.

[32] Angelia Nedić and Dimitri Bertsekas. Convergence rate of incremental subgradient algorithms. In *Stochastic optimization: algorithms and applications*, pages 223–264. Springer, 2001.

[33] Dimitri P Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38, 2011.

[34] Aaron J Defazio, Tibério S Caetano, and Justin Domke. Finito: A faster, permutable incremental gradient method for big data problems. *arXiv preprint arXiv:1407.2710*, 2014.

[35] Julien Mairal. Optimization with first-order surrogate functions. *arXiv preprint arXiv:1305.3120*, 2013.

[36] Zeyuan Allen-Zhu and Yang Yuan. Univr: A universal variance reduction framework for proximal stochastic gradient method. *arXiv preprint arXiv:1506.01972*, 2015.

[37] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. *arXiv preprint arXiv:1506.06840*, 2015.

[38] Jason Lee, Tengyu Ma, and Qihang Lin. Distributed stochastic variance reduced gradient methods. *arXiv preprint arXiv:1507.07595*, 2015.

[39] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[40] Mikhail V Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 11(1):23–35, 1998.

[41] Paul Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[42] Vatsal Shah, Anastasios Kyrillidis, and Sujay Sanghavi. Minimum weight norm models do not always generalize well for over-parameterized problems. *arXiv preprint arXiv:1811.07055*, 2018.

[43] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 2020.

[44] Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, 2020.

[45] M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018.

[46] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[47] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning.* Number 10 in 1. Springer series in statistics New York, 2001.

[48] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[49] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

[50] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[51] Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

[52] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[53] L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[54] G. Orr and K.-R. Müller. *Neural networks: tricks of the trade*. Springer, 2003.

[55] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[56] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *International Conference on Machine Learning*, pages 343–351, 2013.

[57] W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

[58] A. Senior, G. Heigold, and K. Yang. An empirical study of learning rates in deep neural networks for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6724–6728. IEEE, 2013.

[59] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[60] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[61] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[62] M. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[63] T. Tieleman and G. Hinton. Lecture 6.5-RMSPro: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[64] T. Dozat. Incorporating Nesterov momentum into Adam. In *International Conference on Learning Representations*, 2016.

[65] J. Zhang and I. Mitliagkas. YellowFin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*, 2017.

[66] S. Gunasekar, J. Lee, D. Soudry, and N. Srebro. Characterizing implicit bias in terms of optimization geometry. *arXiv preprint arXiv:1802.08246*, 2018.

[67] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

[68] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

[69] Phuong Thi Tran et al. On the convergence proof of amsgrad and a new version. *IEEE Access*, 7:61706–61716, 2019.

[70] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[71] A. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4151–4161, 2017.

[72] M. S. Nacson, J. Lee, S. Gunasekar, N. Srebro, and D. Soudry. Convergence of gradient descent on separable data. *arXiv preprint arXiv:1803.01905*, 2018.

[73] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[74] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[75] D. Dowler. Bounding the norm of matrix powers, 2013.

[76] Xiaoxia Wu, Simon S Du, and Rachel Ward. Global convergence of adaptive gradient methods for an over-parameterized neural network. *arXiv preprint arXiv:1902.07111*, 2019.

[77] Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter?, 2020.

[78] M. C. Mukkamala and M. Hein. Variants of RMSProp and AdaGrad with logarithmic regret bounds. *arXiv preprint arXiv:1706.05507*, 2017.

[79] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[80] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[81] M. Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

[82] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[83] Y. Bansal, M. Advani, D. Cox, and A. Saxe. Minnorm training: an algorithm for training over-parameterized deep neural networks, 2018.

[84] Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[85] Vatsal Shah, Xiaoxia Wu, and Sujay Sanghavi. Choosing the sample with lowest loss makes sgd robust. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2120–2130, Online, 26–28 Aug 2020. PMLR.

[86] Herman Kahn and Andy W Marshall. Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

[87] Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

[88] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9, 2015.

[89] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. *arXiv preprint arXiv:1803.00942*, 2018.

[90] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025, 2014.

[91] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[92] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 147–156. IEEE, 2013.

[93] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[94] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[95] Jan Visek et al. The least weighted squares ii. consistency and asymptotic normality. *Bulletin of the Czech Econometric Society*, 9, 2002.

[96] Jan Amos Visek. The least trimmed squares. part i: Consistency. *Kybernetika*, 42(1):1–36, 2006.

[97] Peter J Huber. *Robust statistics*. Springer, 2011.

[98] Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, pages 721–729, 2015.

[99] Kush Bhatia, Prateek Jain, Parameswaran Kamalaruban, and Purushottam Kar. Consistent robust regression. In *Advances in Neural Information Processing Systems*, pages 2110–2119, 2017.

[100] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pages 5739–5748, 2019.

[101] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

[102] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016.

[103] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60. ACM, 2017.

[104] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018.

[105] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

[106] Farhad Pourkamali Anaraki and Shannon Hughes. Memory and computation efficient pca via very sparse random projections. In *International Conference on Machine Learning*, pages 1341–1349, 2014.

[107] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[108] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[109] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[110] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[111] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

[112] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

[113] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[114] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz Kandola. The perceptron algorithm with uneven margins. In *ICML*, volume 2, pages 379–386, 2002.

[115] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–112, 2019.

[116] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[117] Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*, 2018.

[118] Mor Shpigel Nacson, Nathan Srebro, and Daniel Soudry. Stochastic gradient descent on separable data: Exact convergence with a fixed learning rate. *arXiv preprint arXiv:1806.01796*, 2018.

[119] Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

[120] Jiong Zhang, Hsiang-fu Yu, and Inderjit S Dhillon. Autoassist: A framework to accelerate training of deep neural networks. *arXiv preprint arXiv:1905.03381*, 2019.

[121] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, 2019.

[122] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[123] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[124] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*. Springer, 2018.

[125] CX LING. Data mining for direct marketing: problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining (KDD-98)*, pages 73–79. AAAI Press, 1998.

[126] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[127] Show-Jane Yen and Yue-Shi Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3):5718–5727, 2009.

[128] Gilles Vandewiele, Isabelle Dehaene, György Kovács, Lucas Sterckx, Olivier Janssens, Femke Ongenae, Femke De Backere, Filip De Turck, Kristien Roelens, Johan Decruyenaere, Sofie Van Hoecke, and Thomas

Demeester. Overly optimistic prediction results on imbalanced data: Flaws and benefits of applying over-sampling, 2020.

[129] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer, 2004.

[130] Lara Lusa et al. Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14(1):106, 2013.

[131] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

[132] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[133] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *2016 international joint conference on neural networks (IJCNN)*, pages 4368–4374. IEEE, 2016.

[134] Nabila Abraham and Naimul Mefraz Khan. A novel focal tversky loss function with improved attention u-net for lesion segmentation. In

2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pages 683–687. IEEE, 2019.

[135] Haishuai Wang, Zhicheng Cui, Yixin Chen, Michael Avidan, Arbi Ben Abdallah, and Alexander Kronzer. Predicting hospital readmission via cost-sensitive deep learning. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(6):1968–1978, 2018.

[136] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.

[137] Chong Zhang, Kay Chen Tan, and Ruoxu Ren. Training cost-sensitive deep belief networks on imbalance data problems. In *2016 international joint conference on neural networks (IJCNN)*, pages 4362–4367. IEEE, 2016.

[138] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on knowledge and data engineering*, 18(1):63–77, 2005.

[139] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. Cost-sensitive learning methods for imbalanced data. In *The 2010 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[140] Peng Cao, Dazhe Zhao, and Osmar Zaiane. An optimized cost-sensitive svm for imbalanced data learning. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 280–292. Springer, 2013.

[141] Jing Lu, Peilin Zhao, and Steven CH Hoi. Online passive-aggressive active learning. *Machine Learning*, 103(2):141–183, 2016.

[142] Jonathon Byrd and Zachary C Lipton. What is the effect of importance weighting in deep learning? *arXiv preprint arXiv:1812.03372*, 2018.

[143] Bo Yuan and Xiaoli Ma. Sampling+ reweighting: boosting the performance of adaboost on imbalanced datasets. In *The 2012 international joint conference on neural networks (IJCNN)*, pages 1–6. IEEE, 2012.

[144] Ashish Anand, Ganesan Pugalenthi, Gary B Fogel, and PN Suganthan. An approach for classification of highly imbalanced data using weighting and undersampling. *Amino acids*, 39(5):1385–1391, 2010.

[145] Yan Yan, Tianbao Yang, Yi Yang, and Jianhui Chen. A framework of online learning with imbalanced streaming data. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[146] Yong Zhang and Dapeng Wang. A cost-sensitive ensemble method for class-imbalanced datasets. In *Abstract and applied analysis*, volume 2013. Hindawi, 2013.

[147] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings*

of the 24th international conference on Machine learning, pages 935–942, 2007.

[148] Yu Sui, Xiaohui Zhang, Jiajia Huan, and Haifeng Hong. Exploring data sampling techniques for imbalanced classification problems. In *Fourth International Workshop on Pattern Recognition*, volume 11198, page 1119813. International Society for Optics and Photonics, 2019.

[149] Vatsal Shah, Xiaoxia Wu, and Sujay Sanghavi. Choosing the sample with lowest loss makes sgd robust. *arXiv preprint arXiv:2001.03316*, 2020.

[150] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[151] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[152] Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533, 2011.

[153] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov,

Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[154] Geovani Nunes Grapiglia and Yurii Nesterov. Tensor methods for finding approximate stationary points of convex functions. *arXiv preprint arXiv:1907.07053*, 2019.

[155] Albert B Novikoff. On convergence proofs for perceptrons. Technical report, STANFORD RESEARCH INST MENLO PARK CA, 1963.

[156] Werner Krauth and Marc Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Mathematical and General*, 20(11):L745, 1987.

[157] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[158] Mark D McDonnell and Tony Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE, 2015.

[159] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, and Jakub Konecny. Stop wasting my gradients: Practical SVRG. *To*

appear in Advances in Neural Information Processing Systems, 2015.

[160] 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.

[161] Yadong Mu, Wei Liu, and Wei Fan. Stochastic gradient made stable: A manifold propagation approach for large-scale optimization. *arXiv preprint arXiv:1506.08350*, 2015.

[162] Philipp Moritz, Robert Nishihara, and Michael I Jordan. A linearly-convergent stochastic l-bfgs algorithm. *arXiv preprint arXiv:1508.02087*, 2015.

[163] Richard H Byrd, SL Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *arXiv preprint arXiv:1401.7020*, 2014.

[164] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[165] A. Blum and R. Rivest. Training a 3-node neural network is NP-complete. In *Advances in neural information processing systems*, pages 494–501, 1989.

[166] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar. Theory of deep learning III: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2017.

[167] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[168] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *arXiv preprint arXiv:1806.01811*, 2018.

[169] Qian Qian and Xiaoyuan Qian. The implicit bias of adagrad on separable data. *arXiv preprint arXiv:1906.03559*, 2019.

[170] Samet Oymak and Mahdi Soltanolkotabi. Overparameterized nonlinear learning: Gradient descent takes the shortest path? *arXiv preprint arXiv:1812.10004*, 2018.

[171] Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *arXiv preprint arXiv:1902.04674*, 2019.

[172] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[173] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[174] Samet Oymak and Mahdi Soltanolkotabi. Overparameterized nonlinear learning: Gradient descent takes the shortest path? In *International Conference on Machine Learning*, pages 4951–4960, 2019.

[175] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.

[176] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. *arXiv preprint arXiv:2002.11328*, 2020.

[177] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. *arXiv preprint arXiv:1810.07288*, 2018.

[178] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 795–811, Cham, 2016. Springer International Publishing.

[179] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2348–2358, 2017.

[180] Yuege Xie, Xiaoxia Wu, and Rachel Ward. Linear convergence of adaptive stochastic gradient descent. *arXiv preprint arXiv:1908.10525*, 2019.

[181] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[182] Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *arXiv preprint arXiv:1902.00744*, 2019.

[183] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pages 1002–1012, 2017.

[184] Jie Chen and Ronny Luss. Stochastic gradient descent with biased but consistent gradient estimators. *arXiv preprint arXiv:1807.11880*, 2018.

[185] RA Poliquin and R Tyrrell Rockafellar. Tilt stability of a local minimum. *SIAM Journal on Optimization*, 8(2):287–299, 1998.

[186] Art B Owen. A robust hybrid of lasso and ridge regression, 2007.

[187] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[188] Pranjal Awasthi, Maria Florina Balcan, and Philip M Long. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 449–458, 2014.

[189] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Reviews*, 60(2):223–311, 2018.

[190] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[191] Art B Owen. Infinitely imbalanced logistic regression. *Journal of Machine Learning Research*, 8(Apr):761–773, 2007.

[192] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[193] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[194] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004.

[195] Byungju Kim and Junmo Kim. Adjusting decision boundary for class imbalanced learning. *arXiv preprint arXiv:1912.01857*, 2019.

[196] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, 2000.

Vita

Vatsal Shah received his M.Tech. in Electrical Engineering with a specialization in Communications and Signal Processing and B.Tech in Electronics and Electrical Communication Engineering (with a minor in Center of Studies in Resource Engineering) from the Indian Institute of Technology Bombay in 2014. He is currently working towards Ph.D. in Electrical Engineering at the University of Texas at Austin. His research interests include stochastic optimization, machine learning and graphical models. He has held internship positions at Amazon, Austin, TX, U.S.A. (2017); Amazon Web Services (AWS), Palo Alto, CA, U.S.A. (2017); Technicolor Research, Los Altos, CA, U.S.A. (2016); KTH Royal Institute of Technology, Stockholm, Sweden (2014).

Permanent address: vatsalshah1106@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.