博士論文

Efficient Machine Learning from Gradient Method Perspective in Finite and Infinite Dimensional Spaces

(有限・無限次元空間における勾配法の観点からの効率的機械学習)

二反田　篤史

Abstract

Along with remarkable attention to artificial intelligence technology in recent years, further development of machine learning as its core technology is expected. In order to realize highly accurate and advanced applications by machine learning, not only large-scale datasets and complex modeling but also learning algorithms that work well for them are necessary. In this thesis, we tackle this problem. The main body of the thesis is composed of two parts. In the former part, we develop several efficient stochastic optimization methods for large-scale problems by utilizing their structures such as convexity, smoothness, finite-sum, and difference-of-convex structures. In the latter part, we propose learning methods for optimizing probability measures by extending optimization theory in the case of finite-dimensional spaces to obtain highly accurate models. The other purposes of the part are exploring the connection between optimization and sampling methods and evaluating representational performance of neural networks having the residual structure.

Acknowledgments

I would like to express my gratitude to my brilliant supervisor Taiji Suzuki for great support throughout my journey at Tokyo institute of technology and the University of Tokyo in the doctoral course. His knowledge is very wide and deep, so I got a lot of inspirations from him and I learned many important things and thoughts through research discussions. It certainly will be useful in the future research life. I was also inspired by his active work on the front lines of research. I also would like to greatly thanks Kenji Yamanishi. When I transferred to the University of Tokyo in the doctoral course, I am pleased that he accepted me into his laboratory.

I would like to thank all my dissertation committee members including Taiji Suzuki, Kenji Yamanishi, Takaaki Ohnishi, Akiko Takeda, Tomonari Sei, and Issei Sato for taking time to review my thesis. I appreciate my laboratory members in both Tokyo institute of technology and the University of Tokyo for their support at the laboratory. I would like to thank, particularly, Tomoya Murata and Tetsu Yamashita for developing a line of research from the optimization method presented in Chapter 3. Their work made certain progress in the stochastic optimization literature. I was delighted and impressed. Discussions with Tomoya Morata and Taiji Suzuki about stochastic optimization were quite meaningful.

I would like to thank Takahito Tanabe who was my boss when I worked at NTTDATA Mathematical Systems Inc. for allowing me to proceed to the doctoral course. I am also thankful that he suggested me for a technical adviser when I told him my resignation. I also respect Hiroshi Yamashita who made a significant contribution in mathematical optimization while operating the company. It is thanks to him that I was able to study while working for the company.

I finally would like to thank my family, particularly, parents and wife for all their support and encouragement.

Contents

Chapter 1

Introduction

Artificial intelligence technology has been extensively developed due to rapid improvement in computer performance and expansion of data storage capacity. For instance, the accuracy of such techniques as image, character, and speech recognition are approaching to the same level as human beings, and AI of board games like AlphaGO is growing stronger than the top professional, moreover, vast studies on more sophisticated technologies such as automatic machine translation, data generations, autonomous robots, and self-driving cars are now progressing. Machine learning has been received a lot of attention as a core technology of modern artificial intelligence. Typical goal of machine learning is to find a pattern or construct a function predicting unseen data from a given training data. Various applications are constructed according to the combination of data type and model type. This goal is basically achieved by modeling functions with finite-dimensional parameters in Euclidean spaces and finding good parameters by minimizing the empirical risk with some regularizations, which evaluates model fitness to a training data. Since the risk we really want to optimize is the expected risk which corresponds to an empirical risk with infinite samples almost surely at each parameter, large-scale datasets is required to make the gap between an empirical risk and an expected risk sufficiently small and to obtain good models providing better results. It is known that stochastic optimization methods such as stochastic gradient descent are significantly useful (Bousquet and Bottou, 2008) in solving large-scale problems thanks to cheap computational cost per iteration compared to deterministic optimization methods including steepest descent and Newton method. However, there is much room for improvement over stochastic gradient descent even if problems are convex which is preferable structure for optimization methods. In addition, to obtain much higher accurate results than those of convex problems, complicated modeling of functions is often required, which may be nonconvex and infinite-dimension problems. As a result, optimization for such complex problems become harder than those for finite-dimensional convex problems. Therefore, sophisticated optimization methods are desired for solving such problems efficiently.

In this thesis, we tackle these problems by proposing new optimization methods for each problem setting and verify their effectiveness theoretically and empirically. The thesis will make a first step toward my research mission to unravel the mystery of the success of deep learning with stochastic optimization mathematically from the infinite-dimensional optimization perspective. As for the importance of the optimization, Leonhard Euler mentioned *”Nothing in the world takes place without optimization, and there is no doubt that all aspects of the world that have a rational basis can be explained by optimization methods”* (1774). I think this viewpoint is also valid in explaining machine learning models and methods, which may include stochastic optimization methods not only for learning model parameters but also for learning model structures. This intuition is based on the belief and experience that the solution of the appropriate optimization problem will have good mathematical properties as seen in Dirichlet's principle, Hodge theory, and so on. What kind of theory should be constructed for explaining deep neural networks with the optimization? Since the optimization theory is basically a local theory, it is common that only convergence to a local solution is guaranteed in the learning for deep neural networks. The problem of whether or not the global optimum can be obtained by stochastic optimization methods is really important for unraveling the property of deep learning, hence problem settings and conditions to guarantee such a convergence have been investigated in the literature. This line of research usually make extra assumptions to get closer to the goal, but, conversely, I believe it may be also important to relax the conditions and investigate truly important properties. This is based on the idea that the approach that looks opposite to the purpose is sometimes important as seen in the work by Kiyoshi Oka, a mathematician in the theory of functions of several complex variables, who accomplished a great achievement and succeeded in connecting the super local theory and the global theory. There is also a famous episode that he gave Fields Medalist Heisuke Hironaka this kind of advice. I refer the reader to essays (in Japanese) by Kiyoshi Oka to learn his thoughts. I will summarize my achievement toward my mission at the moment though it is far from their achievement, but believe we will get there someday.

Our contribution is mainly composed of two parts. In part I, we propose efficient stochastic optimization methods for large-scale finite-sum convex problems and difference of convex problems, and show these methods provide faster convergence complexities than standard optimization methods by utilizing these specific structures. In part II, we extend problem settings from parameter optimization in a finite-dimensional space to probability measure optimization in an infinite-dimensional space. Namely, we propose new machine learning methods using functional gradients in infinite-dimensional spaces. Since, this method basically performs in an infinite-dimensional space, it has much more powerful optimization ability than that in a finite-dimensional one. We explain this phenomena intuitively and theoretically and show how to overcome the limitation of finite-dimensional methods by providing convergence analyses for several problem settings. We remark that since the proposed methods are derived by extending optimization methods in a finite-dimensional space in a natural way, it enable us to analyze the behavior of the proposed methods by leveraging the existing optimization theory. Moreover, we believe that this attempt is useful for investigating the performance of specific deep networks and the connection between some optimization methods and sampling methods.

Part I: Stochastic Optimization Methods for Large Scale Problems

Chapter 2. The stochastic gradient descent (Robbins and Monro, 1951) is the workhorse method for large-scale machine learning problems thanks to its scalability, applicability for various problems, simplicity of implementation, and good performance. In chapter 2, we briefly review stochastic gradient descent method to clarify our contributions in Part I over this baseline and its several variants. Stochastic gradient methods were called stochastic approximation in the early days, and analyzed in an asymptotic way (Kushner and Yin, 2003). Although such an analysis is still useful, recently provided non-asymptotic analyses (Bach and Moulines, 2011; Rakhlin et al., 2012; Ghadimi and Lan, 2013b; Bottou et al., 2016) enable us to analyze stochastic gradient methods more directly from traditional optimization viewpoint such as first-order optimization methods mainly developed by Nesterov to derive convergence rates. We first introduce minimization problems of an expected risk and an empirical risk which are main target of stochastic gradient methods in machine learning context. We next introduce convergence criterion for nonconvex and convex problems, and introduce two complexity measures named the total complexity and the iteration complexity under stochastic optimization settings. The theoretical performance of stochastic methods are evaluated by these complexities. Utilizing the above notions, we introduce existing convergence analyses for stochastic convex and nonconvex problems in reference to Rakhlin et al. (2012); Ghadimi and Lan (2013b); Bottou et al. (2016); Johnson and Zhang (2013) to give theoretical comparison with our proposed methods later. We also introduce lower bounds on the complexities of specific algorithm classes for these problems. Finally, we explain the importance of variance reduction for stochastic gradients based on these theoretical results to achieve better complexities.
Chapter 3. In this chapter, we focus on the regularized empirical risk minimization problem and propose a fast stochastic optimization method utilizing the finite-sum structure and acceleration schemes. We first explain the tradeoff between stochastic gradient methods and deterministic gradient descent methods pointed out by Johnson and Zhang (2013). That is, the advantage of stochastic gradient descent over deterministic gradient descent is a cheap computational cost at each iteration. In contrast, deterministic gradient descent achieves the linear convergence rate for strongly convex problems at the price of heavy cost at each iteration. Note that, due to the variance of stochastic gradients, stochastic gradient descent obtains a slower convergence rate. To remedy this issue, Johnson and Zhang (2013) proposed stochastic variance reduced gradient (SVRG) and showed that this techniques leads to the linear convergence. We briefly review these results. How-

ever, it is known that the convergence rate of the gradient descent is not optimal and is slower than accelerated first order methods such as Nesterov (1983, 2004, 2013). Therefore, a natural question arises whether the optimal iteration complexity can be achieved by combining SVRG and such an acceleration method. Nitanda (2014) showed that it is achieved by these two techniques with minibatching of reasonable size, maintaining the same total complexity as SVRG. On the other hand, the optimal total complexity, which is faster than SVRG, was given by Woodworth and Srebro (2016); Arjevani and Shamir (2016) and achieved by several optimal methods (Shalev-Shwartz and Zhang, 2014; Lin et al., 2015; Frostig et al., 2015; Zhang and Xiao, 2017; Allen-Zhu, 2017). However, these methods cannot achieve both optimal complexities simultaneously. More recently, Murata and Suzuki (2017) showed that these optimal complexities are obtained by extending the method (Nitanda, 2014) to doubly accelerated scheme. In this chapter, we show that Acc-Prox-SVRG (Nitanda, 2014) with or without an acceleration technique (Frostig et al., 2015) also achieves the both optimal complexities like DASVRDA (Murata and Suzuki, 2017). A notable common feature of Acc-Prox-SVRG and DASVRDA is minibatching. Furthermore, we show the necessity of minibatching for the optimal iteration complexity by giving lower-bounds on the minibatch size.

Chapter 4. As mentioned above, several effective methods are recently proposed for the smooth finite-sum problems (empirical risk minimization). SAG (Roux et al., 2012; Schmidt et al., 2017) and SAGA (Defazio et al., 2014) are non-accelerated variance reduced methods achieving the same total complexity as SVRG for the strongly convex problems. However, many problems arising in machine learning may be non-strongly convex. An advantage of the SAG and SAGA is that they support general convex problems. More recently, Gong and Ye (2014) showed that SVRG has linear rate of convergence under optimally strong convexity that is a quite weaker condition than the strong convexity. In this chapter, we propose another acceleration method called AMSVRG that incorporates different acceleration scheme from Acc-Prox-SVRG. We show that AMSVRG has the similar feature as Acc-Prox-SVRG even for the optimal strongly convex problems. In addition, we give a direct convergence analysis of AMSVRG for general convex problems.

Chapter 5. There is a strong need to develop better optimization methods for nonconvex problems. Generally speaking, a nonconvex problem is really difficult to solve. However, if the problem possesses a special structure, there is a possibility to construct effective algorithms by making full use of this special structure. For instance, in the previous chapters, we show that acceleration is realized by utilizing finite-sum structure. In this chapter, we consider difference of convex functions (DC) programming (Tao, 1986) as a special structure. DC structure can be often encountered. Indeed, several tasks are formulated as DC programming (see e.g., Argyriou et al. (2006), feature selection in support vector machines by Le Thi et al. (2008)). DC algorithm (DCA) developed by Tao (1986) is the most basic and practical method for solving DC program, which generates a sequence by solving convex sub-problems by linearizing the concave part of objective successively. Due to the simplicity, efficiency, and robustness, DCAs have been widely applied to many fields. In this chapter, we introduce a more suitable variant of DCA called stochastic proximal DC algorithm (SPD) which works effectively not only under a deterministic setting but also a stochastic setting. We give convergence analyses of the method and show how the convergence complexities would be improved by utilizing additional assumptions over vanilla stochastic gradient descent. A main application treated in this chapter is training Boltzman machines (BMs) that are energy-based generative models over binary observations and binary hidden units. Restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs) (Salakhutdinov and Hinton, 2009) are special forms of BMs. These models are used for several purpose including dimension reduction, recommendation, and unsupervised ensemble methods. However, training of RBMs and DBMs is still quite difficult. We show the proposed method can be recognized an extension of expectation-maximization (EM) and Monte Carlo EM (MCEM) algorithms on training BMs. Although the effectiveness of EM algorithms on training BMs was known empirically (e.g., Ikeda (2000); Yasuda and Tanaka (2008); Yasuda et al. (2012)), they lack a convergence analysis. In other words, we extend the EM algorithm on training BMs to a more theoretically better method.

Part II: New Machine Learning Methods using Functional Gradient

Chapter 6 In this thesis, we point out that several problems in machine learning can be formalized as problems of optimizing probability measures. Concretely, we treat three tasks (ensemble method of classifiers, learning adversarial generative models, and functional gradient boosting methods) and explain how these problems can be recognized as optimization of probability measures. We first cast these problems as problems of optimizing transport maps in L_2-space with a given base probability measure to construct computationally tractable methods and we propose variants of functional gradient methods. In this chapter, we introduce the most basic form of these methods by using simplest formalization of the problem. We next give a convergence result by extending a nonconvex analysis introduced in Chapter 2, naturally. Finally, we explain the connection between residual networks (He et al., 2016) which is the state-of-the-art model in computer vision and the proposed method. In other words, analyses in this part can bring theoretical insight into prominent performance of residual networks. We moreover explain the reason why proposed method has a superior optimization ability compared to that in a finite-dimensional space. We here briefly review the literature in this line of research. From the perspective of optimizing the probability measure, gradient-based Bayesian inference methods (Welling and Teh (2011); Dai et al. (2016); Liu and Wang (2016)) are related to ours. Stochastic variational gradient descent (SVGD) proposed in Liu and Wang (2016) is most related to our work. This work has a similar flavor to our method and convergence analysis from gradient flow perspective were also given in Liu (2017); Chen and Zhang (2017). However, SVGD is specialized to minimizing the Kullback–Leibler-divergence. In contrast, our method does not require special structure of a loss function and can be applicable to a wider class of problems as shown in this thesis. We also remark the connection with a normalizing flow (Rezende and Mohamed (2015)) that approximates Bayes posterior through deep neural networks. Like the normalizing flow, our method constructs a deep residual network as transport maps.

Chapter 7. In this chapter, we consider ensemble learning problems with the L_1-regularization in an infinite-dimensional space of base classifiers; in other words, optimization problems of probability measures to sample base classifiers contained in an ensemble. Several methods (Schapire et al., 1998; Mason et al., 2000; Friedman, 2001; Bengio et al., 2006; Bach, 2014) for this problem have been proposed due to its better statistical performance (Schapire et al., 1998; Koltchinskii and Panchenko, 2002; Bartlett et al., 2006). Like boosting methods, these methods adopt the strategy that base classifiers are added iteratively in a greedy fashion to avoid the difficulty of handling L_1-regularization in an infinite-dimensional space. In spite of the success of boosting methods in data analysis, it seems to be difficult to apply these methods for large-scale base classifiers such as deep neural networks because the above methods require solving nonconvex subproblems for adding base classifiers, which can become intractable. In this chapter, we propose a new method called Stochastic Particle Gradient Descent (SPGD) for learning ensemble. Our method adopt a completely different strategy as essentially explained in Chapter 6. This strategy is in opposition to existing methods that successively increase the number of basis to be combined and easily executable compared to those. In the theoretical analysis, we show the convergence of the proposed method by adapting the result described in Chapter 6 to this problem setting. As a result, good convergence property as fast as that of a stochastic optimization method for finite-dimensional nonconvex problems is obtained. We also show the interior optimality property of the method which characterizes an optimality condition considered in this chapter. We finally provide two practical variants of SPGD method.

Chapter 8. Generative adversarial networks (GANs) (Goodfellow et al., 2014) are considered as a promising scheme for learning generative models. GANs are composed of two networks called a discriminator and a generator. These networks are trained in an adversarial way. Generators generate samples to mimic real samples, whereas discriminators classify real samples and fake samples. Due to the success of generating high quality images by GANs (Radford et al., 2016), many variants of GANs were proposed (Larsen et al., 2016; Salimans et al., 2016; Nowozin et al., 2016; Chen et al., 2016; Zhang et al., 2017). However, difficulty of training GANs also widely known. Some reasons for the difficulty are high nonconvexity of an objective function and the limited representational ability of the generator. In this chapter, we propose a new learning procedure to alleviate the issues of limited representational power and local optimum by introducing a new type

of layer called a gradient layer. The gradient layer finds a direction of improvement in an infinite-dimensional space by computing the functional gradient. As shown in Chapter 6, since the functional gradient is not limited in the tangent space of a finite-dimensional model, it has much more freedom and it can break the limit of the local optimum induced by a finite-dimensional model. We theoretically justify this phenomenon from the functional gradient method perspective and the property of Wasserstein distance. SteinGAN (Wang and Liu, 2016) is closely related to our work and has a similar flavor. However, it adopts different strategy for tracking gradient flow from our methods.
Chapter 9. Since neural networks are highly flexible, an architecture search of networks is very important. Indeed, several studies tackle this problem (Zoph and Le, 2017; Liu et al., 2017, 2018; Pham et al., 2018). One way of architecture search (Bengio et al., 2006; Moghimi et al., 2016; Cortes et al., 2017; Huang et al., 2017a) is based on boosting theory where parts of neural networks are considered as weak learners and they are added in a greedy way like boosting. In particular, Huang et al. (2017a) is interesting and related to our work presented in this chapter, which explores residual network architecture with respect to depth using boosting theory. Residual networks (He et al., 2016) have achieved great success in classification tasks as mentioned before. A notable feature of residual networks is a special layer structure called skip-connection introduced by He et al. (2016) to avoid the vanishing gradient problem, Thanks to their great performance, several variants of residual networks were proposed (Zagoruyko and Komodakis, 2016; Xie et al., 2017; Huang et al., 2017b). Moreover, several theoretical studies have been devoted to reveal a reason of their success and analyze the structure of residual networks. There are mainly two thoughts: one is the ensemble view and the other is the optimization view based on ordinary differential equation. These viewpoints are reminiscent of gradient boosting methods (Mason et al., 1999; Friedman, 2001) which are known to be most useful in data analysis competitions. Although residual networks and gradient boosting are state-of-the-art methods in different domains, there is an interesting similarity like this. However, there are several differences between these two methods. In this chapter, we propose a new gradient boosting method called ResFGB for classification tasks based on these observations with theoretical guarantees. That is, the feature extraction gradually grows by functional gradient methods in the space of feature extractions and a classifier having residual network-like architecture is obtained. The expected benefit of the method over existing gradient boosting methods is representational ability due to a deep architecture rather than a shallow model. Indeed, superior performance of the proposed method is verified in experiments.

Organization of the Thesis

The main body of this thesis is composed of our several works. Papers related to this thesis are listed below.

- Chapter 3 is an extended work from *Stochastic Proximal Gradient Descent with Acceleration Techniques*, A. Nitanda, Neural Information Processing Systems, 2014 (Nitanda, 2014).
- Chapter 4 is based on *Accelerated Stochastic Gradient Descent for Minimizing Finite Sums*, A. Nitanda, Artificial Intelligence and Statistics, 2016 (Nitanda, 2016).
- Chapter 5 is based on *Stochastic Difference of Convex Algorithm and its Application to Training Deep Boltzmann Machines*, A. Nitanda and T. Suzuki, Artificial Intelligence and Statistics, 2017 (Nitanda and Suzuki, 2017a).
- Chapter 7 is based on *Stochastic Particle Gradient Descent for Infinite Ensembles*, A. Nitanda and T. Suzuki, preprint, 2017 (Nitanda and Suzuki, 2017b).
- Chapter 8 is based on *Gradient Layer: Enhancing the Convergence of Adversarial Training for Generative Models*, A. Nitanda and T. Suzuki, Artificial Intelligence and Statistics, 2018 (Nitanda and Suzuki, 2018b).
- Chapter 9 is based on *Functional Gradient Boosting based on Residual Network Perception*, A. Nitanda and T. Suzuki, International Conference on Machine Learning, 2018 (Nitanda and Suzuki, 2018a).

Chapter 2 is a brief literature review of stochastic gradient methods to clarify the contribution of our work presented in Chapter 3, 4, and 5. In Chapter 6, a functional gradient descent which is a commonly used notion in Chapters 7, 8, and 9 is explained with its theoretical property which leads to deeper understanding of Part II.

The relationship between chapters is depicted in Figure 1.1.

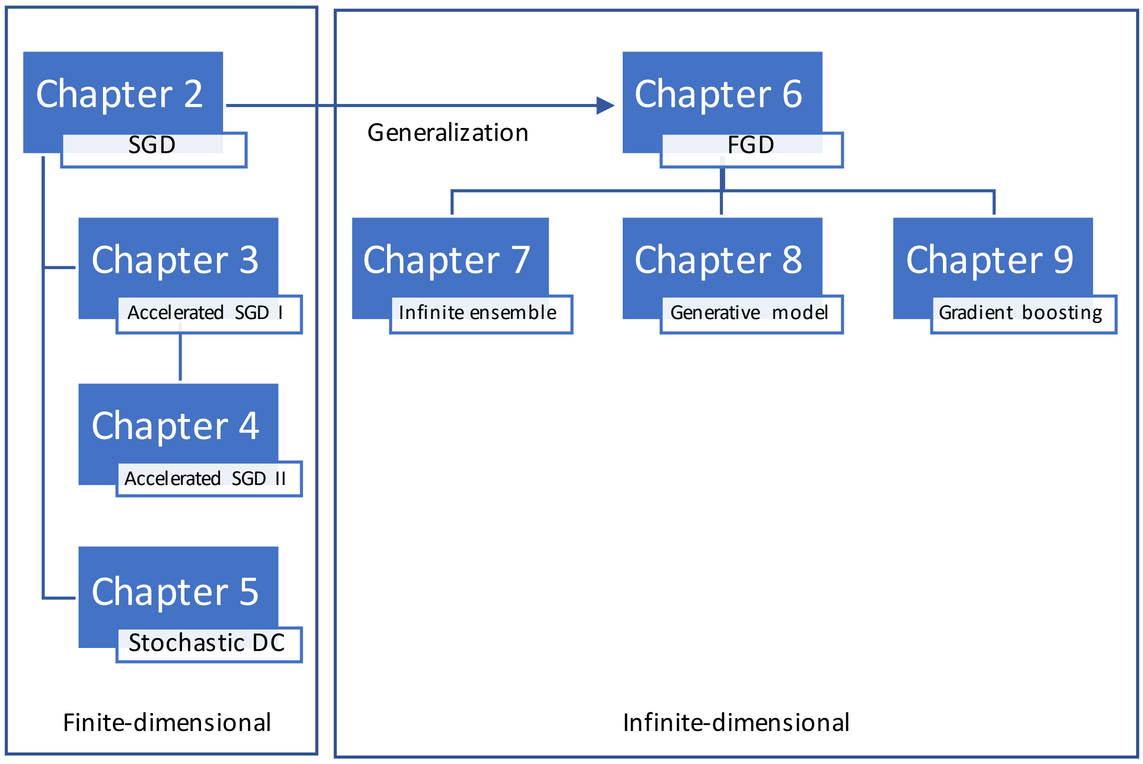

Figure 1.1: The organization of the thesis. FGD is the abbreviation for functional gradient descent

Part I

Stochastic Optimization Methods for Large Scale Problems

Chapter 2

Stochastic Gradient Descent

Stochastic gradient descent (Robbins and Monro, 1951) method is the most popular and useful stochastic optimization method for large-scale machine learning problems. In this chapter, we briefly introduce several properties of stochastic gradient descent and several important notions used frequently in this thesis. We first give problem settings to be solved in machine learning. The ultimate goal of most machine learning problems is to solve *expected risk* minimization problems. Let $g(x, \xi)$ be a loss function that represents fitness between models and data; smaller is better, where $x \in \mathbb{R}^d$ is a parameter and ξ is a random variable usually corresponds to a data. For example, for given data $a \in \mathbb{R}^d$ and label $b \in \mathbb{R}$, if we set $g(x,(a,b)) = \frac{1}{2}(a^\top x - b)^2$, then we obtain regression problem. If we set $g(x,(a,b)) = \log(1+\exp(-bx^\top a))$, then we obtain regularized logistic regression. The expected risk minimization problem is defined as follows:

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \overset{\mathrm{def}}{=} \mathbb{E}[g(x,\xi)] \right\}, \tag{2.1}$$

where $\mathbb{E}$ denotes the expectation with respect to ξ.

Although stochastic gradient descent can be applied for expected risk minimization problems under appropriate settings, this problem is also sometimes intractable depending on an underlying distribution of ξ. Therefore, we often approximate an expected risk using finite-samples $(\xi_i)_{i=1}^n$ obtained independently from ξ. Namely, we use an *empirical risk* which is the average of $g(x,\xi)$ as a surrogate function and solve the following empirical risk minimization problems with some regularization. We simply denote $g_i(x) = g(x,\xi_i)$.

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \overset{\mathrm{def}}{=} \frac{1}{n}\sum_i g_i(x) + h(x) \right\}, \tag{2.2}$$

where $h : \mathbb{R}^d \to$ is called *regularization function* that is used to prevent overfitting (Bousquet and Elisseeff, 2002; Mukherjee et al., 2003; Steinwart and Christmann, 2008; Shalev-Shwartz and Ben-David, 2014). Typical choices of h is $h(x) = \frac{\lambda}{2}\|x\|_1$ ($\|\cdot\|_1$ is the

L_1-norm) and $h(x) = \frac{\lambda}{2}\|x\|_2^2$. We call these regularizers L_1-regularization function and L_2-regularization function, respectively.

Stochastic gradient descent is the iterative method for solving both problems (2.1) and (2.2). Note that although we now explain stochastic gradient descent method for expected risk minimization problems (2.1), we can extend it to empirical risk minimization problems (2.2) in an obvious way. We assume $g(x, \xi)$ is differentiable with respect to x and we denote its partial derivative by $\partial_x g(x, \xi)$. Let k denote the iteration index, x_k be the current iterate, and ξ_k be the random variable having the same distribution as ξ. In this thesis, we assume $\{\xi_k\}_k$ are independent. In stochastic gradient descent method, we randomly choose the value ξ_k of ξ at iteration k and move the parameter x_k along the direction $\partial_x g(x_k, \xi_k)$ with sufficiently small step size η_k which is called learning rate. That is, the update of stochastic gradient descent is described as follows:

$$x_{k+1} \leftarrow x_k - \eta_k \partial_x g(x_k, \xi_k).$$

To introduce the convergence property of stochastic gradient descent, we give the important notion of Lipschitz smoothness which is frequently used in this thesis. Let $\langle, \rangle_2$ and $\|\cdot\|_2$ denote Euclidean inner-product and norm, respectively.

Definition 1 (Lipschitz smoothness). *Let $g : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. We call that g is L-smooth if there exists a positive value $L > 0$ such that $\forall x, \forall y \in \mathbb{R}^d$,*

$$\|\nabla g(x) - \nabla g(y)\| \leq L\|x - y\|_2.$$

The Lipschitz smoothness implies very useful inequality for analyzing iterative optimization methods. As for its proof, see Nesterov (2004).

Proposition 1. *Let $g : \mathbb{R}^d \to \mathbb{R}$ be an L-Lipschitz smooth function. Then, for $\forall x, \forall y \in \mathbb{R}^d$, we get*

$$|g(y) - g(x) - \langle \nabla g(x), y - x\rangle_2| \leq \frac{L}{2}\|x - y\|_2^2.$$

The convexity of functions is preferable for optimization methods because it may lead to the global convergence property and faster convergence rate than that for nonconvex problems. We here introduce the definition of convexity.

Definition 2 (Convexity). *Let $\mathcal{W} \in \mathbb{R}^d$ is a convex set and $g : \mathcal{W} \to \mathbb{R}$ be a real valued function. We call that g is convex if $\forall x, \forall y \in \mathcal{W}$ and $\forall t \in [0, 1]$,*

$$g(tx + (1-t)y) \leq tg(x) + (1-t)g(y).$$

Moreover, we call that g is μ-strongly convex if there exists a positive value $\mu \geq 0$ such that $\forall x, \forall y \in \mathcal{W}$ and $\forall t \in [0, 1]$,

$$g(tx + (1-t)y) \leq tg(x) + (1-t)g(y) - \frac{1}{2}\mu t(1-t)\|x - y\|_2^2.$$

When g is differentiable, the condition of strong convexity is equivalent to the following:

$$g(x) + \langle \nabla g(x), y - x \rangle_2 + \frac{\mu}{2}\|y - x\|_2^2 \leq g(y) \;\; (\forall x, \forall y \in \mathcal{W}),$$

When a function g is L-Lipschitz smooth and μ-strongly convex, the ratio L/μ called condition number is defined, which is very important quality because this number usually affects the convergence speed of gradient-based optimization methods. For more useful inequalities of Lipschitz smooth function and (strongly) convex functions, we refer the reader to Nesterov (2004).

2.1 Convergence Criterion and Complexity

Due to the difficulty of obtaining global minima of nonconvex problems, the expected gradient norm $\mathbb{E}\|\nabla f(x_k)\|_2^2$, where the expectation is taken with respect to random variables in a stochastic optimization method, is adopted usually as a convergence criterion (Ghadimi and Lan, 2013b) for nonconvex problems. Therefore, convergence criterion for nonconvex problems is described as follows: for a given threshold $\epsilon > 0$,

$$\mathbb{E}\|\nabla f(x_k)\|_2^2 \leq \epsilon.$$

On the other hand, for convex problems, the objective gap $f(x) - f_*$, where $f_* = \inf_{x \in \mathbb{R}^d} f(x)$, is adopted as a convergence criterion because global convergence is guaranteed in several optimization methods, that is, convergence criterion for convex problems is described as follows: for a given threshold $\epsilon > 0$,

$$\mathbb{E}[f(x_k) - f_*] \leq \epsilon.$$

The performance of stochastic optimization methods is evaluated by a cost spent to find an ϵ-accurate solution in terms of the above criterion. There are two types of costs commonly used in this literature; the first is *total complexity* that is the number of gradient evaluations and the second is *iteration complexity* that is the number of updates of the parameter. In this thesis, we use both types of complexities. We note that these qualities coincide each other for the vanilla stochastic gradient descent, thus, we say simply the complexity for such a case.

2.2 Convergence Analysis for Nonconvex Problem

Under Lipschitz smoothness assumption, a convergence analysis with several learning rate strategies for nonconvex problems is provided in Ghadimi and Lan (2013b). We introduce their result with a constant learning strategy in the following theorem.

Theorem 1 (Ghadimi and Lan (2013b)). *Let $g(x,\xi)$ be L-Lipschitz smooth with respect to x and the variance of stochastic gradient $\partial_x g(x,\xi)$ is uniformly bounded by $\sigma^2 > 0$, that is, $\mathbb{E}\|\partial_x g(x,\xi) - \nabla f(x)\|_2^2 \le \sigma^2$ for $\forall x \in \mathbb{R}^d$. Consider running stochastic gradient descent for T iterations with a constant learning rate $\eta_k = \eta > 0$ from an initial point $x_0 \in \mathbb{R}^d$. We assume $f_* = \inf_{x\in\mathbb{R}^d} f(x) > -\infty$. Then, if $\eta \le 1/L$, it follows that*

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x_k)\|_2^2 \le \frac{2}{\eta T}(f(x_0) - f_*) + \frac{\eta L \sigma^2}{2}.$$

By taking the sufficiently small learning rate, we can get the total complexity immediately from this theorem.

Corollary 1. *Let us make the same assumption as Theorem 1. If we set a constant learning rate $\eta = \min\left\{\frac{1}{L}, \frac{\epsilon}{L\sigma^2}\right\}$. Then, if $T \ge \frac{4L}{\epsilon}(f(x_0) - f_*)\max\left\{1, \frac{\sigma^2}{\epsilon}\right\}$, then we get*

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x_k)\|_2^2 \le \epsilon.$$

This corollary states that the total complexity of stochastic gradient descent to obtain an ϵ-accurate solution is at most $O(L\sigma^2/\epsilon^2)$. In Chapter 5, we further investigate nonconvex problems by utilizing specific structure called difference of convex (Tao, 1986).

2.3 Convergence Analysis for Convex Problem

There are mainly two types of convergence proofs of stochastic gradient descent for convex problems; whether it relies on the smoothness or not. The smoothness does not always accelerate the order of convergence rate in general stochastic optimization (Tsybakov, 2003) except for specific functions, (for instance, Bach and Moulines (2013) provides convergence rates $O(1/T)$ for the square and logistic losses without strong convexity) but the proof based on Proposition 1 allows us to be free from a boundedness assumption on gradient norms. In other words, proofs not relying on the smoothness for strongly convex problems are valid only for the projected stochastic gradient descent introduced later, to ensure the boundedness of gradients.

Since a convergence complexity of the order $O(1/\epsilon^2)$ for general convex problems can be obtain immediately from Corollary 1, we introduce a convergence analysis of order $O(1/\epsilon)$ for strongly convex problems. We also note that following convergence rates of stochastic gradient descent lead to those for non-strongly convex problems by regularizing techniques, moreover, recently proposed successively regularizing techniques (Allen-Zhu and Hazan, 2016) leads to those without $\log(1/\epsilon)$ factor unlike normal regularizing strategy. For more direct analyses of projected stochastic gradient descent in the case of general convex problems using the gradient boundedness, we refer the reader to Nemirovski et al. (2009); Bubeck (2015).

Theorem 2 (Bottou et al. (2016))**.** *Let $g(x,\xi)$ be L-Lipschitz smooth with respect to x and $f(x)$ be μ-strongly convex. We assume that the variance of stochastic gradient $\partial_x g(x,\xi)$ is uniformly bounded by $\sigma^2 > 0$, that is, $\mathbb{E}\|\partial_x g(x,\xi) - \nabla f(x)\|_2^2 \le \sigma^2$ for $\forall x \in \mathbb{R}^d$. We assume $f_* = \inf_{x\in\mathbb{R}^d} f(x) > -\infty$. Consider running stochastic gradient descent for T iterations from an initial point $x_0 \in \mathbb{R}^d$.*
(i) If we use a constant learning rate $\eta_k = \eta > 0$ satisfying $\eta \le 1/L$, then

$$\mathbb{E}[f(x_{T-1}) - f_*] \le \frac{\eta L\sigma^2}{2\mu} + (1-\eta\mu)^{T-1}\left(f(x_0) - f_* - \frac{\eta L \sigma^2}{2\mu}\right).$$

(ii) If we use diminishing learning rates $\eta_k = \frac{\beta}{\gamma+k+1}$ for some $\beta > 1/\mu$ and $\gamma > 0$, satisfying $\eta_0 \le 1/L$, then

$$\mathbb{E}[f(x_{T-1}) - f_*] \le \frac{\nu}{\gamma + T},$$

where

$$\nu \overset{\mathrm{def}}{=} \max\left\{\frac{L\beta^2\sigma^2}{2(\beta\mu - 1)}, (\gamma+1)(f(x_0) - f_*)\right\}.$$

From this theorem, we can immediately derive a corollary concerning complexities by setting (i) $\eta = \min\left\{\frac{\mu\epsilon}{L\sigma^2}, \frac{1}{L}\right\}$ for the former part and (ii) $\beta = 2/\mu$ and $\gamma = 2L/\mu - 1$ for the latter part. We denote the condition number by $\kappa = L/\mu$.

Corollary 2. *Let us make the same assumption as Theorem 2.*
(i) If we set a constant learning rate $\eta_k = \min\left\{\frac{\mu\epsilon}{L\sigma^2}, \frac{1}{L}\right\}$, then complexities to obtain ϵ-accurate solution is

$$1 + \kappa \max\left\{1, \frac{\sigma^2}{\mu\epsilon}\right\} \log \frac{L(f(x_0) - f_*)}{\epsilon}.$$

(ii) If we set diminishing learning rates $\eta_k = 2/(2\kappa + k)\mu$, then complexities to obtain ϵ-accurate solution is

$$\frac{2\kappa}{\epsilon}\max\left\{\frac{2\sigma^2}{\mu}, f(x_0) - f_*\right\}.$$

We next introduce convergence analysis not relying on the smoothness. Since analyses of this type require the boundedness of gradients, the projected version of stochastic gradient descent is essential and slightly better result than the above is achieved. Let $\mathcal{W} \subset \mathbb{R}^d$ be a convex domain. Then, the update of projected stochastic gradient descent is described as follows:

$$x_{k+1} \leftarrow \Pi_{\mathcal{W}}\left(x_k - \eta_k \partial_x g(x_k, \xi_k)\right),$$

where $\Pi_{\mathcal{W}}$ is the projection operator onto a convex domain $\mathcal{W}$. In Rakhlin et al. (2012), the convergence rate $O(1/T)$ of projected stochastic gradient descent with *α-suffix averaging* is obtained for strongly convex problems without the smoothness. The α-suffix averaging is defined by

$$\overline{x}_T^\alpha = \frac{x_{(1-\alpha)T+1} + \cdots + x_T}{\alpha T},$$

for a constant $\alpha \in (0,1)$ where we simply assume that αT and $(1-\alpha)T$ are integers.

Theorem 3 (Rakhlin et al. (2012))**.** *Let $f(x)$ be μ-strongly convex and assume $\mathbb{E}\|\partial_x g(x,\xi)\|_2^2 \leq G^2$. We assume $f_* = \inf_{x\in\mathcal{W}} f(x) > -\infty$. Consider running projected stochastic gradient descent for T iterations with an initial point $x_0 \in \mathbb{R}^d$. Then, we get*

$$\mathbb{E}[f(\overline{x}_{T-1}^\alpha) - f_*] \leq \frac{2 + 2.5\log\left(\frac{1}{1-\alpha}\right)}{\alpha}\frac{G^2}{\mu T}. \tag{2.3}$$

We note that similar results to this theorem was given in Lacoste-Julien et al. (2012); Bubeck (2015). Moreover, the rate of $O(1/T)$ was also obtained in Nemirovski et al. (2009) for strongly convex problems when the optimal point is an interior in $\mathcal{W}$, but it requires both the boundedness assumption on gradients and Lipschitz smoothness. A result concerning a complexity is obtained as follows.

Corollary 3. *Let us make the same assumption as Theorem 3. Then, a complexity to obtain ϵ-accurate solution by α-suffix projected stochastic gradient descent is*

$$\frac{2 + 2.5\log\left(\frac{1}{1-\alpha}\right)}{\alpha}\frac{G^2}{\mu\epsilon}.$$

2.4 Optimal Complexity for Stochastic Convex Problems

According to Nemirovskii and Yudin (1983); Agarwal et al. (2009), optimal complexities to obtain ϵ-accurate solutions are

$$O\left(\sqrt{\frac{L}{\epsilon}} + \frac{\sigma^2}{\epsilon^2}\right) \tag{2.4}$$

in the case of stochastic convex problems and

$$O\left(\sqrt{\frac{L}{\mu}}\log\left(\frac{L}{\epsilon}\right) + \frac{\sigma^2}{\mu\epsilon}\right) \tag{2.5}$$

in the case of stochastic strongly convex problems. Note that the above rates are essentially composed of a deterministic (bias) and stochastic (variance) optimization terms.

Indeed, these rates correspond to deterministic optimal rates were obtained in Nesterov (1983, 2004) when $\sigma = 0$ and it is known that accelerated gradient methods (Nesterov, 1983, 2004) achieve these optimal rates.

In a stochastic optimization case, we notice that stochastic gradient methods achieve (asymptotic) optimal rates from the previous analyses. A non-asymptotic analysis was originally provided by Bach and Moulines (2011). More precisely they showed that stochastic gradient descent with a constant small learning rate achieves an optimal rate for stochastic strongly convex problems though it may be unstable and the Polyak-Ruppert averaging (Ruppert, 1988; Polyak and Juditsky, 1992) variant achieves optimal rates for both stochastic strongly convex and non-strongly convex problems.

Although stochastic gradient descent methods are basically asymptotic optimal in terms of only stochastic terms which are dominant in (2.4) and (2.5), there are some methods that achieve optimal rates in terms of both deterministic and stochastic terms. Such methods are called uniformly optimal (Nemirovskii and Yudin, 1983). For instance, AC-SA (Ghadimi and Lan, 2013a) and ORDA (Chen et al., 2012) and their multi-stage variants (Chen et al., 2012; Ghadimi and Lan, 2013a) are uniformly optimal.

For the non-strongly convex least squares regression problems, a much improved optimal complexity was shown. It can be obtained by combining the optimal deterministic optimization term (Nesterov, 1983) and the optimal stochastic term (Tsybakov, 2003):

$$O\left(\sqrt{\frac{1}{\epsilon}} + \frac{\sigma^2}{\epsilon}\right). \tag{2.6}$$

We notice that this bound achieves a rate of $O(1/\epsilon)$ without strong convexity. In Bach and Moulines (2013), asymptotic optimal complexity of averaging stochastic gradient descents without strong convexity was shown, but the deterministic term was $O(1/\epsilon)$ which is slower than that of (2.6). More recently, the uniform optimal complexity (2.6) was achieved by accelerated stochastic averaged gradient descent (Dieuleveut et al., 2017).

2.5 Variance Reduction Methods

As argued earlier, convergence rates of stochastic optimization methods are essentially sum of the deterministic and stochastic optimization terms. Thus, variance reduction techniques are important to accelerate the convergence speed of these methods. The simplest method is minibatching that is a simple modification of the methods. While in vanilla stochastic gradient methods, deterministic gradients are approximated by stochastic gradients consisting of a single instance, in minibatch methods, stochastic gradients are computed by multiple instances to reduce the variance of estimators. One benefit due to minibatching is speeding up by parallel computing of stochastic gradients. In this viewpoint, total computing time is almost proportional to the iteration complexity and its theoretical limit corresponds to the deterministic optimization (bias) term in its complexity. Since the stochastic term is much slower than deterministic term usually, very large minibatch size is required to achieve this limit for sufficiently small ϵ. For example, the iteration complexity of the uniformly optimal methods with minibatch size b is

$$O\left(\sqrt{\frac{L}{\mu}}\log\left(\frac{L}{\epsilon}\right)+\frac{\sigma^2}{\mu b\epsilon}\right),$$

thus, minibatch size for reaching the optimal iteration complexity is

$$O\left(\frac{\sigma^2}{\mu\epsilon}\left(\sqrt{\frac{L}{\mu}}\log\left(\frac{L}{\epsilon}\right)\right)^{-1}\right).$$

This size is unrealistic for a small precision ϵ. Therefore, estimating sufficient minibatch size to achieve the optimal iteration complexity is an important problem. We tackle this problem in Chapter 3 and 4 by combining other variance reduction techniques.

Chapter 3

Accelerated Variance Reduced Stochastic Gradient Descent I

Proximal gradient descent and stochastic proximal gradient descent are popular methods for solving regularized risk minimization problems in machine learning and statistics. In this chapter, we propose and analyze an accelerated variant of these methods in the mini-batch setting. That is, the method incorporates three techniques: Nesterov's acceleration method, a variance reduction for the stochastic gradient, and minibatching. We show that our method can achieve both the optimal total and iteration complexity simultaneously by utilizing these techniques. Furthermore, we show the necessity of minibatching for the optimal iteration complexity by giving lower-bounds on the minibatch size.

This chapter is based on the work *Stochastic Proximal Gradient Descent with Acceleration Techniques*, A. Nitanda, Neural Information Processing Systems, 2014 (Nitanda, 2014).

3.1 Overview

In this chapter, we focus on the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = g(x) + h(x) \right\}, \tag{3.1}$$

where g is the average of the smooth convex functions $g_1, \ldots, g_n$ from $\mathbb{R}^d$ to $\mathbb{R}$, i.e., $g(x) = \frac{1}{n}\sum_{i=1}^{n} g_i(x)$ and $h : \mathbb{R}^d \to \mathbb{R}$ is a convex function. We assume that h can be non-differentiable, but *relatively simple* which means that *proximal operator* can be computed efficiently. Typical examples of h are L_1-regularization function $\lambda\|x\|_1$ and L_2-regularization function $\frac{\lambda}{2}\|x\|_2^2$, where $\lambda > 0$ is a regularization parameter. Since h may be non-differentiable, we consider (stochastic) proximal gradient descent rather than vanilla (stochastic) gradient descent in this chapter. The definitions of the proximal operator and (stochastic) proximal gradient descent will be given in the next section.

The advantage of stochastic proximal gradient descent over its deterministic variant is that at each iteration, it only requires the computation of a single gradient $\nabla g_{i_k}(x_k)$. In contrast, each iteration of proximal gradient descent evaluates the n gradients. Thus the computational cost of stochastic gradient per iteration is $1/n$ that of the exact gradient. However, due to the variance introduced by random sampling, stochastic proximal gradient descent obtains a slower convergence rate as shown in the previous chapter.

Under the smoothness and the strong convexity assumptions, (proximal) gradient descent with a constant learning rate $\eta_k = \frac{1}{L}$, where L is the smoothness parameter, achieves a linear convergence rate. On the other hand, for stochastic (proximal) gradient descent, because of the variance introduced by random sampling, we need to choose diminishing learning rate $\eta_k = O(1/k)$ or small learning rate depending on the required optimization accuracy ϵ, and thus the stochastic (proximal) gradient descent converges at a sub-linear rate.

To improve the stochastic (proximal) gradient descent, we need a variance reduction technique, which allows us to take a larger learning rate. Recently, several studies proposed such variance reduction methods for the various special cases of (3.1). In the case where $g_i(x)$ is Lipschitz smooth and $h(x)$ is strongly convex, Shalev-Shwartz and Zhang (2012, 2013b) proposed a proximal stochastic dual coordinate ascent (Prox-SDCA); the same authors developed accelerated variants of SDCA (Shalev-Shwartz and Zhang, 2013a, 2014). Roux et al. (2012) proposed a stochastic average gradient (SAG) for the case where $g_i(x)$ is Lipschitz smooth, $g(x)$ is strongly convex, and $h(x) \equiv 0$. These methods achieve a linear convergence rate. However, SDCA and SAG need to store all gradients (or dual variables), so that $O(nd)$ storage is required in general problems. Although this can be reduced to $O(n)$ for linear prediction problems, these methods may be unsuitable for more complex and large-scale problems. More recently, Johnson and Zhang (2013) proposed stochastic variance reduction gradients (SVRG) for the case where $g_i(x)$ is L-Lipschitz smooth, $g(x)$ is μ-strongly convex, and $h(x) \equiv 0$. SVRG achieves the following total complexity (total number of component gradient evaluations to find an ϵ-accurate solution) and the iteration complexity (the number of iterations to find an ϵ-accurate solution)

$$O\left((n+\kappa)\log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\kappa\log\left(\frac{1}{\epsilon}\right)\right),$$

respectively, where κ is the condition number L/μ. Note that this method need not store all gradients unlike SAG, Furthermore, Xiao and Zhang (2014) proposed a proximal variant of SVRG called Prox-SVRG and showed that it achieves the same complexity as SVRG.

However, this iteration complexity is slower than the optimal iteration complexity (Nesterov, 2004):

$$O\left(\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right)\right).$$

We can show that this complexity is not reached by SVRG even if utilizing the minibatch because SVRG is an stochastic variant of deterministic gradient descent.

On the other hand, the following lower bounds on the total complexity has been shown for the problem (3.1) (Agarwal and Bottou, 2014; Woodworth and Srebro, 2016; Arjevani and Shamir, 2016).

$$\Omega\left(n + \sqrt{n\kappa}\log\left(\frac{1}{\epsilon}\right)\right).$$

It is well known that the optimal iteration complexities is achieved by deterministic acceleration methods (Nesterov, 2004, 2005, 2013; Allen-Zhu and Orecchia, 2014). As for the optimal total complexity, several studies (Shalev-Shwartz and Zhang, 2014; Lin et al., 2015; Frostig et al., 2015; Zhang and Xiao, 2017; Allen-Zhu, 2017) have proposed the optimal methods in terms of this complexity.

However, these methods cannot achieve both optimal complexities simultaneously. More recently, an exceptional method was proposed by Murata and Suzuki (2017), namely, they showed that these optimal complexities are obtained by extending Acc-Prox-SVRG (Nitanda, 2014) to doubly accelerated scheme.

In this chapter, we first introduce the method *Accelerated Mini-Batch Prox-SVRG* (Acc-Prox-SVRG), which is originally proposed in a short version of this work (Nitanda, 2014). Acc-Prox-SVRG incorporates two acceleration techniques in the mini-batch setting: Nesterov's acceleration method (Nesterov, 2004) and an variance reduction technique of SVRG (Johnson and Zhang, 2013). We show that the total and iteration complexities of this method with reasonable minibatch size $O(\sqrt{\kappa})$ are

$$O\left((n+\kappa)\log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right)\right).$$

Moreover, we show that Acc-Prox-SVRG with APPA (Frostig et al., 2015) using the *optimal minibatch* size $O(\sqrt{n})$ achieves the following total and iteration complexities.

$$O\left((n+\sqrt{n\kappa})\log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right)\right).$$

Namely, the optimal total complexity and the optimal iteration complexity are achieved up to a logarithmic factor simultaneously like DASVRDA (Murata and Suzuki, 2017). A notable common feature of our method and DASVRDA is minibatching. In this chapter, we show the necessity of minibatching by providing a lower-bound on the minibatch size for achieving the optimal iteration complexity, and show that these two methods achieve it by the optimal minibatch size.

A short version of this work have been published at NIPS 2014 (Nitanda, 2014). Extensions from the conference paper are: (i) both the optimal total and iteration complexities are achieved by applying APPA to Acc-Prox-SVRG with the efficient minibatch size $O(\sqrt{n})$ in the case of $\kappa > n$, (ii) we show that this minibatch is a lower bound to achieve the optimal iteration complexity.

3.2 Preliminary

In this section, we introduce some notions, assumptions, and methods used in this chapter. Since h can be non-differentiable, vanilla gradient descent and stochastic gradient descent are not directly applied to the problem (3.1) as mentioned earlier. Thus, we first introduce the proximal operator and proximal variants of (stochastic) gradient descent.

The definition of the proximal operator is

$$\mathrm{prox}_{\eta h}(y) = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \left\{ \frac{1}{2}\|x - y\|^2 + \eta h(x) \right\}.$$

In (stochastic) proximal gradient descent, the updated is constructed by the composition of (stochastic) gradient step and the proximal operator. Namely, proximal gradient descent is performed as follows; at iteration $k = 1, 2, \ldots$,

$$x_{k+1} = \mathrm{prox}_{\eta_k h}\left(x_k - \eta_k \nabla g(x_k)\right),$$

and stochastic proximal gradient descent is performed as follows; at iteration $k = 1, 2, \ldots$, we pick i_k randomly from $\{1, 2, \ldots, n\}$, and take the following update:

$$x_{k+1} = \mathrm{prox}_{\eta_k h}\left(x_k - \eta_k \nabla g_{i_k}(x_k)\right).$$

In this chapter, we make the following assumptions to provide a convergence analysis.

Assumption 1. *Let L and μ be positive values such that $L \geq \mu$. We assume that each convex function $g_i(x)$ is L-Lipschitz smooth and $g(x)$ is μ-strongly convex, that is, it follows that*

$$g_i(x) \leq g_i(y) + \langle \nabla g_i(y), x - y \rangle_2 + \frac{L}{2}\|x - y\|^2,$$
$$g(x) \geq g(y) + \langle \nabla g(y), x - y \rangle_2 + \frac{\mu}{2}\|x - y\|^2.$$

Assumption 2. *The regularization function $h(x)$ is a lower semi-continuous proper convex function; however, it can be non-differentiable or non-continuous.*

3.2.1 Stochastic Variance Reduction Gradient

In this section, we briefly review stochastic variance reduction gradient (SVRG) (Johnson and Zhang, 2013). To ensure the convergence of stochastic gradient descent, the learning rate must decay to zero so that we reduce the variance effect of the stochastic gradient. This slows down the convergence. Variance reduction techniques (Johnson and Zhang, 2013; Xiao and Zhang, 2014; Konečný and Richtárik, 2013; Konečný et al., 2016) such as SVRG have been proposed to solve this problem. We introduce prox-SVRG in a minibatch setting. Prox-SVRG is a multi-stage scheme. During each stage, this method performs m-iterations of stochastic proximal gradient descent using the following direction,

$$v_k = \nabla g_{I_k}(x_k) - \nabla g_{I_k}(\tilde{x}) + \nabla g(\tilde{x}),$$

where $\tilde{x}$ is a starting point at stage, k is an iteration index, $I_k = \{i_1, \ldots, i_b\}$ is a uniformly randomly chosen size b subset of $\{1, 2, \ldots, n\}$, and $f_{I_k} = \frac{1}{b}\sum_{j=1}^{b} f_{i_j}$. Note that v_k is an unbiased estimator of gradient $\nabla g(x_k)$: $\mathbb{E}_{I_k}[v_k] = \nabla g(x_k)$, where $\mathbb{E}_{I_k}$ denotes the expectation with respect to I_k. The overall procedure of SVRG is summarized in Algorithm 1.

Algorithm 1 Proximal Stochastic Variance Reduced Gradient (Prox-SVRG)

```
Input: the number of outer-iterations $T$, the number of inner-iterations $m$, learning rate
$\eta$, mini-batch size $b$, and initial point $\tilde{x}_1$
for $s = 1$ to $T$ do
  $\tilde{x} \leftarrow \tilde{x}_s$
  $\tilde{v} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \nabla g_i(\tilde{x})$
  $x_1 \leftarrow \tilde{x}$
  for $k = 1$ to $m$ do
    Randomly pick subset $I_k \subset \{1, 2, \ldots, n\}$ of size $b$
    $v_k \leftarrow \nabla g_{I_k}(y_k) - \nabla g_{I_k}(\tilde{x}) + \tilde{v}$
    $x_{k+1} \leftarrow \mathrm{prox}_{\eta h}(y_k - \eta v_k)$
  end for
  $\tilde{x}_{s+1} \leftarrow \frac{1}{m}\sum_{k=2}^{m+1} x_m$
end for
Return $\tilde{x}_{T+1}$
```

Under the smoothness assumption, it is shown that variances of v_k approach to zero as optimization proceeds and SVRG with $b = 1$ achieves total complexity of $O((n+\kappa)\log\frac{1}{\epsilon})$ and iteration complexity of $O(\kappa \log \frac{1}{\epsilon})$ for the strongly convex problems, where κ is the condition number (for the proof, see Johnson and Zhang (2013); Xiao and Zhang (2014)).

3.2.2 Accelerated Proximal Gradient Descent

In this section, we briefly introduce accelerated proximal gradient descent (APG) (Nesterov, 2004). In APG, two sequences of parameters are updated rather than one sequence like usual gradient descent, to accelerate the convergence speed of gradient descent method. Indeed, for Lipschitz-smooth and strongly convex function g, it is shown that APG achieves the optimal iteration complexity of $O(\sqrt{\kappa}\log(1/\epsilon))$, where κ is the condition number. We note that the proof of this convergence rate is essentially included in that of our method proposed in the next section. The overall procedure of APG is provided in Algorithm 2.

Algorithm 2 Accelerated Proximal Gradient Descent (APG)

```
**Input:** the number of iterations $T$, learning rate $\eta$, non-negative sequence $\beta_1, \ldots, \beta_m$, and initial point $x_1$
$y_1 \leftarrow x_1$
**for** $s = 1$ **to** $T$ **do**
    $v_s \leftarrow g_{I_s}(y_s)$
    $x_{s+1} \leftarrow \mathrm{prox}_{\eta h}(y_s - \eta v_s)$
    $y_{s+1} \leftarrow x_{s+1} + \beta_s(x_{s+1} - x_s)$
**end for**
Return $x_{T+1}$
```

3.2.3 APPA Acceleration

In this section, we introduce a technique APPA (Frostig et al., 2015) that is a meta-algorithm to accelerate an existing optimization methods. For instance, in the case of $\kappa > n$, the total complexities of SAG, SAGA, MISO, and SVRG are improved from $O(\kappa \log(1/\epsilon))$ to $O(\sqrt{n\kappa}\log(1/\epsilon))$, that is, they reach the optimal total complexity up to a logarithmic factor. In APPA, the objective function 3.1 with a proximal term at a current iterate is approximately minimized successively by using an existing method and an acceleration step is taken. The overall scheme of APPA is described in Algorithm 3. We denote by F_s^* the infimum $\inf_{x\in\mathbb{R}^d} F_s(x)$, where F_s is defined in Algorithm 3.

The following lemma is key result to derive a total complexity of an optimization method accelerated by APPA. Concretely, Lemma 1 gives an iteration complexity of Algorithm 3 ignoring a complexity of an inner-solver $\mathcal{M}$.

Lemma 1 (Frostig et al. (2015))**.** *Suppose that f is μ-strongly convex and $\gamma \geq 3\mu/2$. Consider Algorithm 3. If a generated sequence $\{w_s\}_{s=1}^{T_c+1}$ satisfies the following*

$$\mathbb{E}[F_s(w_{s+1})] - F_s^* \leq \frac{1}{4}\left(\frac{\mu}{\mu+2\gamma}\right)^{\frac{3}{2}}(F_s(w_s) - F_s^*). \tag{3.2}$$

Then, there exists a positive value C_0 which only depend on w_0 such that

$$\mathbb{E}[f(w_s)] - f_* \leq \left(1 - \frac{1}{2}\sqrt{\frac{\mu}{\mu+2\gamma}}\right)^s C_0. \tag{3.3}$$

From this lemma, we can get the complexity of APPA for the case of $\gamma \geq 3\mu/2$:

$$O\left(\sqrt{\frac{\gamma}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right). \tag{3.4}$$

Algorithm 3 APPA

```
Input: the number of iterations $T_c$, strong convexity $\mu$, positive parameter $\gamma > 2\mu$
inner-solver $\mathcal{M}$, the number of iterations $T_{\mathcal{M}}$ for $\mathcal{M}$, and initial point $w_1$
$\rho = \frac{\mu+2\gamma}{\mu}$
$\zeta = \frac{2}{\mu} + \frac{1}{\gamma}$
$v_1 \leftarrow w_1$
for $s = 1$ to $T_c$ do
    $z_s \leftarrow \frac{1}{1+\rho^{-1/2}} w_s + \frac{\rho^{-1/2}}{1+\rho^{-1/2}} v_s$
    Solve the following problem approximately by running $\mathcal{M}$ starting from $w_s$ for $T_{\mathcal{M}}$-iterations:
    $w_{s+1} \leftarrow \arg\min_{x \in \mathbb{R}^d} F_s(x) \overset{\text{def}}{=} f(x) + \frac{\gamma}{2}\|x - z_s\|^2$
    $g_s \leftarrow \gamma(z_s - w_{s+1})$
    $v_{s+1} \leftarrow (1 - \rho^{-1/2}) v_s + \rho^{-1/2}(y_s - \zeta g_s)$
end for
Return $w_{T_c+1}$
```

Note that although we here incorporate APPA, Catalyst in Lin et al. (2015, 2017) which is another proximal acceleration scheme also has a similar property.

3.3 Accelerated Mini-Batch Prox-SVRG

In this section, we give a detailed description of the proposed method called *accelerated proximal SVRG (Acc-Prox-SVRG)*. Since we are attempting to get the optimal iteration complexity with reasonable minibatch size, we incorporate (Nesterov, 2004) and prox-SVRG (Xiao and Zhang, 2014). Acc-Prox-SVRG is a multi-stage scheme like prox-SVRG. During each stage, this method performs m-iterations of APG by using variance reduced stochastic gradient with minibatch:

$$v_k = \nabla g_{I_k}(y_k) - \nabla g_{I_k}(\tilde{x}) + \nabla g(\tilde{x}), \tag{3.5}$$

where $I_k = \{i_1, \ldots, i_b\}$ is a randomly chosen size b subset of $\{1, 2, \ldots, n\}$ and $g_{I_k} = \frac{1}{b}\sum_{j=1}^{b} g_{i_j}$. At the beginning of each stage, the initial point x_1 is set to be $\tilde{x}$, and at the end of stage, $\tilde{x}$ is updated. Conditioned on y_k, we can take expectation with respect to I_k and obtain $\mathbb{E}_{I_k}[v_k] = \nabla g(y_k)$, so that v_k is an unbiased estimator. As described in the next section, the conditional variance $\mathbb{E}_{I_k}\|v_k - \nabla g(y_k)\|^2$ can be much smaller than $\mathbb{E}_i\|\nabla g_i(y_k) - \nabla g(y_k)\|^2$ near the optimal solution. The overall procedure of Acc-Prox-SVRG is given in Algorithm 4.

In our analysis, we focus on a basic variant of Algorithm 4 with $\beta_k = \frac{1-\sqrt{\mu\eta}}{1+\sqrt{\mu\eta}}$. We note that Acc-Prox-SVRG can be further accelerated by applying APPA in an obvious way: running Algorithm 3 using Algorithm 4 as an inner-solver $\mathcal{M}$.

Algorithm 4 Acc-Prox-SVRG

```
Input: the number of outer-iterations $T$, the number of inner-iterations $m$, learning rate
$\eta$, mini-batch size $b$, non-negative sequence $(\beta_k)_{k=1}^m$, and initial point $\tilde{x}_1$
for $s = 1$ to $T$ do
    $\tilde{x} \leftarrow \tilde{x}_s$
    $\tilde{v} \leftarrow \frac{1}{n}\sum_{i=1}^n \nabla g_i(\tilde{x})$
    $x_1 = y_1 \leftarrow \tilde{x}$
    for $k = 1$ to $m$ do
        Randomly pick subset $I_k \subset \{1, 2, \ldots, n\}$ of size $b$
        $v_k \leftarrow \nabla g_{I_k}(y_k) - \nabla g_{I_k}(\tilde{x}) + \tilde{v}$
        $x_{k+1} \leftarrow \mathrm{prox}_{\eta h}(y_k - \eta v_k)$
        $y_{k+1} \leftarrow x_{k+1} + \beta_k(x_{k+1} - x_k)$
    end for
    $\tilde{x}_{s+1} \leftarrow x_{m+1}$
end for
Return $\tilde{x}_{T+1}$
```

Table 3.1: The best achievable total and iteration complexities.

Algorithm	Condition	b, m	η	Total complexity	Iteration complexity
Acc-Prox-SVRG	$n \geq \kappa$	$O(\sqrt{\kappa})$	$O\left(\frac{1}{L}\right)$	$O\left(n \log\left(\frac{1}{\epsilon}\right)\right)$	$O\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$
with APPA	$n < \kappa$	$O(\sqrt{n})$	$O\left(\frac{1}{L-\mu}\right)$	$\tilde{O}\left(\sqrt{n\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$	$\tilde{O}\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$

3.4 Analysis

In this section, we present our analysis of the convergence rates of Algorithm 4 and it with APPA under Assumptions 1 and 2. All missing proofs are provided in the Appendix of this chapter. We summarize the best achievable complexities derived in this section in Table 3.1. Note that the optimal total and iteration complexities are obtained simultaneously up to a logarithmic factor. The notation $\tilde{O}$ hides logarithmic terms of μ and γ (parameter of APPA).

We first give useful notion *estimate sequence* which was developed to prove fast convergence rate of APG (Nesterov, 2004). In the following analysis, we may omit the outer index s for notational simplicity. By the definition of a proximity operator, there exists a subgradient $\xi_k \in \partial h(x_{k+1})$ such that

$$x_{k+1} = y_k - \eta\left(v_k + \xi_k\right).$$

We define the estimate sequence $\Phi_k(x)$ $(k = 1, 2, \ldots, m+1)$ by

$$\begin{aligned}\Phi_1(x) &= f(x_1) + \frac{\mu}{2}\|x - x_1\|^2\\ \Phi_{k+1}(x) &= (1 - \sqrt{\mu\eta})\Phi_k(x) + \sqrt{\mu\eta}\{g_{I_k}(y_k) + \langle v_k, x - y_k\rangle_2 + \frac{\mu}{2}\|x - y_k\|^2\\ &\quad + h(x_{k+1}) + \langle \xi_k, x - x_{k+1}\rangle_2\}, \quad \mathit{for}\ k \geq 1.\end{aligned}$$

We set

$$\Phi_k^* = \min_{x\in\mathbb{R}^d} \Phi_k(x) \ \ \mathit{and} \ \ z_k = \arg\min_{x\in\mathbb{R}^d} \Phi_k(x).$$

Since $\nabla^2\Phi_k(x) = \mu I_n$, it follows that for $\forall x \in \mathbb{R}^d$,

$$\Phi_k(x) = \frac{\mu}{2}\|x - z_k\|^2 + \Phi_k^*. \tag{3.6}$$

The following lemma is the key to the analysis of our method.

Lemma 2. *Consider Algorithm 4 under Assumptions 1 and 2. If $\eta \leq \frac{1}{2L}$, then for $k \geq 1$ we have*

$$\mathbb{E}\left[\Phi_k(x)\right] \leq f(x) + (1 - \sqrt{\mu\eta})^{k-1}(\Phi_1 - f)(x), \tag{3.7}$$

$$\mathbb{E}\left[f(x_k)\right] \leq \mathbb{E}\left[\Phi_k^* + \sum_{l=1}^{k-1}(1 - \sqrt{\mu\eta})^{k-1-l}\left\{-\frac{\mu}{2}\frac{1-\mu\eta}{\sqrt{\mu\eta}}\|x_l - y_l\|^2 + \eta\|\nabla g(y_l) - v_l\|^2\right\}\right] \tag{3.8}$$

where the expectation is taken with respect to the history of random variables $I_1, \ldots, I_{k-1}$.

Note that if the conditional variance of v_l is equal to zero, we immediately obtain a linear convergence rate from inequalities (3.7) and (3.8). Our bound on the variance of v_k is given in the following lemma.

Lemma 3. *Consider Algorithm 4 under Assumption 1. Let $x_* = \arg\min_{x\in\mathbb{R}^d} f(x)$. Conditioned on y_k, we have that*

$$\mathbb{E}_{I_k}\|v_k - \nabla g(y_k)\|^2 \leq \frac{1}{b}\frac{n-b}{n-1}\left(2L^2\|y_k - x_k\|^2 + 8L(f(x_k) - f(x_*) + f(\tilde{x}) - f(x_*))\right).$$

Utilizing these lemmas, we can give the convergence rate of Algorithm 4.

Theorem 4. *Consider Algorithm 4. Suppose Assumption 1 and 2 hold. Let $\eta \leq \min\left\{\frac{(pb)^2}{64}\left(\frac{n-1}{n-b}\right)^2\frac{\mu}{L^2}, \frac{1}{2L}\right\}$ and $0 < p < 1$. Then we have*

$$\mathbb{E}\left[f(\tilde{x}_{s+1}) - f(x_*)\right] \leq \left((1 - (1-p)\sqrt{\mu\eta})^m + \frac{p}{1-p}\right)(2+p)(f(\tilde{x}_s) - f(x_*)). \tag{3.9}$$

Moreover, if $m \geq \frac{1}{(1-p)\sqrt{\mu\eta}} \log \frac{1-p}{p}$, *then it follows that*

$$\mathbb{E}\left[f(\tilde{x}_{s+1}) - f(x_*)\right] \leq \frac{2p(2+p)}{1-p}(f(\tilde{x}_s) - f(x_*)). \tag{3.10}$$

From Theorem 4, we can see that for small $0 < p$, the total and iteration complexities of Acc-Prox-SVRG are (total number of component gradient evaluations to find an ϵ-accurate solution) is

$$O\left(\left(n + \frac{b}{\sqrt{\mu\eta}}\right) \log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\frac{1}{\sqrt{\mu\eta}} \log\left(\frac{1}{\epsilon}\right)\right).$$

Thus, we have the following corollary:

Corollary 4. *Consider Algorithm 4. Suppose Assumption 1 and 2. Let* p *be sufficiently small, as stated above, and* $\eta = \min\left\{\frac{(pb)^2}{64}\left(\frac{n-1}{n-b}\right)^2 \frac{\mu}{L^2}, \frac{1}{2L}\right\}$. *If mini-batch size* b *is smaller than* $\left\lceil \frac{8\sqrt{\kappa}n}{\sqrt{2}p(n-1)+8\sqrt{\kappa}} \right\rceil$, *then the learning rate* η *is equal to* $\frac{(pb)^2}{64}\left(\frac{n-1}{n-b}\right)^2 \frac{\mu}{L^2}$ *and the total and iteration complexities are*

$$O\left(\left(n + \frac{n-b}{n-1}\kappa\right) \log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\frac{n-b}{n-1}\frac{\kappa}{b} \log\left(\frac{1}{\epsilon}\right)\right).$$

Otherwise, $\eta = \frac{1}{2L}$ *and the total and iteration complexities become*

$$O\left(\left(n + b\sqrt{\kappa}\right) \log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right).$$

We denote $b_0 = \frac{8\sqrt{\kappa}n}{\sqrt{2}p(n-1)+8\sqrt{\kappa}}$ and note that $b_0 = O\left(\min\{n, \sqrt{\kappa}\}\right)$. From Corollary 4, the total and iterations complexities of Algorithm 4 decrease monotonically with respect to b when $b < \lceil b_0 \rceil$ and the total complexity increases monotonically when $b \geq \lceil b_0 \rceil$. Moreover, if $b = 1$, then Algorithm 4 has the same complexities as that of Prox-SVRG, while if $b = n$ then the complexities of this method are equal to those of APG. Therefore, with an appropriate mini-batch size b_0, Algorithm 4 may outperform both Prox-SVRG and APG. Even if the mini-batch is not appropriate, then Algorithm 4 is still comparable to Prox-SVRG or APG. The following total and overall complexity are achieved Algorithm 4 by setting $b = b_0$,

$$O\left(\left(n + \min\{\kappa, n\sqrt{\kappa}\}\right) \log\left(\frac{1}{\epsilon}\right)\right) \text{ and } O\left(\min\{\sqrt{\kappa}, n\} \log\left(\frac{1}{\epsilon}\right)\right).$$

Therefore, we see that the optimal total and iteration complexities are achieved simultaneously up to a logarithmic factor by Algorithm 4 with an efficient minibatch size of $b = O(\sqrt{\kappa})$ when $n \geq \kappa$.

We next derive complexities of Algorithm 4 with APPA from results introduced in Section 3.2.3 in the case of $n < \kappa$. To do so, we specify complexities of Algorithm 4 for solving subproblems with required accuracy (3.2) by APPA. Since $g_i(x) + \frac{\gamma}{2}\|x - z_s\|^2$ is $L+\gamma$-smooth and $g(x) + \frac{\gamma}{2}\|x - z_s\|^2$ is $\mu+\gamma$-strongly convex, the condition numbers of subproblems are $\kappa' \overset{\mathrm{def}}{=} \frac{L+\gamma}{\mu+\gamma}$. We assume $\gamma > 3\mu/2$ below.

By choosing p to be satisfied $2p(2+p)/1-p \leq 1/2$, we see that the objective F_s can be halved by running Algorithm 4 with the setting: $b = O(\min\{n, \sqrt{\kappa'}\})$, $\eta = O(1/(L+\gamma))$, and $m = O(\sqrt{\kappa'})$, Moreover, we note that the number of outer iteration of Algorithm 4 to satisfy requirement (3.2) is $O(\log(\gamma/\mu))$. Thus, by combining these observations and (3.4), we can immediately obtain an iteration complexity of Algorithm 4 with APPA:

$$\tilde{O}\left(\sqrt{\kappa + \frac{\gamma}{\mu}} \log\left(\frac{1}{\epsilon}\right)\right).$$

Setting $\gamma = \frac{L}{n} - \mu$ under the ill-conditioned case of $n < \kappa$, we see $b = O(\sqrt{n})$, $\eta = O(1/(L-\mu))$, and $m = O(\sqrt{n})$. Moreover, we conclude that the optimal total and iteration complexities are achieved simultaneously up to a logarithmic factor when $n < \kappa$ by Algorithm 4 with APPA under these settings,

$$\tilde{O}\left(\sqrt{n\kappa} \log\left(\frac{1}{\epsilon}\right)\right) \text{ and } \tilde{O}\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right).$$

3.4.1 Fast Iteration Complexity and Necessary Minibatch size

As mentioned above, Nesterov's acceleration achieves the best iterations complexity (Nesterov, 2004) for deterministic optimization problems. As for a specific class of stochastic optimization methods, we can also confirm that the same lower bound on the iteration complexity is held. We first consider the class $\mathcal{A}$ of first-order stochastic optimization methods that correspond to deterministic methods when the variance of stochastic gradients are zero. Then, for any method $\mathcal{M} \in \mathcal{A}$, we can choose an objective function g such that the deterministic variant of $\mathcal{M}$ needs the iteration complexity $O(\sqrt{\kappa}\log(1/\epsilon))$ to minimize g. Thus, considering the empirical minimize problem derived by setting $g_i = g$ for all $i \in \{1, \ldots, n\}$, the same lower bound is provided for the class $\mathcal{A}$.

However, unlike deterministic optimization methods, this lower bound cannot be obtained by direct application of Nesterov's acceleration to vanilla stochastic methods without the variance reduction. Moreover, we observe Nesterov's acceleration with either SVRG with minibatch size of 1 or only reasonable size minibatching is not enough to achieve the optimal iteration complexity from the analysis in the previous section and Cotter et al. (2011). Namely, under stochastic setting, we observe that (i) Nesterov's acceleration + small mini-batching may have the almost same convergence rate as that of SGD as indicated by Cotter et al. (2011), (ii) Nesterov's acceleration + SVRG without mini-batching has the same iteration complexity as SVRG, as shown in this chapter, (iii) mini-batching + SVRG has the almost same iteration complexity as deterministic gradient descent because SVRG is a stochastic variant of it. On the other hand, by combining three techniques: Nesterov's acceleration, mini-batching, and SVRG, our method achieves the optimal iteration complexity with the efficient minibatch size $O(\sqrt{\kappa})$ or $O(\sqrt{n})$, as shown in the previous section. These observations leads to the conjecture that we need not only SVRG but also mini-batching to obtain sufficiently small variance for the acceleration scheme. We support this observations for a specific algorithm class rigorously by giving a lower bound on the minibatch size for achieving the optimal iteration complexity. Let b be a positive integer and $\mathcal{A}$ be a set of first-order stochastic optimization methods satisfying the following properties.

- The total complexity is lower-bounded by $\Omega(n+\sqrt{n\kappa}\log(1/\epsilon))$. For instance, such a class is introduced in Arjevani and Shamir (2016), which includes major variance reduced methods SAG, SAGA, SVRG, and these accelerated variants by APPA and Catalyst.
- Achieving the optimal iteration complexity $O(\sqrt{\kappa}\log(1/\epsilon))$.
- The order of the total complexity is equal to minibatch size b times the iteration complexity.

We remark that our proposed method and notable methods such as SAG, SAGA, SVRG, APPA, and Catalyst using sufficiently large minibatch of size b are included this class $\mathcal{A}$. Moreover, we remark that when $n \geq \kappa$, Acc-Prox-SVRG with $b = O(\sqrt{\kappa})$ does not satisfy the last property in the above list, but it will be satisfied if resetting $b \leftarrow \tilde{O}(n/\sqrt{\kappa})$.

For an arbitrarily method $\mathcal{M}$, we can choose an empirical risk such that it takes the total complexity of $\Omega(\sqrt{n\kappa}\log(1/\epsilon))$ from the property of $\mathcal{A}$. Then, it clearly follows that

$$n + \sqrt{n\kappa}\log\left(\frac{1}{\epsilon}\right) \lesssim b\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right).$$

Therefore, we get a lower-bound $\tilde{\Omega}(\max\{n/\sqrt{\kappa}, \sqrt{n}\})$ on minibatch size b, where $\tilde{\Omega}$ hides a constant and a logarithmic term. We immediately conclude the following statement from this consideration.

Proposition 2. *We consider a first-order stochastic optimization method $\mathcal{M}$ with minibatch of size b. We assume that $\mathcal{M}$ is included in the class $\mathcal{A}$ described above. Then, the minibatch size b is $\tilde{\Omega}(\max\{n/\sqrt{\kappa}, \sqrt{n}\})$.*

We find that these lower-bound are attained by Acc-Prox-SVRG with $b = \tilde{O}(n/\sqrt{\kappa})$ when $n \geq \kappa$ and by Acc-Prox-SVRG with Catalyst and $b = O(\sqrt{n})$ when $n < \kappa$.

In the rest of this section, we explain this minibatch efficiency discussed above leads to effective parallelization. An obvious benefit by minibatching is speeding up by parallel computing of stochastic gradients with minibatch as done in (Dekel et al., 2012; Agarwal and Duchi, 2011; Shalev-Shwartz and Zhang, 2013a). In this view point, we notice that the optimal iteration complexity gives the theoretical limit of the running time by minibatch parallelization if we ignore the communication cost. This limit is attained when we used some stochastic accelerated methods and the number of processors is proportional to the minibatch size providing the optimal iteration complexity. Therefore, to estimate such a minibatch size is an important problem. We note that many of the optimal methods in terms of the total complexity such as Acc-SDCA (Shalev-Shwartz and Zhang, 2014), APCG (Lin et al., 2014), and SPDC (Zhang and Xiao, 2017) require $b = O(n)$ for achieving the optimal iteration complexity. On the other hand, as shown in Proposition 2, our method can obtain the optimal iteration complexity by the optimal minibatch size, which means the effectiveness of the method by minibatch parallelization. In addition, we note that DASVRDA (Murata and Suzuki, 2017) that has the same property of our proposed method.

3.5 Numerical Experiments

In this section, we compare Acc-Prox-SVRG with Prox-SVRG and APG on L_1-regularized multi-class logistic regression with the regularization parameter λ. Table 3.2 provides details of the datasets and regularization parameters utilized in our experiments. These datasets can be found at the LIBSVM website[1]. The best choice of mini-batch size is $b = \lceil b_0 \rceil$, which allows us to take a large learning rate, $\eta = \frac{1}{2L}$. Therefore, we have $m \geq O(\sqrt{\kappa})$ and $\beta_k = \frac{\sqrt{2\kappa}-1}{\sqrt{2\kappa}+1}$. When the number of components n is very large compared with $\sqrt{\kappa}$, we see that $b_0 = O(\sqrt{\kappa})$; for this, we set $m = \delta b$ ($\delta \in \{0.1,\ 1.0,\ 10\}$) and $\beta_k = \frac{b-2}{b+2}$ varying b in the set $\{100,\ 500,\ 1000\}$. We ran Acc-Prox-SVRG using values of η from the range $\{0.01,\ 0.05,\ 0.1,\ 0.5,\ 1.0,\ 5.0,\ 10.0\}$, and we chose the best η in each mini-batch setting.

Figure 3.1 compares Acc-Prox-SVRG with Prox-SVRG and APG. The horizontal axis is the number of single-component gradient evaluations. For Acc-Prox-SVRG, each iteration computes the $2b$ gradients, and at the beginning of each stage, the n component gradients are evaluated. For Prox-SVRG, each iteration computes the two gradients, and at the beginning of each stage, the n gradients are evaluated. For APG, each iteration evaluates n gradients.

As can be seen from Figure 3.1, Acc-Prox-SVRG with good values of b performs better than or is comparable to Prox-SVRG and is much better than results for APG. On the other hand, for relatively large b, Acc-Prox-SVRG may perform worse because of an

[1]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

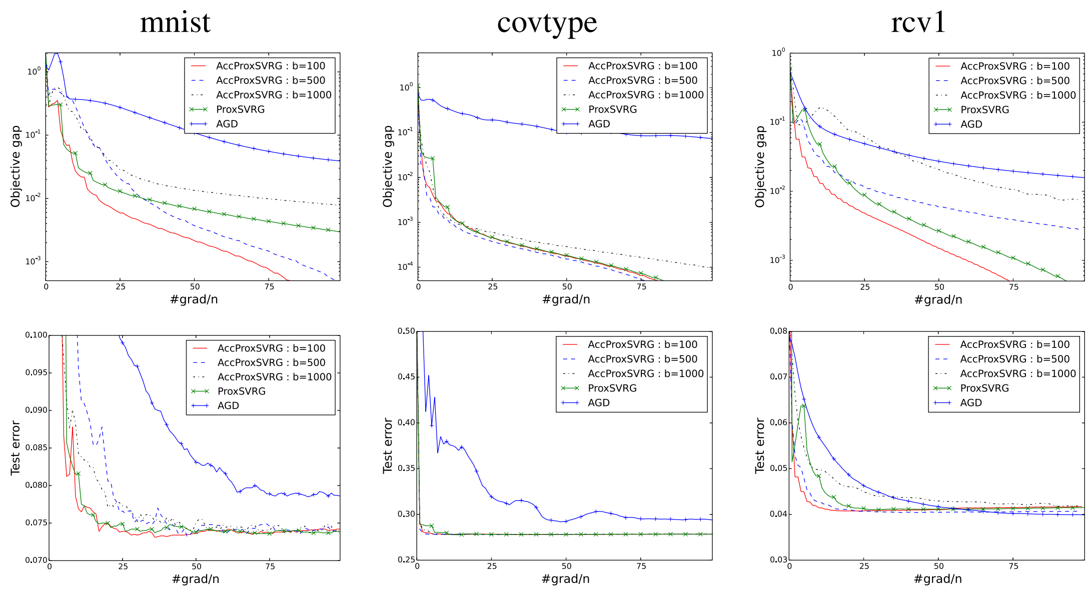

Figure 3.1: Comparison of Acc-Prox-SVRG with Prox-SVRG and APG. Top: Objective gap of L_1 regularized multi-class logistic regression. Bottom: Test error rates.

Table 3.2: Details of data sets and regularization parameters.

Dataset	classes	Training size	Testing size	Features	λ
mnist	10	60,000	10,000	780	10^{-5}
covtype	7	522,910	58,102	54	10^{-6}
rcv1	2	20,242	677,399	47,236	10^{-5}

overestimation of b_0, and hence the worse estimates of m and β_k.

3.6 Appendix

In this section, we provide proofs for a convergence analysis of our method.

3.6.1 Proof of Lemma 2

We first prove auxiliary lemmas for Lemma 2.

Lemma 4. *If $\eta < \frac{1}{\mu}$, then for $k \geq 1$ we have*

$$z_{k+1} = (1-\sqrt{\mu\eta})z_k + \sqrt{\mu\eta}y_k - \sqrt{\frac{\eta}{\mu}}(v_k+\xi_k) \tag{3.11}$$

$$z_k - y_k = \frac{1}{\sqrt{\mu\eta}}(y_k - x_k). \tag{3.12}$$

Proof of Lemma 4. From the definition of estimate sequence and (3.6), we have that for $k \geq 1$

$$\frac{\mu}{2}\|x-z_{k+1}\|^2 + \Phi_{k+1}^* = (1-\sqrt{\mu\eta})\left(\frac{\mu}{2}\|x-z_k\|^2+\Phi_k^*\right) + \sqrt{\mu\eta}\Big(g_{I_k}(y_k) + \langle v_k, x-y_k\rangle_2 \\ +\frac{\mu}{2}\|x-y_k\|^2 + h(x_{k+1}) + \langle \xi_k, x-x_{k+1}\rangle_2\Big).$$

By differentiating at y_k this equality, we obtain

$$\mu(y_k - z_{k+1}) = (1-\sqrt{\mu\eta})\mu(y_k-z_k) + \sqrt{\mu\eta}(v_k+\xi_k).$$

Hence, we have

$$z_{k+1} = (1-\sqrt{\mu\eta})z_k + \sqrt{\mu\eta}y_k - \sqrt{\frac{\eta}{\mu}}(v_k+\xi_k),$$

and that is exactly (3.11) of Lemma 4.

Next, we prove (3.12) of Lemma 4 by induction. It is true for $k=1$. We assume it is true for k, then it follows from (3.11) of Lemma 4 that,

$$\begin{aligned} z_{k+1} - y_{k+1} &= (1-\sqrt{\mu\eta})z_k + \sqrt{\mu\eta}y_k - \sqrt{\frac{\eta}{\mu}}(v_k+\xi_k) - y_{k+1} \\ &= \frac{1}{\sqrt{\mu\eta}}(y_k - \eta(v_k+\xi_k)) - \frac{1-\sqrt{\mu\eta}}{\sqrt{\mu\eta}}x_k - y_{k+1} \\ &= \frac{1}{\sqrt{\mu\eta}}x_{k+1} - \frac{1-\sqrt{\mu\eta}}{\sqrt{\mu\eta}}x_k - y_{k+1}. \end{aligned}$$

From the update rule of y_{k+1}, we have

$$\begin{aligned}
-\frac{1-\sqrt{\mu\eta}}{\sqrt{\mu\eta}}x_k &= \frac{1+\sqrt{\mu\eta}}{\sqrt{\mu\eta}}\left(y_{k+1} - \left(1+\frac{1-\sqrt{\mu\eta}}{1+\sqrt{\mu\eta}}\right)x_{k+1}\right) \\
&= \frac{1+\sqrt{\mu\eta}}{\sqrt{\mu\eta}}y_{k+1} - \frac{2}{\sqrt{\mu\eta}}x_{k+1}.
\end{aligned}$$

Hence, we get

$$z_{k+1} - y_{k+1} = \frac{1}{\sqrt{\mu\eta}}(y_{k+1} - x_{k+1}).$$

□

Lemma 5. *For $k \geq 1$, we have*

$$\langle \nabla g(y_k) + \xi_k, v_k + \xi_k\rangle_2 = \frac{1}{2}\left(\|\nabla g(y_k) + \xi_k\|^2 + \|v_k + \xi_k\|^2 - \|\nabla g(y_k) - v_k\|^2\right), \quad (3.13)$$

$$\|v_k + \xi_k\|^2 \leq 2\left(\|\nabla g(y_k) + \xi_k\|^2 + \|\nabla g(y_k) - v_k\|^2\right), \quad (3.14)$$

$$\|\nabla g(y_k) + \xi_k\|^2 \leq 2\left(\|v_k + \xi_k\|^2 + \|\nabla g(y_k) - v_k\|^2\right). \quad (3.15)$$

Proof of Lemma 5. Averaging

$$\langle \nabla g(y_k) + \xi_k, v_k + \xi_k\rangle_2 = \|\nabla g(y_k) + \xi_k\|^2 + \langle \nabla g(y_k) + \xi_k, v_k - \nabla g(y_k)\rangle_2$$

and

$$\langle \nabla g(y_k) + \xi_k, v_k + \xi_k\rangle_2 = \|v_k + \xi_k\|^2 + \langle v_k + \xi_k, \nabla g(y_k) - v_k\rangle_2,$$

we get (3.13) of Lemma 5:

$$\langle \nabla g(y_k) + \xi_k, v_k + \xi_k\rangle_2 = \frac{1}{2}\left(\|\nabla g(y_k) + \xi_k\|^2 + \|v_k + \xi_k\|^2 - \|\nabla g(y_k) - v_k\|^2\right).$$

The inequality (3.14) of Lemma 5 is shown as follows:

$$\begin{aligned}
\|v_k + \xi_k\|^2 &= \|v_k + \xi_k + \nabla g(y_k) + \xi_k - (\nabla g(y_k) + \xi_k)\|^2 \\
&= \|\nabla g(y_k) + \xi_k\|^2 + 2(\nabla g(y_k) + \xi_k, v_k - \nabla g(y_k)) + \|v_k - \nabla g(y_k)\|^2 \\
&\leq 2(\|\nabla g(y_k) + \xi_k\|^2 + \|v_k - \nabla g(y_k)\|^2).
\end{aligned}$$

In the last inequality, we use

$$|\langle a, b\rangle_2| \leq \frac{\|a\|^2 + \|b\|^2}{2}, \qquad for\ \forall a, \forall b \in \mathbb{R}^d.$$

The inequality (3.15) of Lemma 5 can be proved in a similar way. □

We here prove Lemma 2.

Proof of Lemma 2. We prove inequalities (3.7) and (3.8) by induction. Obviously, the inequality (3.7) holds for $k = 1$. We assume it is also true for k. From the definition of estimate sequence, we have

$$\begin{aligned}
\mathbb{E}\left[\Phi_{k+1}(x)\right] &= (1-\sqrt{\mu\eta})\mathbb{E}\left[\Phi_k(x)\right] + \sqrt{\mu\eta}\,\mathbb{E}\left[g_{I_k}(y_k) + \langle v_k, x - y_k\rangle_2\right. \\
&\quad \left. + \frac{\mu}{2}\|x - y_k\|^2 + h(x_{k+1}) + \langle \xi_k, x - x_{k+1}\rangle_2\right] \\
&\le (1-\sqrt{\mu\eta})f(x) + (1-\sqrt{\mu\eta})^k(\Phi_1 - f)(x) \\
&\quad + \sqrt{\mu\eta}\,\mathbb{E}\left[g(y_k) + \langle \nabla g(y_k), x - y_k\rangle_2 + \frac{\mu}{2}\|x - y_k\|^2 + h(x_{k+1}) + \langle \xi_k, x - x_{k+1}\rangle_2\right] \\
&\le (1-\sqrt{\mu\eta})f(x) + (1-\sqrt{\mu\eta})^k(\Phi_1 - f)(x) + \sqrt{\mu\eta}(g(x) + h(x)) \\
&= f(x) + (1-\sqrt{\mu\eta})^k(\Phi_1 - f)(x),
\end{aligned}$$

where for the first inequality we used induction hypothesis, $\mathbb{E}_{I_k}[g_{I_k}(y_k)] = \mathbb{E}[g(y_k)]$ and $\mathbb{E}_{I_k}[v_k] = \mathbb{E}[\nabla g(y_k)]$, for the last inequality we used the convexity of g and h. Hence, the inequality (3.7) follows.

We next prove (3.8). From the definition of Φ_1, $\Phi_1^* = f(x_1)$. we assume (3.8) is true for k. Using the equation (3.11), we have

$$\begin{aligned}
\|y_k - z_{k+1}\|^2 &= \left\|(1-\sqrt{\mu\eta})(y_k - z_k) + \sqrt{\frac{\eta}{\mu}}(v_k + \xi_k)\right\|^2 \\
&= (1-\sqrt{\mu\eta})^2\|y_k - z_k\|^2 + 2\sqrt{\frac{\eta}{\mu}}(1-\sqrt{\mu\eta})\,\langle y_k - z_k, v_k + \xi_k\rangle_2 + \frac{\eta}{\mu}\|v_k + \xi_k\|^2.
\end{aligned}$$

From the above equation and (3.6) with $x = y_k$, we get

$$\begin{aligned}
\Phi_{k+1}(y_k) &= \Phi_{k+1}^* \\
&\quad + \frac{\mu}{2}\left\{(1-\sqrt{\mu\eta})^2\|y_k - z_k\|^2 + 2\sqrt{\frac{\eta}{\mu}}(1-\sqrt{\mu\eta})\,\langle y_k - z_k, v_k + \xi_k\rangle_2 + \frac{\eta}{\mu}\|v_k + \xi_k\|^2\right\}.
\end{aligned}$$

On the other hand, from the definition of the estimate sequence and (3.6),

$$\Phi_{k+1}(y_k) = (1-\sqrt{\mu\eta})\left(\Phi_k^* + \frac{\mu}{2}\|y_k - z_k\|^2\right) + \sqrt{\mu\eta}(g_{I_k}(y_k) + h(x_{k+1}) + (\xi_k, y_k - x_{k+1})).$$

Therefore, from these two equations, we have

$$\begin{aligned}
\Phi_{k+1}^* &= (1-\sqrt{\mu\eta})\Phi_k^* + \frac{\mu}{2}(1-\sqrt{\mu\eta})\sqrt{\mu\eta}\|y_k - z_k\|^2 + \sqrt{\mu\eta}(g_{I_k}(y_k) + h(x_{k+1}) \\
&\quad + \langle \xi_k, y_k - x_{k+1}\rangle_2) - (1-\sqrt{\mu\eta})\sqrt{\mu\eta}\,\langle y_k - z_k, v_k + \xi_k\rangle_2 - \frac{\eta}{2}\|v_k + \xi_k\|^2. \quad (3.16)
\end{aligned}$$

Since g is Lipschitz smooth, we bound $f(x_{k+1})$ as follows:

$$f(x_{k+1}) \le g(y_k) + \langle \nabla g(y_k), x_{k+1} - y_k\rangle_2 + \frac{L}{2}\|x_{k+1} - y_k\|^2 + h(x_{k+1}). \qquad (3.17)$$

Using (3.16), (3.17), (3.12), and $x_{k+1} - y_k = -\eta(v_k + \xi_k)$ we have

$$\begin{aligned}
&\mathbb{E}_{I_k}\left[f(x_{k+1}) - \Phi^*_{k+1}\right] \\
&\underset{(3.16),(3.17)}{\leq} \mathbb{E}_{I_k}\Big[(1-\sqrt{\mu\eta})(-\Phi^*_k + g(y_k) + h(x_{k+1})) + \langle \nabla g(y_k), x_{k+1} - y_k\rangle_2 \\
&+ \sqrt{\mu\eta}\,\langle \xi_k, x_{k+1} - y_k\rangle_2 + \frac{L}{2}\|x_{k+1} - y_k\|^2 - \frac{\mu}{2}(1-\sqrt{\mu\eta})\sqrt{\mu\eta}\|y_k - z_k\|^2 \\
&+ (1-\sqrt{\mu\eta})\sqrt{\mu\eta}\,\langle y_k - z_k, v_k + \xi_k\rangle_2 + \frac{\eta}{2}\|v_k + \xi_k\|^2\Big] \\
&\underset{(3.12)}{=} \mathbb{E}_{I_k}\Big[(1-\sqrt{\mu\eta})(-\Phi^*_k + g(y_k) + h(x_{k+1}) + \langle x_k - y_k, v_k + \xi_k\rangle_2) - \eta(\nabla g(y_k), v_k + \xi_k) \\
&- \eta\sqrt{\mu\eta}\,\langle \xi_k, v_k + \xi_k\rangle_2 - \frac{\mu}{2}\frac{1-\sqrt{\mu\eta}}{\sqrt{\mu\eta}}\|y_k - x_k\|^2 + \frac{\eta}{2}(L\eta + 1)\|v_k + \xi_k\|^2\Big], \qquad (3.18)
\end{aligned}$$

where for the first inequality we used $\mathbb{E}_{I_k}[g_{I_k}(y_k)] = g(y_k)$.

Here, we give the following,

$$\begin{aligned}
&\mathbb{E}_{I_k}\left[g(y_k) + h(x_{k+1}) + \langle x_k - y_k, v_k + \xi_k\rangle_2\right] \\
&= \mathbb{E}_{I_k}\left[g(y_k) + \langle v_k, x_k - y_k\rangle_2 + h(x_{k+1}) + \langle \xi_k, x_k - x_{k+1}\rangle_2 + \langle \xi_k, x_{k+1} - y_k\rangle_2\right] \\
&\leq \mathbb{E}_{I_k}\left[g(x_k) - \frac{\mu}{2}\|x_k - y_k\|^2 + h(x_k) - \eta\,\langle \xi_k, v_k + \xi_k\rangle_2\right], \qquad (3.19)
\end{aligned}$$

where for the first inequality we used $\mathbb{E}_{I_k}[v_k] = \nabla g(y_k)$ and convexity of g and h. Thus we have

$$\begin{aligned}
&\mathbb{E}_{I_k}\left[f(x_{k+1}) - \Phi^*_{k+1}\right] \\
&\underset{(3.18),(3.19)}{\leq} \mathbb{E}_{I_k}\Big[(1-\sqrt{\mu\eta})(f(x_k) - \Phi^*_k) - \frac{\mu}{2}\frac{1-\mu\eta}{\sqrt{\mu\eta}}\|x_k - y_k\|^2 \\
&- \eta\,\langle \nabla g(y_k) + \xi_k, v_k + \xi_k\rangle_2 + \frac{\eta}{2}(1 + L\eta)\|v_k + \xi_k\|^2\Big] \\
&\underset{(3.13)}{\leq} \mathbb{E}_{I_k}\Bigg[(1-\sqrt{\mu\eta})(f(x_k) - \Phi^*_k) - \frac{\mu}{2}\frac{1-\mu\eta}{\sqrt{\mu\eta}}\|x_k - y_k\|^2 \\
&- \frac{\eta}{2}\|\nabla g(y_k) + \xi_k\|^2 + \frac{L\eta^2}{2}\|v_k + \xi_k\|^2 + \frac{\eta}{2}\|v_k - \nabla g(y_k)\|^2\Bigg] \\
&\underset{(3.14),\eta\leq\frac{1}{2L}}{\leq} \mathbb{E}_{I_k}\left[(1-\sqrt{\mu\eta})(f(x_k) - \Phi^*_k) - \frac{\mu}{2}\frac{1-\mu\eta}{\sqrt{\mu\eta}}\|x_k - y_k\|^2 + \eta\|v_k - \nabla g(y_k)\|^2\right].
\end{aligned}$$

By taking expectation with respect to the history of random variables $I_1, \ldots, I_{k-1}$, the induction hypothesis finishes the proof of (3.8). □

3.6.2 Proof of Lemma 3

Lemma 3 is the key lemma which give a bound on the variance. Now we give this proof.

Proof of Lemma 3. We set $v_j^1 = \nabla g_j(y_k) - \nabla g_j(\tilde{x}) + \tilde{v}$. Since

$$v_k = \frac{1}{b}\sum_{j\in I_k} v_j^1,$$

conditional variance of v_k is as follows (see Freund (1971); Nitanda (2016))

$$\mathbb{E}_{I_k}\|v_k - \nabla g(y_k)\|^2 = \frac{1}{b}\frac{n-b}{n-1}\mathbb{E}_j\|v_j^1 - \nabla g(y_k)\|^2,$$

where expectation in right hand side is taken with respect to $j \in \{1,\dots,n\}$. Therefore, it suffices to prove that

$$\mathbb{E}_j\|v_j^1 - \nabla g(y_k)\|^2 \le 2L^2\|y_k - x_k\|^2 + 8L(f(x_k) - f(x_*) + f(\tilde{x}) - f(x_*)). \tag{3.20}$$

For $i \in \{1,\dots,n\}$, we set

$$\phi_i(x) = g_i(x) - (g_i(x_*) + \langle\nabla g_i(x_*), x - x_*\rangle_2).$$

We have that $\phi_i(x_*) = \min_x \phi_i(x)$ since $\nabla\phi_i(x_*) = 0$ and convexity of ϕ_i. Since $\nabla\phi_i$ is Lipschitz continuous with L, it follows that (see (Nesterov, 2004, Theorem 2.1.5))

$$\frac{1}{2L}\|\nabla\phi_i(x)\|^2 \le \phi_i(x) - \phi_i(x_*) = \phi_i(x),$$

Thus,

$$\|\nabla g_i(x) - \nabla g_i(x_*)\|^2 \le 2L(g_i(x) - g_i(x_*) - \langle\nabla g_i(x_*), x - x_*\rangle_2).$$

Averaging from $i = 1$ to n, we have

$$\frac{1}{n}\sum_{i=1}^n \|\nabla g_i(x) - \nabla g_i(x_*)\|^2 \le 2L(g(x) - g(x_*) - \langle\nabla g(x_*), x - x_*\rangle_2).$$

By the optimality of x_*, $-\nabla g(x_*)$ is a subgradient of h at x_*, so that

$$\langle -\nabla g(x_*), x - x_*\rangle_2 \le h(x) - h(x_*).$$

Hence we get

$$\frac{1}{n}\sum_{i=1}^n \|\nabla g_i(x) - \nabla g_i(x_*)\|^2 \le 2L(g(x) - g(x_*) + h(x) - h(x_*)) = 2L(f(x) - f(x_*)). \tag{3.21}$$

We now bound left hand side of (3.20) as follows:

$$\begin{aligned}
&\mathbb{E}_j\|v_j^1 - \nabla g(y_k)\|^2 \\
&\quad = \mathbb{E}_j\|\nabla g_j(y_k) - \nabla g_j(\tilde{x}) - (\nabla g(y_k) - \nabla g(\tilde{x}))\|^2 \\
&\quad \le \mathbb{E}_j\|\nabla g_j(y_k) - \nabla g_j(\tilde{x})\|^2 \\
&\quad \le 2\mathbb{E}_j\|\nabla g_j(y_k) - \nabla g_j(x_k)\|^2 + 4\mathbb{E}_j\|\nabla g_j(x_k) - \nabla g_j(x_*)\|^2 + 4\mathbb{E}_j\|\nabla g_j(x_*) - \nabla g_j(\tilde{x})\|^2 \\
&\quad \le 2L^2\|y_k - x_k\|^2 + 8L(f(x_k) - f(x_*) + f(\tilde{x}) - f(x_*)),
\end{aligned}$$

where for the first inequality we used $\mathbb{E}\|\zeta - \mathbb{E}\zeta\|^2 \le \mathbb{E}\|\zeta\|^2$ for any random vector ζ, for the second inequality, we used $\|a+b\|^2 \le 2\|a\|^2 + 2\|b\|^2$, and for the last inequality, we used L-Lipschitz continuity and (3.21). This finishes the proof of lemma. □

3.6.3 Proof of Theorem 4

We give the proof of the main convergence theorem.

Proof of Theorem 4. From (3.8), (3), and (3.7) with $x = x_*$, it follows that

$$\begin{aligned}
\mathbb{E}\left[f(x_k) - f(x_*)\right] \le (1-\sqrt{\mu\eta})^{k-1}(\Phi_1 - f)(x_*) + \mathbb{E}\Big[\textstyle\sum_{l=1}^{k-1}(1-\sqrt{\mu\eta})^{k-1-l} \\
\cdot \Big\{\Big(-\tfrac{\mu}{2}\tfrac{1-\mu\eta}{\sqrt{\mu\eta}} + \tfrac{n-b}{n-1}\tfrac{2L^2\eta}{b}\Big)\|x_l - y_l\|^2 + \tfrac{n-b}{n-1}\tfrac{8L\eta}{b}(f(x_l) - f(x_*) + f(\tilde{x}) - f(x_*))\Big\}\Big].
\end{aligned}$$

If $\eta \le \min\left\{\frac{(pb)^2}{64}\left(\frac{n-1}{n-b}\right)^2\frac{\mu}{L^2}, \frac{1}{2L}\right\}$, then the coefficients of $\|x_l - y_l\|^2$ are non-positive for $p \le 2$. Indeed, using

$$\eta \le \frac{(pb)^2}{64}\left(\frac{n-1}{n-b}\right)^2\frac{\mu}{L^2} \Rightarrow \frac{n-b}{n-1}\frac{L\eta}{b} \le \frac{p}{8}\sqrt{\mu\eta}, \quad \textit{for}\ \ p > 0, \tag{3.22}$$

we get

$$\begin{aligned}
-\tfrac{\mu}{2}\tfrac{1-\mu\eta}{\sqrt{\mu\eta}} + \tfrac{n-b}{n-1}\tfrac{2L^2\eta}{b} &\le -\tfrac{\mu}{2}\tfrac{1-\mu\eta}{\sqrt{\mu\eta}} + \tfrac{L}{2}\sqrt{\mu\eta} \\
= \tfrac{1}{2\sqrt{\mu\eta}}(-\mu + \mu^2\eta + \mu L\eta) \underset{\mu\le L}{\le} \tfrac{1}{2\sqrt{\mu\eta}}&(-\mu + 2\mu L\eta) \underset{\eta\le\frac{1}{2L}}{\le} 0.
\end{aligned}$$

Thus, using (3.22) again with $p \le 1$, we have

$$\begin{aligned}
\mathbb{E}\left[f(x_k) - f(x_*)\right] &\le (1-\sqrt{\mu\eta})^{k-1}(\Phi_1 - f)(x_*) \\
&\quad + \mathbb{E}\left[\sum_{l=1}^{k-1}(1-\sqrt{\mu\eta})^{k-1-l}p\sqrt{\mu\eta}(f(x_l) - f(x_*) + f(\tilde{x}) - f(x_*))\right] \\
&\le (1-\sqrt{\mu\eta})^{k-1}(\Phi_1 - f)(x_*) + p(f(\tilde{x}) - f(x_*))
\end{aligned}$$

$$+\mathbb{E}\left[\sum_{l=1}^{k-1}(1-\sqrt{\mu\eta})^{k-1-l}p\sqrt{\mu\eta}(f(x_l)-f(x_*))\right], \tag{3.23}$$

where for the last inequality we used $\sum_{l=1}^{k-1}(1-\sqrt{\mu\eta})^{k-1-l} \leq \sum_{t=0}^{\infty}(1-\sqrt{\mu\eta})^t = \frac{1}{\sqrt{\mu\eta}}$.

We denote $E[f(x_k)-f(x_*)]$ by V_k, and we use W_k to denote the last expression in (3.23). Thus, for $k \geq 1$, $V_k \leq W_k$. For $k \geq 2$, we have

$$\begin{aligned} W_k &= (1-\sqrt{\mu\eta})\left\{(1-\sqrt{\mu\eta})^{k-2}(\Phi_1 - f)(x_*) + pV_1 + \sum_{l=1}^{k-2}(1-\sqrt{\mu\eta})^{k-2-l}p\sqrt{\mu\eta}\, V_l\right\} \\ &\quad + p\sqrt{\mu\eta}\, V_{k-1} + p\sqrt{\mu\eta}\, V_1 \leq (1-\sqrt{\mu\eta}(1-p))W_{k-1} + p\sqrt{\mu\eta}\, W_1. \end{aligned}$$

Since $0 < \sqrt{\mu\eta}(1-p) < 1$, the above inequality leads to

$$W_k = \left((1-(1-p)\sqrt{\mu\eta})^{k-1} + \frac{p}{1-p}\right)W_1. \tag{3.24}$$

From the strong convexity of g (and f), we can see

$$W_1 = (1+p)(f(\tilde{x})-f(x_*)) + \frac{\mu}{2}\|\tilde{x}-x_*\|^2 \leq (2+p)(f(\tilde{x})-f(x_*)).$$

Thus, for $k \geq 2$, we have

$$V_k \leq W_k \leq \left((1-(1-p)\sqrt{\mu\eta})^{k-1} + \frac{p}{1-p}\right)(2+p)(f(\tilde{x})-f(x_*)),$$

and that is exactly (3.9). Using $\log(1-\alpha) \leq -\alpha$ and $m \geq \frac{1}{(1-p)\sqrt{\mu\eta}}\log\frac{1-p}{p}$, we have

$$\log(1-(1-p)\sqrt{\mu\eta})^m \leq -m(1-p)\sqrt{\mu\eta} \leq -\log\frac{1-p}{p},$$

so that

$$(1-(1-p)\sqrt{\mu\eta})^m \leq \frac{p}{1-p}.$$

This finishes the proof of Theorem 4. □

Chapter 4

Accelerated Variance Reduced Stochastic Gradient Descent II

In the previous chapter, we proposed a method incorporating an accelerated gradient descent technique by Nesterov, a stochastic variance reduction gradient, and minibatching. In this chapter, we proposed a new method by adopting another accelerated method for smooth convex finite-sum problems. An important feature of this method is that unlike Acc-Prox-SVRG, it can be directly applied to general convex and optimal strongly convex problems that is a weaker condition than strong convexity. Thus, we extend results shown in the previous chapter to these problems. In experiments, we show the effectiveness of the method.

This chapter is based on the work *Accelerated Stochastic Gradient Descent for Minimizing Finite Sums*, A. Nitanda, Artificial Intelligence and Statistics, 2016 (Nitanda, 2016).

4.1 Overview

In this chapter, we consider the following empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \overset{\mathrm{def}}{=} \frac{1}{n} \sum_{i=1}^{n} g_i(x) \right\}, \tag{4.1}$$

where $g_1, \ldots, g_n$ are smooth convex functions from $\mathbb{R}^d$ to $\mathbb{R}$. In machine learning, we often encounter optimization problems as mentioned earlier. Note that each $g_i(x)$ may include smooth regularization terms.

As mentioned in the previous chapter, several studies recently proposed effective methods (SAG (Roux et al., 2012; Schmidt et al., 2017), SDCA (Shalev-Shwartz and Zhang, 2012, 2013b), SVRG (Johnson and Zhang, 2013), S2GD (Konečný and Richtárik, 2013), Acc-Prox-SDCA (Shalev-Shwartz and Zhang, 2014), Prox-SVRG (Xiao and Zhang, 2014), MISO (Mairal, 2015), SAGA (Defazio et al., 2014), APCG (Lin et al., 2014), Acc-Prox-SVRG (Nitanda, 2014), mS2GD (Konečnỳ et al., 2016), SPDC (Zhang and Xiao, 2017)) for solving strongly convex finite-sum problems. These methods attempt to reduce the variance of the stochastic gradient and achieve the linear convergence rates like a deterministic gradient descent. Moreover, because of the computational efficiency of each iteration, the overall complexities of these methods are less than those of the deterministic and stochastic gradient descent methods.

However, many problems arise in machine learning may become not strongly convex. An advantage of the SAG and SAGA is that they support general convex problems. Although we can apply any of these methods to non-strongly convex functions by adding a slight L_2-regularization, this modification increases the difficulty of model selection. In the general convex case, the overall complexities of SAG and SAGA are $O((n+L)/\epsilon)$. This complexity is less than that of the deterministic gradient descent, which have a complexity of $O(nL/\epsilon)$, and is a trade-off with $O(n\sqrt{L/\epsilon})$, which is the complexity of the AGD.

More recently, Gong and Ye (2014) showed that Prox-SVRG has linear rate of convergence for *optimal strongly convex* problems that is quite weaker condition than the strong convexity. Our proposed method called AMSVRG in this chapter is similar to that in the previous chapter, that is, it incorporates an accelerated gradient descent and stochastic variance reduced gradient in a mini-batch setting. The difference between this method and Acc-Prox-SVRG is the type of accelerated methods. Namely, AMSVRG incorporates Allen-Zhu and Orecchia (2014), which is similar to Nesterov's acceleration (Nesterov, 2005), whereas Acc-Prox-SVRG incorporates Nesterov (2004). An important feature of AMSVRG is that it can be directly applied to general convex and optimal strongly convex problems. We show that for general convex problems, AMSVRG achieves an total complexity of

$$\tilde{O}\left(n+\min\left\{\frac{L}{\epsilon}, n\sqrt{\frac{L}{\epsilon}}\right\}\right),$$

where the notation $\tilde{O}$ hides constant and logarithmic terms. This complexity is less than that of SAG, SAGA, and AGD. In the optimal strongly convex case, our method achieves an total complexity

$$\tilde{O}\left(n+\min\left\{\kappa,\ n\sqrt{\kappa}\right\}\right),$$

where κ is the condition number L/μ. Moreover, an iteration complexity (the number of iterations needed to find an ϵ-accurate solution in expectation) is

$$O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right),$$

This iteration complexity is the same as that of deterministic acceleration methods, i.e., best iteration complexity. Thus, AMSVRG method converges quickly for general convex and optimal strongly convex problems by parallelization of minibatch computation. We also note that AMSVRG has the same important feature of Acc-Prox-SVRG described in the previous chapter, namely, it can achieve the both optimal iteration and total complexities simultaneously by applying APPA or Catalyst if needed.

4.2 Preliminary

In this section, we introduce some notions, assumptions, and methods used in this chapter. To provide a convergence analysis, we make the Lipschitz smoothness assumption.

Assumption 3. *Let L be a positive value. We assume that each function $g_i(x)$ is convex and L-Lipschitz smooth, that is, it follows that,*

$$g_i(x) \le g_i(y) + \langle \nabla g_i(y), x - y \rangle_2 + \frac{L}{2}\|x - y\|^2.$$

While we assumed the strong convexity of loss functions in the previous chapter, we suppose a much weaker condition called *optimal strongly convex* in this chapter. We next introduce its definition.

4.2.1 Optimal Strongly Convex

We make an optimal strongly convex assumption as follows.

Assumption 4. *Let C be a subset of $\mathbb{R}^d$ and X_* denote the optimal set. We assume $X_* \neq \phi$. f(x) is μ-optimal-strongly convex on C, i.e., there exists $\mu > 0$ such that for all $x \in C \setminus X_*$,*

$$f_* + \frac{\mu}{2}\|x - \Pi_{X_*}(x)\|^2 \le f(x),$$

where f_* is the optimal value and Π_{X_*} denotes the projection onto X_*.

Obviously, we can see that optimal strong convexity is a weaker condition than strong convexity. Since $f_* + \frac{L}{2}\|x - \Pi_{X_*}(x)\|^2 \ge f(x)$ by L-smoothness, we have $\mu \le L$. We denote the ratio between L and μ by κ and we call it the *condition number*.

The main differences between strong convexity and optimal strong convexity are that the latter condition admits an infinite number of solutions and linear parts of the function. Thus, optimal strongly convex is a very large class.

Two quantities $\frac{1}{2}\|x - \Pi_{X_*}(x)\|^2$ and $f(x) - f_*$ are optimality measures and continuous functions, so that the ratio between these two values: $\mu(x) = \frac{2(f(x)-f_*)}{\|x-\Pi_{X_*}(x)\|^2}$ is positive continuous on the complement of X_*. Let $C \subset \mathbb{R}^d$ be a compact subset. Then, $\mu \overset{\text{def}}{=} \inf_{x \in C \setminus X_*} \mu(x)$ gives the optimal strong convexity parameter on C. Since $C \setminus U$ is also compact, where U is an arbitrary small open neighborhood of X_*, $\mu(x)$ has positive minimum values on $C \setminus U$. This means that whether Assumption 4 is satisfied or not depend on the behavior of f around the boundary of $C \cap X_*$. Therefore, many problems belong to the class of optimal strongly convex on compact set.

A smoothed hinge loss function (see Figure 4.1)

$$f(x) = \begin{cases} \frac{1}{2} - x & (x \leq 0), \\ \frac{1}{2}(1-x)^2 & (0 < x \leq 1), \\ 0 & (1 < x), \end{cases}$$

is a simple example of optimal strongly function on a bounded region. Let $C = [-a, a]$ $(a > 1)$ be a bounded range. Since $X_* = [1, \infty)$ and $\Pi_{X_*}(x) = 1$ for $x \notin X_*$, we can easily see that optimal strong convexity parameter on C for smoothed hinge loss is

$$\mu = \inf_{x \in [-a,1)} \mu(x) = \frac{2f(-a)}{|1+a|^2} = \frac{1+2a}{|1+a|^2} > 0.$$

Here, we checked the value of μ, but we can conclude the positivity of μ by the fact that $C = [-a, a]$ is compact and f is quadratic around $\partial(C \cap X_*) = \{1\}$.

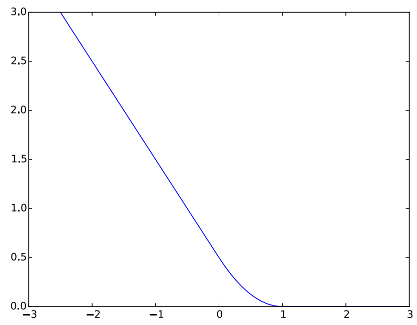
Figure 4.1: Smoothed hinge loss.

In our analyses, for optimal strongly convex problems we assume that points generated by algorithm is contained in C. For monotonic algorithms (generating decreasing sequence $f(w_s)_{s=1,2,\ldots}$), we may consider the case where C is the sublevel set $\{x \in \mathbb{R}^d; f(x) \leq c\}$ and this assumption holds for sufficiently large $c \geq f_*$. Here, we give the condition of compactness of sublevel set.

Proposition 3. *Let f be C^1 class convex function and X_* be the optimal set of f. If X_* is compact, then for $c \geq f_*$, the sublevel set $\{x \in \mathbb{R}^d; f(x) \leq c\}$ is also compact.*

Thus, by the above discussion, the monotonic algorithm deals with many problems as optimal strongly convex problems and potentially converge fast. We propose such a method later.

4.2.2 Accelerated Gradient Descent

In this section, we review the recently proposed accelerated gradient method. We first introduce some notations. In this section, $\|\cdot\|$ denotes the general norm on $\mathbb{R}^d$. Let

$d(x) : \mathbb{R}^d \to \mathbb{R}$ be a distance generating function (i.e., 1-strongly convex smooth function with respect to $\|\cdot\|$). Accordingly, we define the Bregman divergence by

$$V_x(y) = d(y) - (d(x) + \langle \nabla d(x), y - x \rangle_2), \quad \forall x, \forall y \in \mathbb{R}^d,$$

where $(,)$ is the Euclidean inner product. The accelerated method proposed in Allen-Zhu and Orecchia (2014) called *Linear Coupling* uses a gradient step and mirror descent steps and takes a linear combination of these points. The procedure of this method is described in Algorithm 5.

Algorithm 5 Linear Coupling

```
Input: the number of iterations $m$, learning rates $\eta$, $(\alpha_{k+1})_{k=0}^m$, coefficients $(\tau_k)_{k=0}^m$,
and initial points $y_0, z_0$
for $k = 0$ to $m$ do
    $x_{k+1} \leftarrow (1 - \tau_k) y_k + \tau_k z_k$
    $v_{k+1} \leftarrow \nabla f(x_{k+1})$
    $y_{k+1} \leftarrow \arg\min_{y \in \mathbb{R}^d} \{ \eta \langle v_{k+1}, y - x_{k+1} \rangle_2 + \frac{1}{2}\|y - x_{k+1}\|^2 \}$   $(GD\ step)$
    $z_{k+1} \leftarrow \arg\min_{z \in \mathbb{R}^d} \{ \alpha_{k+1} \langle v_{k+1}, z - z_k \rangle_2 + V_{z_k}(z) \}$   $(MD\ step)$
end for
Return $y_{m+1}$
```

Then, with appropriate parameters, $f(y_k)$ converge to the optimal value as fast as the Nesterov's accelerated methods (Nesterov, 2004, 2005) for non-strongly convex problems. Moreover, in the strongly convex case, we obtain the same fast convergence as Nesterov's methods by restarting this entire procedure.

In the rest of this chapter, we only consider the Euclidean norm, i.e., $\|\cdot\| = \|\cdot\|_2$.

4.2.3 Stochastic Variance Reduction Gradient

The proposed method in this chapter, we combine stochastic variance reduced gradient with the accelerated method as done in the previous chapter to reduce the variance effect of stochastic gradients. We here list the step of SVRG with minibatch of size b again. SVRG is a multi-stage scheme. During each stage, this method performs m SGD iterations using the following direction,

$$v_k = \nabla g_{I_k}(x_k) - \nabla g_{I_k}(\tilde{x}) + \nabla f(\tilde{x}),$$

where $\tilde{x}$ is a starting point at stage, k is an iteration index, $I_k = \{i_1, \ldots, i_b\}$ is a uniformly randomly chosen size b subset of $\{1, 2, \ldots, n\}$, and $g_{I_k} = \frac{1}{b}\sum_{j=1}^{b} g_{i_j}$. Note that v_k is an unbiased estimator of gradient $\nabla f(x_k)$: $\mathbb{E}_{I_k}[v_k] = \nabla f(x_k)$, where $\mathbb{E}_{I_k}$ denotes the expectation with respect to I_k. A bound on the variance of v_k is given in the following lemma, which is proved in the Appendix.

Lemma 6. *Suppose Assumption 3 holds, and let* $x_* = \underset{x\in\mathbb{R}^d}{\arg\min} f(x)$. *Conditioned on* x_k, *we have*

$$\mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2 \leq 4L\frac{n-b}{b(n-1)}\left(f(x_k) - f_* + f(\tilde{x}) - f_*\right). \tag{4.2}$$

Due to this lemma, SVRG with $b = 1$ achieves a complexity of $O((n+\kappa)\log\frac{1}{\epsilon})$.

4.3 Single-Stage AMSVRG

We now introduce a new method *Accelerated efficient Mini-batch SVRG (AMSVRG)* which incorporates AGD and SVRG in a mini-batch setting. Our method is a multi-stage scheme similar to SVRG. During each stage, this method performs several APG-like (Allen-Zhu and Orecchia, 2014) iterations combining stochastic gradient descent (SGD) and stochastic mirror descent (SMD) steps with SVRG direction in a mini-batch setting. Each stage of AMSVRG is described in Algorithm 6.

Algorithm 6 Single-Stage of AMSVRG

```
Input: the number of iterations $m$, learning rates $\eta$, $(\alpha_{k+1})_{k=0}^m$, mini-batch sizes $(b_{k+1})_{k=0}^m$, coefficients $(\tau_k)_{k=0}^m$, and initial points $y_0, z_0$
$\tilde{v} \leftarrow \frac{1}{n}\sum_{i=1}^n \nabla g_i(y_0)$
for $k = 0$ to $m$ do
    $x_{k+1} \leftarrow (1-\tau_k)y_k + \tau_k z_k$
    Randomly pick subset $I_{k+1} \subset \{1, 2, \ldots, n\}$ of size $b_{k+1}$
    $v_{k+1} \leftarrow \nabla g_{I_{k+1}}(x_{k+1}) - \nabla g_{I_{k+1}}(y_0) + \tilde{v}$
    $y_{k+1} \leftarrow \arg\min_{y\in\mathbb{R}^d}\left\{ \eta\langle v_{k+1}, y - x_{k+1}\rangle_2 + \frac{1}{2}\|y - x_{k+1}\|^2 \right\}$   $(SGD\ step)$
    $z_{k+1} \leftarrow \arg\min_{z\in\mathbb{R}^d}\left\{ \alpha_{k+1}\langle v_{k+1}, z - z_k\rangle_2 + V_{z_k}(z) \right\}$   $(SMD\ step)$
end for
Option I-a: $w \leftarrow y_{m+1}$,
Option I-b: $w \leftarrow \frac{1}{m+1}\sum_{k=1}^{m+1} x_k$,
Option II: If $f(y_1) < f(w)$, then $w \leftarrow y_1$,
Return $w$
```

4.4 Convergence Analysis of the Single-Stage AMSVRG

Before we introduce the multi-stage scheme, we show the convergence of single-stage version Algorithm 6. The following lemma is the key to the analysis of our method and gives us an insight on how to construct algorithms.

Lemma 7. *Consider Algorithm 6 under Assumption 3. We set* $\delta_k = \frac{n-b_k}{b_k(n-1)}$. *Let* $x_* \in \arg\min_{x\in\mathbb{R}^d} f(x)$. *If* $\eta = \frac{1}{L}$, *then we have,*

$$\begin{aligned}\sum_{k=0}^{m}&\alpha_{k+1}\left(\frac{1}{\tau_k} - (1+4\delta_{k+1})L\alpha_{k+1}\right)\mathbb{E}[f(x_{k+1})-f_*)] + L\alpha_{m+1}^2\mathbb{E}[f(y_{m+1})-f_*] \\ &\leq V_{z_0}(x_*) + \sum_{k=1}^{m}\left(\alpha_{k+1}\frac{1-\tau_k}{\tau_k} - L\alpha_k^2\right)\mathbb{E}[f(y_k)-f_*] \\ &+ \left(\alpha_1\frac{1-\tau_0}{\tau_0} + 4L\sum_{k=0}^{m}\alpha_{k+1}^2\delta_{k+1}\right)(f(y_0)-f_*).\end{aligned}$$

The following two lemmas are well known and useful for showing the convergence of stochastic gradient descent and stochastic mirror descent, respectively, and they are also useful for proving Lemma 7.

Lemma 8. *(Stochastic Gradient Descent). Suppose Assumption 3 holds, and let* $\eta = \frac{1}{L}$. *Conditioned on* x_k, *it follows that for* $k \geq 1$,

$$\mathbb{E}_{I_k}[f(y_k)] \leq f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2 + \frac{1}{2L}\mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2. \tag{4.3}$$

Lemma 9. *(Stochastic Mirror Descent). Conditioned on* x_k, *we have that for arbitrary* $u \in \mathbb{R}^d$,

$$\alpha_k\left\langle \nabla f(x_k), z_{k-1}-u\right\rangle_2 \leq V_{z_{k-1}}(u) - \mathbb{E}_{I_k}[V_{z_k}(u)] + \frac{\alpha_k^2}{2}\left(\|\nabla f(x_k)\|^2 + \mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2\right). \tag{4.4}$$

From now on we consider Algorithm 6 with Option I-a and we set η, α_{k+1}, and τ_k as follows: For $k = 0, 1, \ldots$ we set

$$\eta = \frac{1}{L}, \quad \alpha_{k+1} = \frac{1}{4L}(k+2), \quad \frac{1}{\tau_k} = L\alpha_{k+1} + \frac{1}{2}. \tag{4.5}$$

Theorem 5. *Consider Algorithm 6 with Option I-a under Assumption 3. For* $p \in \left(0, \frac{1}{2}\right]$, *we choose* $b_{k+1} \in \mathbb{Z}_+$ *such that* $4L\delta_{k+1}\alpha_{k+1} \leq p$. *Then, we have*

$$\mathbb{E}[f(w)-f_*] \leq \mathbb{E}[f(y_{m+1})-f_*] \leq \frac{16L}{(m+2)^2}V_{z_0}(x_*) + \frac{5}{2}p(f(y_0)-f_*).$$

Moreover, if $m \geq 4\sqrt{\frac{LV_{z_0}(x_*)}{q(f(y_0)-f_*)}}$ *for* $q > 0$, *then it follows*

$$\mathbb{E}[f(w)-f_*] \leq \mathbb{E}[f(y_{m+1})-f_*] \leq \left(q + \frac{5}{2}p\right)(f(y_0)-f_*).$$

Let $b_{k+1}, m \in \mathbb{Z}_+$ be the minimum values satisfying the assumption of Theorem 5 for $p = q = \epsilon$, i.e., $b_{k+1} = \left\lceil \frac{n(k+2)}{\epsilon(n-1)+k+2} \right\rceil$ and $m = \left\lceil 4\sqrt{\frac{LV_{z_0}(x_*)}{\epsilon(f(y_0)-f_*)}} \right\rceil$. Then, from Theorem 5, we have an upper bound on the total complexity (total number of processed examples to obtain ϵ-accurate solution in expectation):

$$O\left(n + \sum_{k=0}^{m} b_{k+1}\right) \le O\left(n + m\frac{nm}{\epsilon n + m}\right) = O\left(n + \frac{nL}{\epsilon^2 n + \sqrt{\epsilon L}}\right),$$

where we used the monotonicity of b_{k+1} with respect to k for the first inequality. Note that the notation O also hides $V_{z_0}(x_*)$ and $f(y_0) - f_*$.

4.5 Multi-Stage AMSVRG

In this subsection, we introduce and analyze AMSVRG, as described in Algorithm 7.

Algorithm 7 Multi-Stage AMSVRG

```
Input: the number of outer-iterations $T$, the number of inner-iterations $(m_s)_{s=0}^{T}$, learning rates $\eta$, $(\alpha_{k+1})_{k\in\mathbb{Z}_+}$, mini-batch sizes $(b_{k+1})_{k\in\mathbb{Z}_+}$, coefficients $(\tau_k)_{k\in\mathbb{Z}_+}$, and initial point $w_0$
for $s = 0$ to $T$ do
  $y_0 \leftarrow w_s, \ z_0 \leftarrow w_s$
  $w_{s+1} \leftarrow \textbf{Algorithm6}(m_s, \ \eta, \ (\alpha_{k+1})_{k\in\mathbb{Z}_+}, \ (b_{k+1})_{k\in\mathbb{Z}_+}, \ (\tau_k)_{k\in\mathbb{Z}_+}, y_0, \ z_0)$
end for
Return $w_{T+1}$
```

If we run Algorithm 6 with Option II in AMSVRG, it follows that $f(w) \le f(y_1)$. Since $x_1 = y_0 = z_0$, the step to obtain y_1 corresponds to the deterministic gradient descent from the starting point at each stage. This means that AMSVRG (with Option II) is monotonic that generates decreasing sequence $\{f(w_s)\}_{s=0,1,\dots}$. Note that Option II requires computational cost for computing function values of $O(n)$ but the order of total complexity does not increase.

4.6 Convergence Analysis of Multi-Stage AMSVRG

General Convex

We consider the convergence of AMSVRG for general convex problems under the following boundedness assumption which has been used in a several studies to analyze incremental and stochastic methods (e.g., Bottou and Le Cun (2005); Gürbüzbalaban et al. (2015)).

Assumption 5. *(Boundedness) There is a compact subset $\Omega \subset \mathbb{R}^d$ such that the sequence $\{w_s\}$ generated by AMSVRG is contained in Ω.*

Note that, if we change the initialization of z_0 from $z_0 \leftarrow w_s$ to $z_0 \leftarrow z$, where z is a constant, the above method with this modification will achieve the same convergence for general convex problems without the boundedness assumption (see Appendix). However, this modified version is slower than the above scheme for the strongly convex case, thus, we here consider this version described in Algorithm 7.

From Theorem 5, we can see that for small p and q (e.g. $p = 1/10,\ q = 1/4$), the expected value of the objective function is halved at every stage under the assumptions of Theorem 5. Hence, running AMSVRG for $O(\log(1/\epsilon))$ outer iterations achieves an ϵ-accurate solution in expectation. Here, we consider the complexity at stage s to halve the expected objective value. Let $b_{k+1}, m_s \in \mathbb{Z}_+$ be the minimum values satisfying the assumption of Theorem 5, i.e., $b_{k+1} = \left\lceil \frac{n(k+2)}{p(n-1)+k+2} \right\rceil$ and $m_s = \left\lceil 4\sqrt{\frac{LV_{w_s}(x_*)}{q(f(w_s)-f_*)}} \right\rceil$. If the initial objective gap $f(w_s) - f_*$ in stage s is larger than ϵ, then the complexity at stage is

$$\begin{aligned} O\left(n + \sum_{k=0}^{m_s} b_{k+1}\right) &\leq O\left(n + \frac{nm_s^2}{n+m_s}\right) \\ &= O\left(n + \frac{nL}{n(f(w_s)-f_*) + \sqrt{(f(w_s)-f_*)L}}\right) \\ &\leq O\left(n + \frac{nL}{\epsilon n + \sqrt{\epsilon L}}\right), \end{aligned}$$

where we used the monotonicity of b_{k+1} with respect to k for the first inequality. Note that by Assumption 5, $\{V_{w_s}(x_*)\}_{s=1,2,\ldots}$ are uniformly bounded and notation O also hides $V_{w_s}(x_*)$. The above analysis implies the following theorem.

Theorem 6. *Consider AMSVRG under Assumptions 3 and 5. We set η, α_{k+1}, and τ_k as in (4.5). Let $b_{k+1} = \left\lceil \frac{n(k+2)}{p(n-1)+k+2} \right\rceil$ and $m_s = \left\lceil 4\sqrt{\frac{LV_{w_s}(x_*)}{q(f(w_s)-f_*)}} \right\rceil$, where p and q are small values described above. Then, the total complexity to run AMSVRG for $O(\log(1/\epsilon))$ outer iterations or to obtain an ϵ-accurate solution is*

$$O\left(\left(n + \frac{nL}{\epsilon n + \sqrt{\epsilon L}}\right)\log\left(\frac{1}{\epsilon}\right)\right).$$

Optimal Strongly Convex

Next, we consider the optimal strongly convex case. We assume that f is a μ-optimal-strongly convex function on $C \subset \mathbb{R}^d$. In this case, we choose the distance generating function $d(x) = \frac{1}{2}\|x\|^2$, so that the Bregman divergence becomes $V_x(y) = \frac{1}{2}\|x-y\|^2$.

Let the parameters be the same as in Theorem 6 with $x_* = \Pi_{X_*}(w_s)$ at stage s. Then, the expected value of the objective function is halved at every stage. Moreover, we assume that $\{w_s\}_{s=0,1,\dots} \subset C$. As mentioned in Section 2, for monotonic methods, we may consider the case where C is the sublevel set $\{x \in \mathbb{R}^d; f(x) \le c\}$ and this assumption holds for sufficiently large level. Since, by definition of optimal strong convexity, we have $m_s = \left\lceil 4\sqrt{\frac{L\|w_s - \Pi_{X_*}(w_s)\|^2}{2q(f(w_s)-f_*)}} \right\rceil \le \left\lceil 4\sqrt{\frac{\kappa}{q}} \right\rceil$, the complexity at each stage is

$$O\left(n + \sum_{k=0}^{m_s} b_{k+1}\right) \le O\left(n + \frac{n\kappa}{n+\sqrt{\kappa}}\right).$$

Thus, we have the following theorem.

Theorem 7. *Consider AMSVRG under Assumptions 3 and 4. Let parameters $\eta, \alpha_{k+1}, \tau_k, m_s$, and b_{k+1} be the same as those in Theorem 6 with $x_* = \Pi_{X_*}(w_s)$ at stage s. If $\{w_s\}_{s=0,1,\dots} \subset C$, then the total complexity for obtaining ϵ-accurate solution in expectation is*

$$O\left(\left(n + \frac{n\kappa}{n+\sqrt{\kappa}}\right)\log\left(\frac{1}{\epsilon}\right)\right),$$

and its iteration complexity is

$$O\left(\frac{n\sqrt{\kappa}}{n+\sqrt{\kappa}}\log\left(\frac{1}{\epsilon}\right)\right).$$

Table 4.1 lists the overall complexities of the AGD, SAG, SVRG, Acc-Prox-SVRG, Acc-SDCA, APCG, SPDC, and AMSVRG. The notation $\tilde{O}$ hides constant and logarithmic terms. By simple calculations, we see that

$$\frac{n\kappa}{n+\sqrt{\kappa}} = \frac{1}{2}H(\kappa, n\sqrt{\kappa}\,),$$
$$\frac{nL}{\epsilon n + \sqrt{\epsilon L}} = \frac{1}{2}H\left(\frac{L}{\epsilon}, n\sqrt{\frac{L}{\epsilon}}\right),$$

where $H(\cdot,\cdot)$ is the harmonic mean whose order is the same as $\min\{\cdot,\cdot\}$. Thus, as shown in Table 4.1, the complexity of AMSVRG is less than or equal to that of other methods in general convex and optimal strongly convex. Note that *Optimal Methods* in the table includes Acc-SDCA (Shalev-Shwartz and Zhang, 2014), APCG (Lin et al., 2014), SPDC (Zhang and Xiao, 2017).

Table 4.1: Comparison of total complexity.

Algorithm	General Convex	Optimal Strongly Convex	Strongly Convex
AGD	$\tilde{O}\left(n\sqrt{\frac{L}{\epsilon}}\right)$	$\tilde{O}\left(n\sqrt{\kappa}\right)$	$\tilde{O}\left(n\sqrt{\kappa}\right)$
SAG	$\tilde{O}\left(\frac{n+L}{\epsilon}\right)$	$\tilde{O}\left(\frac{n+L}{\epsilon}\right)$	$\tilde{O}\left(n+\kappa\right)$
SVRG	-	$\tilde{O}\left(n+\kappa\right)$	$\tilde{O}\left(n+\kappa\right)$
Acc-SVRG	-	-	$\tilde{O}\left(n+\kappa \wedge\ n\sqrt{\kappa}\right)$
Optimal Methods	-	-	$\tilde{O}\left(n+\kappa \wedge \sqrt{n\kappa}\right)$
AMSVRG	$\tilde{O}\left(n+\frac{L}{\epsilon}\wedge n\sqrt{\frac{L}{\epsilon}}\right)$	$\tilde{O}\left(n+\kappa \wedge n\sqrt{\kappa}\right)$	$\tilde{O}\left(n+\kappa \wedge n\sqrt{\kappa}\right)$

4.6.1 Fast Iteration Complexity and its Benefits

In the above convergence analyses, we find that our proposed method AMSVRG achieves the optimal iteration complexity for general, optimal strongly, and strongly convex problems with minibatch sizes of $\sqrt{L/\epsilon}$, $\sqrt{\kappa}$, and $\sqrt{\kappa}$, respectively by combining an accelerated gradient method and a stochastic variance reduced gradient. We note that the previous method (Nitanda, 2014) achieves it only for the strongly convex problem. As seen in the previous chapter, this feature of our acceleration scheme leads to some advantages: effective parallelization and better performance for linear-model on sparse dataset without using sparse structure.

Firstly, AMSVRG achieves the optimal iteration complexity with reasonable minibatch sizes as described above, while many existing methods cannot achieve this rate. This means, our method may become much faster more efficient than the other methods including optimal methods in terms of the total complexity, under the parallelization settings.

Next, we discuss the performance of AMSVRG for linear model that takes a form of $g_i(x) = l(a_i^T x)$ on sparse datasets $\{a_i\}_{i=1,\ldots,n}$. Since $\nabla g_i(x) = l'(a_i^T x)a_i$, some algorithms such as SGD and SVRG can be updated efficiently by using sparsity of a_i. It is unclear whether AMSVRG can be also implemented efficiently, but our acceleration scheme reduces the number of dense computations, consequently AMSVRG has the same complexity as sparse implementation of SVRG, without using sparse structure for problems with large condition number. Let d_0 be the maximum number of non-zero elements of a_i. Then, the total complexity including d and d_0 of AMSVRG is as follows:

$$\tilde{O}\left(nd_0 + m(b_m d_0 + d)\right) \leq \tilde{O}\left(nd_0 + \kappa\left(d_0 + \frac{d}{\sqrt{\kappa}}\right)\right),$$

where we used $m \leq O(\sqrt{\kappa})$ and $b_m \leq O(m) = O(\sqrt{\kappa})$. Hence, if the condition number is sufficiently large: $d/d_0 \leq \sqrt{\kappa}$, the total complexity is

$$\tilde{O}\left(d_0(n+\kappa)\right).$$

Therefore, AMSVRG efficiently performs on sparse datasets without an implementation trick.

4.6.2 Restart Scheme

The parameters of AMSVRG are essentially η, m_s, and b_{k+1} $(i.e.,\ p)$ because the appropriate values of both α_{k+1} and τ_k can be expressed by $\eta = 1/L$ as in (4.5). It may be difficult to choose an appropriate m_s which is the restart time for Algorithm 6. So, we propose heuristics for determining the restart time.

First, we suppose that the number of components n is sufficiently large such that the complexity of our method becomes $O(n)$. That is, for appropriate m_s, $O(n)$ is an upper bound on $\sum_{k=0}^{m_s} b_{k+1}$ (which is the complexity term). Therefore, we estimate the restart time as the minimum index $m \in \mathbb{Z}_+$ that satisfies $\sum_{k=0}^{m} b_{k+1} \geq n$. This estimated value is upper bound on m_s (in terms of the order). In this chapter, we call this restart method *R1*.

Second, we propose an adaptive restart method using SVRG. In a strongly convex case, we can easily see that if we restart the AGD for general convex problems every $\sqrt{\kappa}$, then the method achieves a linear convergence similar to that for strongly convex problems. The drawback of this restart method is that the restarting time depends on an unknown parameter κ, so several studies (O'Donoghue and Candes, 2015; Giselsson and Boyd, 2014; Su et al., 2014) have proposed effective adaptive restart methods. Moreover, Giselsson and Boyd (2014) showed that this technique also performs well for general convex problems. Inspired by their study, we propose an SVRG-based adaptive restart method called *R2*. That is, if

$$\langle v_{k+1}, y_{k+1} - y_k \rangle_2 > 0,$$

then we return y_k and start the next stage.

Third, we propose the restart method *R3*, which is a combination of the above two ideas. When $\sum_{k=0}^{m} b_{k+1}$ exceeds $10n$, we restart Algorithm 6, and when

$$(v_{k+1}, y_{k+1} - y_k) > 0 \ \wedge \ \sum_{k=0}^{m} b_{k+1} > n,$$

we return y_k and restart Algorithm 6.

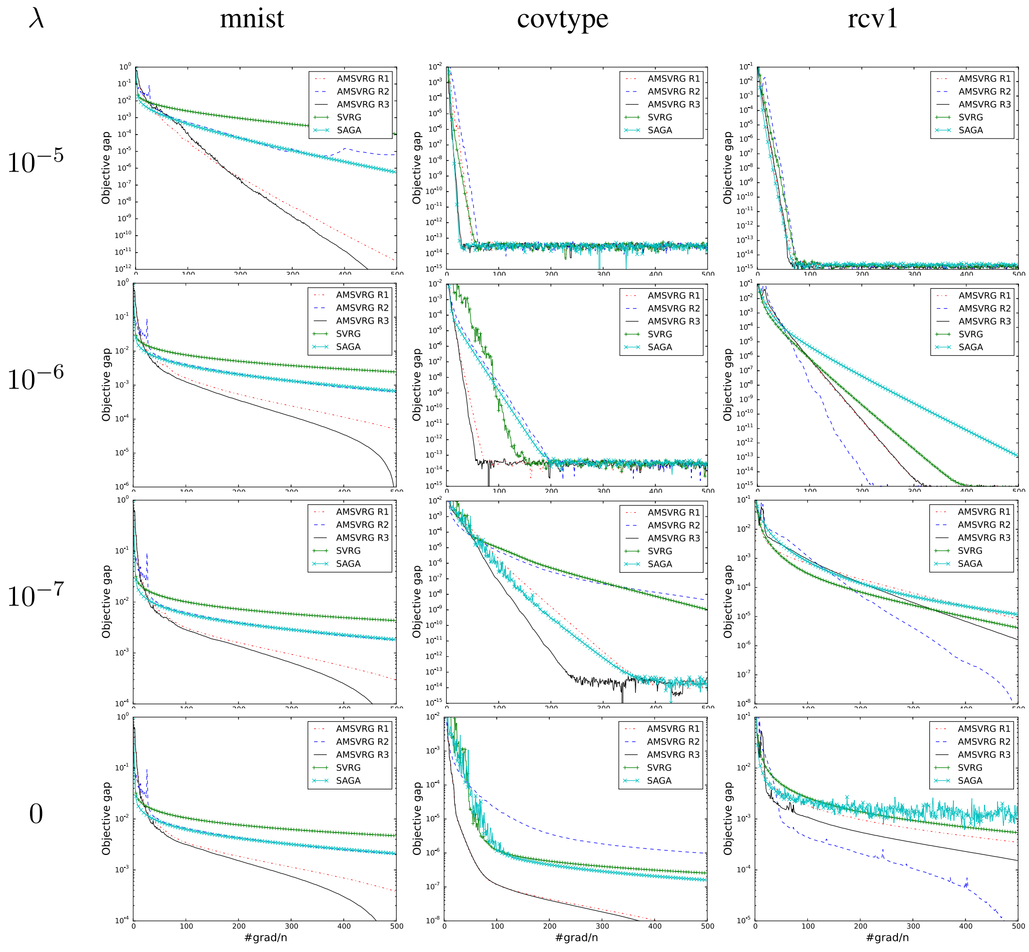

Figure 4.2: Comparison of algorithms applied to L_2-regularized logistic regularization (mnist, covtype, rcv1).

4.7 Numerical Experiments

In this section, we compare AMSVRG with SVRG and SAGA. We ran an L_2-regularized multi-class logistic regularization on *mnist* and *covtype* and ran an L_2-regularized binary-class logistic regularization on *rcv1*. The datasets and their descriptions can be found at the LIBSVM website[1]. In these experiments, we vary regularization parameter λ in $\{0,\ 10^{-7},\ 10^{-6},\ 10^{-5}\}$. We ran AMSVRG using some values of η from $[10^{-2},\ 5 \times 10]$ and p from $[10^{-1},\ 10]$, and then we chose the best η and p.

The results are shown in Figure 4.2. The horizontal axis is the number of single-component gradient evaluations. Our methods performed well and outperformed the other

[1]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

methods in some cases. For mnist and covtype, AMSVRG R1 and R3 converged quickly, and for rcv1, AMSVRG R2 worked very well. This tendency was more remarkable when the regularization parameter λ was small.

4.8 Appendix

4.8.1 Proof of the Proposition 3

We now prove the Proposition 3 that gives the condition of compactness of sublevel set.

Proof. Let $B^d(r)$ and $S^{d-1}(r)$ denote the ball and sphere of radius r, centered at the origin. By affine transformation, we can assume that X_* contains the origin 0, $X_* \subset B^d(1)$, and $X_* \cap S^{d-1}(1) = \phi$. Then, we have that for $\forall x \in S^{d-1}(1)$,

$$\langle \nabla f(x), x \rangle_2 \geq f(x) - f(0) > 0,$$

where we use convexity for the first inequality and $0 \in X_* \wedge x \notin X_*$ for the second inequality. We denote the minimum value of $(\nabla f(x), x)$ on $S^{d-1}(1)$ by α. Since $(\nabla f(x), x)$ is positive continuous, we have $\alpha > 0$. For $\forall r \geq 1$ and $\forall x \in S^{d-1}(r)$, we set $\hat{x} = x/r \in S^{d-1}(1)$, then it follows that

$$\begin{aligned} f(x) &\geq f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle_2 \\ &\geq f(\hat{x}) + (r-1) \langle \nabla f(\hat{x}), \hat{x} \rangle_2 \\ &\geq f_* + (r-1)\alpha \end{aligned}$$

This inequality implies that if $r > 1 + \frac{c - f_*}{\alpha}$, then we have $f(x) > c$ for $\forall x \in S^{d-1}(r)$. Therefore, sublevel set $\{x \in \mathbb{R}^d; f(x) \leq c\}$ is a closed bounded set. □

4.8.2 Proof of the Lemma 6

To prove Lemma 6, the following lemma is required, which is also shown in Freund (1971).

Lemma 10. *Let $\{\xi_i\}_{i=1}^n$ be a set of vectors in $\mathbb{R}^d$ and μ denote an average of $\{\xi_i\}_{i=1}^n$. Let I denote a uniform random variable representing a size b subset of $\{1, 2, \ldots, n\}$. Then, it follows that,*

$$\mathbb{E}_I \left\| \frac{1}{b} \sum_{i \in I} \xi_i - \mu \right\|^2 = \frac{n-b}{b(n-1)} \mathbb{E}_i \|\xi_i - \mu\|^2.$$

Proof. We denote a size b subset of $\{1, 2, \ldots, n\}$ by $S = \{i_1, \ldots, i_b\}$ and denote $\xi_i - \mu$ by $\tilde{\xi}_i$. Then,

$$\mathbb{E}_I \left\| \frac{1}{b} \sum_{i \in I} \xi_i - \mu \right\|^2 = \frac{1}{C(n,b)} \sum_S \left\| \frac{1}{b} \sum_{j=1}^{b} \xi_{i_j} - \mu \right\|^2$$

$$= \frac{1}{b^2 C(n,b)} \sum_S \left\| \sum_{j=1}^{b} \tilde{\xi}_{i_j} \right\|^2$$
$$= \frac{1}{b^2 C(n,b)} \sum_S \left(\sum_{j=1}^{b} \|\tilde{\xi}_{i_j}\|^2 + 2 \sum_{j,k,j<k} \tilde{\xi}_{i_j}^T \tilde{\xi}_{i_k} \right),$$

where $C(\cdot,\cdot)$ is a combination. By symmetry, an each $\tilde{\xi}_i$ appears $\frac{bC(n,b)}{n}$ times and an each pair $\tilde{\xi}_i^T \tilde{\xi}_j$ for $i < j$ appears $\frac{C(b,2)C(n,b)}{C(n,2)}$ times in $\sum_S$. Therefore, we have

$$\mathbb{E}_I \left\| \frac{1}{b} \sum_{i \in I} \xi_i - \mu \right\|^2 = \frac{1}{b^2 C(n,b)} \left(\frac{bC(n,b)}{n} \sum_{i=1}^{n} \|\tilde{\xi}_i\|^2 + \frac{2C(b,2)C(n,b)}{C(n,2)} \sum_{i,j,i<j} \tilde{\xi}_i^T \tilde{\xi}_j \right)$$
$$= \frac{1}{bn} \sum_{i=1}^{n} \|\tilde{\xi}_i\|^2 + \frac{2(b-1)}{bn(n-1)} \sum_{i,j,i<j} \tilde{\xi}_i^T \tilde{\xi}_j.$$

Since, $0 = \| \sum_{i=1}^{n} \tilde{\xi}_i \|^2 = \sum_{i=1}^{n} \|\tilde{\xi}_i\|^2 + 2 \sum_{i,j,i<j} \tilde{\xi}_i^T \tilde{\xi}_j$, we have

$$\mathbb{E}_I \left\| \frac{1}{b} \sum_{i \in I} \xi_i - \mu \right\|^2 = \left(\frac{1}{bn} - \frac{b-1}{bn(n-1)} \right) \sum_{i=1}^{n} \|\tilde{\xi}_i\|^2 = \frac{n-b}{b(n-1)} \frac{1}{n} \sum_{i=1}^{n} \|\tilde{\xi}_i\|^2.$$

This finishes the proof of Lemma. □

We now prove the Lemma 6.

Proof of Lemma 6 . We set $v_j^1 = \nabla g_j(x_k) - \nabla g_j(\tilde{x}) + \tilde{v}$. Using Lemma A and

$$v_k = \frac{1}{b} \sum_{j \in I_k} v_j^1,$$

conditional variance of v_k is as follows

$$\mathbb{E}_{I_k} \|v_k - \nabla f(x_k)\|^2 = \frac{1}{b} \frac{n-b}{n-1} \mathbb{E}_j \|v_j^1 - \nabla f(x_k)\|^2,$$

where expectation in right hand side is taken with respect to $j \in \{1, \ldots, n\}$. By Corollary 3 in Xiao and Zhang (2014), it follows that,

$$\mathbb{E}_j \|v_j^1 - \nabla f(x_k)\|^2 \le 4L(f(x_k) - f(x_*) + f(\tilde{x}) - f(x_*)).$$

This completes the proof of Lemma 6. □

4.8.3 Stochastic gradient descent analysis

Below is the proof of Lemma 8.

Proof of Lemma 8 . It is clear that y_k is equal to $x_k - \eta v_k$. Since $f(x)$ is L-smooth and $\eta = \frac{1}{L}$, we have,

$$\begin{aligned} f(y_k) &\le f(x_k) + \langle \nabla f(x_k), y_k - x_k \rangle_2 + \frac{L}{2}\|y_k - x_k\|^2 \\ &= f(x_k) - \frac{1}{L}\langle \nabla f(x_k), v_k \rangle_2 + \frac{1}{2L}\|v_k\|^2. \end{aligned}$$

v_k is an unbiased estimator of gradient $\nabla f(x_k)$, that is, $\mathbb{E}_{I_k}[v_k] = \nabla f(x_k)$. Hence, we have

$$\mathbb{E}_{I_k}\|v_k\|^2 = \|\nabla f(x_k)\|^2 + \mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2.$$

Using above two expressions, we get

$$\begin{aligned} \mathbb{E}_{I_k}[f(y_k)] &= f(x_k) - \frac{1}{L}\|\nabla f(x_k)\|^2 + \frac{1}{2L}\mathbb{E}_{I_k}\|v_k\|^2 \\ &= f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2 + \frac{1}{2L}\mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2. \end{aligned}$$

□

4.8.4 Stochastic mirror descent analysis

We give the proof of Lemma 9.

Proof of Lemma 9 . The following are basic properties of Bregman divergence.

$$\langle \nabla V_x(y), u - y \rangle_2 = V_x(u) - V_y(u) - V_x(y), \tag{4.6}$$

$$V_x(y) \ge \frac{1}{2}\|x - y\|^2. \tag{4.7}$$

Using (4.6) and (4.7), we have

$$\begin{aligned} \alpha_k \langle v_k, z_{k-1} - u \rangle_2 &= \alpha_k \langle v_k, z_{k-1} - z_k \rangle_2 + \alpha_k (v_k, z_k - u) \\ &= \alpha_k \langle v_k, z_{k-1} - z_k \rangle_2 - \left\langle \nabla V_{z_{k-1}}(z_k), z_k - u \right\rangle_2 \\ &\underset{(4.6)}{=} \alpha_k \langle v_k, z_{k-1} - z_k \rangle_2 + V_{z_{k-1}}(u) - V_{z_k}(u) - V_{z_{k-1}}(z_k) \\ &\underset{(4.7)}{\le} \alpha_k \langle v_k, z_{k-1} - z_k \rangle_2 - \frac{1}{2}\|z_{k-1} - z_k\|^2 + V_{z_{k-1}}(u) - V_{z_k}(u) \\ &\le \frac{1}{2}\alpha_k^2 \|v_k\|^2 + V_{z_{k-1}}(u) - V_{z_k}(u), \end{aligned}$$

where for the second equality we use stochastic mirror descent step, that is, $\alpha_k v_k + \nabla V_{z_{k-1}}(z_k) = 0$ and for the last inequality we use the Fenchel-Young inequality $\alpha_k \langle v_k, z_{k-1} - z_k\rangle_2 \le \frac{1}{2}\alpha_k^2\|v_k\|^2 + \frac{1}{2}\|z_{k-1} - z_k\|^2$.

By taking expectation with respect to I_k and using $\mathbb{E}_{I_k}\|v_k\|^2 = \|\nabla f(x_k)\|^2 + \mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2$, we have

$$\alpha_k \langle \nabla f(x_k), z_{k-1} - u\rangle_2 \le V_{z_{k-1}}(u) - \mathbb{E}_{I_k}[V_{z_k}(u)] + \frac{1}{2}\alpha_k^2\|\nabla f(x_k)\|^2 + \frac{1}{2}\alpha_k^2\mathbb{E}_{I_k}\|v_k - \nabla f(x_k)\|^2.$$

This finishes the proof of Lemma 9. □

4.8.5 Proof of the Lemma 7

We now prove the Lemma 7 that is the key to the analysis of our method.

Proof. We denote $V_{z_k}(x_*)$ by V_k for simplicity. We get

$$\begin{aligned}
&\alpha_{k+1}\langle \nabla f(x_{k+1}), z_k - x_*\rangle_2 \\
&\le V_k - \mathbb{E}_{I_{k+1}}[V_{k+1}] + L\alpha_{k+1}^2(f(x_{k+1}) - \mathbb{E}_{I_{k+1}}[f(y_{k+1})]) + \alpha_{k+1}^2\mathbb{E}_{I_{k+1}}\|v_{k+1} - \nabla f(x_{k+1})\|^2 \\
&\le V_k - \mathbb{E}_{I_{k+1}}[V_{k+1}] + L\alpha_{k+1}^2(f(x_{k+1}) - \mathbb{E}_{I_{k+1}}[f(y_{k+1})]) \\
&\quad + 4L\alpha_{k+1}^2\delta_{k+1}(f(x_{k+1}) - f(x_*) + f(y_0) - f(x_*)) \\
&= V_k - \mathbb{E}_{I_{k+1}}[V_{k+1}] + (1 + 4\delta_{k+1})L\alpha_{k+1}^2(f(x_{k+1}) - f(x_*)) - L\alpha_{k+1}^2\mathbb{E}_{I_{k+1}}[f(y_{k+1}) - f(x_*)] \\
&\quad + 4L\alpha_{k+1}^2\delta_{k+1}(f(y_0) - f(x_*)),
\end{aligned}$$

where for the first inequality we use Lemma 8 and 9 with $u = x_*$, for the second inequality we use Lemma 6.

By taking the expectation with respect to the history of random variables $I_1, I_2 \ldots$, we have,

$$\begin{aligned}
\alpha_{k+1}\mathbb{E}[\langle \nabla f(x_{k+1}), z_k - x_*\rangle_2] &\le \mathbb{E}[V_k - V_{k+1}] + (1 + 4\delta_{k+1})L\alpha_{k+1}^2\mathbb{E}[f(x_{k+1}) - f(x_*)] \\
&\quad - L\alpha_{k+1}^2\mathbb{E}[f(y_{k+1}) - f(x_*)] + 4L\alpha_{k+1}^2\delta_{k+1}(f(y_0) - f(x_*)),
\end{aligned} \tag{4.8}$$

and we get

$$\begin{aligned}
&\sum_{k=0}^{m}\alpha_{k+1}\mathbb{E}[f(x_{k+1}) - f(x_*)] \\
&\qquad \le \sum_{k=0}^{m}\alpha_{k+1}\mathbb{E}[\langle \nabla f(x_{k+1}), x_{k+1} - x_*\rangle_2] \\
&\qquad = \sum_{k=0}^{m}\alpha_{k+1}(\mathbb{E}[\langle \nabla f(x_{k+1}), x_{k+1} - z_k\rangle_2] + \mathbb{E}[\langle \nabla f(x_{k+1}), z_k - x_*\rangle_2])
\end{aligned}$$

$$= \sum_{k=0}^{m} \alpha_{k+1} \left(\frac{1-\tau_k}{\tau_k} \mathbb{E}[\langle \nabla f(x_{k+1}), y_k - x_{k+1} \rangle_2] + \mathbb{E}[\langle \nabla f(x_{k+1}), z_k - x_* \rangle_2] \right)$$
$$\leq \sum_{k=0}^{m} \left(\alpha_{k+1} \frac{1-\tau_k}{\tau_k} \mathbb{E}[f(y_k) - f(x_{k+1})] + \alpha_{k+1} \mathbb{E}[\langle \nabla f(x_{k+1}), z_k - x_* \rangle_2] \right). \tag{4.9}$$

Using (4.8), (4.9), and $V_{z_{k+1}}(x_*) \geq 0$, we have

$$\sum_{k=0}^{m} \alpha_{k+1} \left(1 + \frac{1-\tau_k}{\tau_k} - (1 + 4\delta_{k+1}) L \alpha_{k+1} \right) \mathbb{E}[f(x_{k+1}) - f(x_*)]$$
$$\leq V_0 + \sum_{k=0}^{m} \alpha_{k+1} \frac{1-\tau_k}{\tau_k} \mathbb{E}[f(y_k) - f(x_*)] - L \sum_{k=0}^{m} \alpha_{k+1}^2 \mathbb{E}[f(y_{k+1}) - f(x_*)]$$
$$+ 4L \sum_{k=0}^{m} \alpha_{k+1}^2 \delta_{k+1} (f(y_0) - f(x_*)).$$

This completes the proof of Lemma 7. □

4.8.6 Proof of Theorem 5

We here give the proof of convergence theorem of AMSVRG.

Proof of Theorem 5. Using Lemma 7 and

$$\tau_0 = 1, \frac{1}{\tau_k} - (1 + 4\delta_{k+1}) L \alpha_{k+1} \geq 0,$$
$$\alpha_{k+1} \frac{1-\tau_k}{\tau_k} - L\alpha_k^2 = L\alpha_{k+1}^2 - \frac{1}{2}\alpha_{k+1} - L\alpha_k^2$$
$$= -\frac{1}{16L} < 0,$$

we have

$$L\alpha_{m+1}^2 \mathbb{E}[f(y_{m+1}) - f_*] \leq V_{z_0}(x_*) + 4L \sum_{k=0}^{m} \alpha_{k+1}^2 \delta_{k+1} (f(y_0) - f_*).$$

This proves the theorem because $4L \sum_{k=0}^{m} \alpha_{k+1}^2 \delta_{k+1} \leq p \sum_{k=0}^{m} \alpha_{k+1} \leq \frac{5p}{32L}(m+2)^2$. □

Algorithm 8 Modified Multi-Stage AMSVRG

```
**Input:** the number of outer-iterations $T$, the number of inner-iterations $(m_s)_{s=0}^T$, learning rates $\eta$, $(\alpha_{k+1})_{k\in\mathbb{Z}_+}$, mini-batch sizes $(b_{k+1})_{k\in\mathbb{Z}_+}$, coefficients $(\tau_k)_{k\in\mathbb{Z}_+}$, and initial point $w_0$
for $s = 0$ to $T$ do
    $y_0 \leftarrow w_s,\ z_0 \leftarrow w_0$
    $w_{s+1} \leftarrow \textbf{Algorithm6}(m_s,\ \eta,\ (\alpha_{k+1})_{k\in\mathbb{Z}_+},\ (b_{k+1})_{k\in\mathbb{Z}_+},\ (\tau_k)_{k\in\mathbb{Z}_+}, y_0,\ z_0)$
end for
Return $w_{T+1}$
```

4.8.7 Modified AMSVRG for general convex problems

We now introduce a modified AMSVRG (described in Algorithm 8) that does not need the boundedness assumption for general convex problems.

We set η, α_{k+1}, and τ_k as in (4.5). Let $b_{k+1} \in \mathbb{Z}_+$ be the minimum values satisfying $4L\delta_{k+1}\alpha_{k+1} \le p$ for small p $(e.g.\ 1/4)$. Let $m_s = \left\lceil 4\sqrt{\frac{LV_{z_0}(x_*)}{\epsilon}} \right\rceil$.

From Theorem 5, we get

$$\mathbb{E}[f(w_{s+1}) - f(x_*)] \le \epsilon + a(f(w_s) - f(x_*)),$$

where $a = \frac{5}{2}p$. Thus, it follows that,

$$\begin{aligned}\mathbb{E}[f(w_{s+1}) - f(x_*)] &\le \sum_{t=0}^{s} a^t\epsilon + a^{s+1}(f(w_0) - f(x_*)) \\ &\le \frac{1}{1-a}\epsilon + a^{s+1}(f(w_0) - f(x_*)).\end{aligned}$$

Hence, running the modified AMSVRG for $O\left(\log\frac{1}{\epsilon}\right)$ outer iterations achieves ϵ-accurate solution in expectation, and a complexity at each stage is

$$\begin{aligned}O\left(n + \sum_{k=0}^{m_s} b_{k+1}\right) &\le O\left(n + \frac{nm_s^2}{n+m_s}\right) \\ &= O\left(n + \frac{nL}{\epsilon n + \sqrt{\epsilon L}}\right) \\ &= O\left(n + \min\left\{\frac{L}{\epsilon}, n\sqrt{\frac{L}{\epsilon}}\right\}\right),\end{aligned}$$

where we used the monotonicity of b_{k+1} with respect to k for the first inequality. Note that $V_{z_0}(x_*)$ is constant (i.e. $V_{w_0}(x_*)$), and O hides this term. From the above analysis, we derive the following theorem.

Theorem 8. *Consider the modified AMSVRG under Assumptions 3. Let parameters be as above. Then the overall complexity for obtaining ϵ-accurate solution in expectation is*

$$O\left(\left(n+\min\left\{\frac{L}{\epsilon}, n\sqrt{\frac{L}{\epsilon}}\right\}\right)\log\left(\frac{1}{\epsilon}\right)\right).$$

Chapter 5

Stochastic Difference of Convex Algorithm

Difference of convex functions (DC) programming is an important approach to nonconvex optimization problems because these structures can be encountered in several fields. Effective optimization methods, called DC algorithms, have been developed in deterministic optimization literature. In machine learning, a lot of important learning problems such as the Boltzmann machines (BMs) can be formulated as DC programming. However, there is no DC-like algorithm guaranteed by convergence rate analysis for stochastic problems that are more suitable settings for machine learning tasks. In this chapter, we propose a stochastic variant of DC algorithm and give computational complexities to converge to a stationary point under several situations. Moreover, we show our method includes expectation-maximization (EM) and Monte Carlo EM (MCEM) algorithm as special cases on training BMs. In other words, we extend EM/MCEM algorithm to more effective methods from DC viewpoint with theoretical convergence guarantees. Experimental results indicate that our method performs well for training binary restricted Boltzmann machines and deep Boltzmann machines without pre-training.

This chapter is based on the work *Stochastic Difference of Convex Algorithm and its Application to Training Deep Boltzmann Machines*, A. Nitanda and T. Suzuki, Artificial Intelligence and Statistics, 2017 (Nitanda and Suzuki, 2017a).

5.1 Overview

There is a strong need to develop better optimization methods for nonconvex problems because many scientific problems are nonconvex. Generally speaking, a nonconvex problem is hard to solve. However, several important problems possess a special structure (quadratic, finite sums, etc.), and it is expected that we can build effective algorithms by making full use of the special structure. In particular, a wide range of problems are re-

Table 5.1: Complexities of SPD

	General case	Smooth concave	Polyak-Łojasiewicz
Outer Iteration Complexity	$O(L_g/\epsilon)$	$O(\min\{L_g, L_h\}/\epsilon)$	$\tilde{O}(CL_g)$
Total Complexity (general)	$O(L_g/\epsilon^2)$	$O(L_g/\epsilon^2)$	$\tilde{O}\left(\frac{CL_g}{\epsilon}\right)$
Total Complexity (variance growth condition)	$\tilde{O}\left(\frac{L_g(1+\beta)}{\epsilon}\right)$	$\tilde{O}\left(\frac{L_g(1+\beta)}{\epsilon}\right)$	$\tilde{O}\left(CL_g(1+\beta)\right)$

duced to *difference of convex functions* (DC) programming (Tao, 1986) which takes the the following form:

$$\min_{x\in\mathbb{R}^d}\{f(x) \overset{\text{def}}{=} g(x) - h(x)\}, \tag{5.1}$$

where g and h are differentiable convex functions from $\mathbb{R}^d$ to $\mathbb{R}$.

In fact, DC structures can be encountered in several fields, e.g., in economics, finance, operations research, and biology. In machine learning, multiple kernel learning (Argyriou et al., 2006) and feature selection in support vector machines (Le Thi et al., 2008) are formulated as DC programs. Moreover, it is shown that: (i) any continuous function over a compact set can be approximated by a DC function by Stone-Weierstrass theorem and DC decomposition of polynomials (ii) any C^2-function f whose eigenvalues of Hessian are lower bounded can be decomposed as a DC function; there exists a convex function h such that $f = g-h$ is DC, where $g = f+h$. The former property is induced by the fact that any continuous function on a compact set is a uniform limit of a sequence of polynomials according to the Stone-Weierstrass theorem and, a polynomial can be decomposed as the sum of a convex polynomial and a concave polynomial (Ferrer, 2001; Wang et al., 2014; Ahmadi and Hall, 2015).

To solve optimization problem (5.1), practical methods are variants of DC algorithms (DCAs) (Tao, 1986) that generate a sequence by solving sub-problems that consist of the sum of the convex part g and the linear approximation of the concave part $-h$ at the current iterate. Due to their simplicity, efficiency, and robustness, DCAs have been widely applied to many fields. It has been shown that a DCA converges to a stationary point even when an objective function is non-differentiable. Moreover, convergence rates have been given (Le Thi et al., 2009; Le Thi and Dinh, 2011) for a special class of DC programs.

Important applications of DC programming are *Boltzmann machines* (BMs) which are energy-based generative models over binary observations and binary hidden units. Restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs) (Salakhutdinov and Hinton, 2009) are special forms of BMs. These models are used for unsupervised learning, dimension reduction, feature extraction, and pre-training or initialization of multi-layer perceptrons. RBMs and DBMs learning are easier than more general Boltzmann machine learning; however, it is still quite difficult. In fact, in recent years, several studies have exploited optimization methods (Salakhutdinov and Hinton, 2009; Cho et al., 2013; Carlson et al., 2015). The log-likelihood of a BM is the subtraction of two composite functions of linear mapping and a log-sum-exp function. That is, BM learning is DC programming. However, there is still no DC-like algorithm with any convergence rate analysis to solve stochastic problems that are more suitable settings for training BMs.

In this chapter, we propose a *stochastic proximal DC algorithm (SPD)*. Our method works effectively not only under a deterministic setting but also under a stochastic setting, where only stochastic gradients are available for the convex part and sometimes for the concave part. Optimization methods built under this setting can be applied to a wider class of problems, including training BMs. Furthermore, we show that Expectation-maximization (EM) and Monte Carlo EM (MCEM) algorithms, which are heavily used for latent variable models, are recognized as special cases of our SPD algorithm, and our algorithm is even more effective than these algorithms.

In addition, we give convergence complexities: the number of iterations of SPD to obtain an ϵ-accurate solution in expectation (i.e., $\mathbb{E}\|\nabla f(x)\|_2^2 < \epsilon$) under several settings: Lipschitz smoothness of g, h and Polyak-Łojasiewicz condition on objective function f, whose definitions will be described in Section 5.4.

SPD requires only approximate solutions of sub-problems in expectation. To solve the sub-problems we can employ stochastic optimization methods for convex problems, which is a very active research area. Moreover, since a sub-problem becomes strongly convex, effective stochastic gradient-based methods (Rakhlin et al., 2012; Chen et al., 2012; Ghadimi and Lan, 2013a; Bottou et al., 2016) can be used as underlying solvers to achieve fast convergence, and we give the total complexity analyses that include the complexity of such a method.

Table 5.1 shows the complexities of our method in general case (g : L_g-smooth), smooth concave case (g, h : L_g, L_h-smooth), and Polyak-Łojasiewicz case. Note that the notation $\tilde{O}$ hides a logarithmic term. The middle row of Table 5.1 lists the total complexities without additional structures for the sub-problem. RSG (Ghadimi and Lan, 2013b) is a stochastic optimization method for solving Lipschitz smooth nonconvex problems, and in this case, we can obtain the same complexity $O(L_g/\epsilon^2)$ as SPD by slight modification of their proof. However, SPD has better practical performance than suggested by the theory because our analyses for the total complexities do not take into account the warm starting for sub-problems solved in SPD repeatedly. An intuition for the practical performance of SPD is described in Section 5.4.

Moreover, if the convex sub-problems have additional special structures such as 2nd-order derivative, noise condition, finite sums, it is possible to show much better convergence by utilizing this information for the convex optimization method used in the inner loop. Especially, we focus on a *variance growth condition* (Bottou et al., 2016; Carlson et al., 2016), defined in Section 5.4 and we show that this condition strictly improves the total complexities as shown in the last row of Table 5.1.

Related Works

The stochastic majorization-minimization method and the online DCA were proposed in Mairal (2013) for nonconvex problems. Although SPD is also a type of majorization-minimization methods, it differs from their methods in several respects. The surrogate function used in Mairal (2013) is more stochastic and is quadratically approximated, and can be solved exactly. On the other hand, the convex part is not approximated in SPD and it is relatively difficult to solve our surrogate function exactly; SPD only requires an approximation to the solution in expectation. Moreover, although they gave convergence analyses, convergence rates for the nonconvex problem were not provided.

The method proposed in Nesterov (2013) for deterministic nonconvex problems is one of the methods which should be compared with our method. Applying their analysis to our problem, the complexity to obtain a *gradient mapping* of norm ϵ is $O(L_h/\epsilon)$. However, the norm of a gradient and the norm of a gradient mapping cannot be directly compared. Furthermore, while the coefficient of their order is always affected by L_h, our method is free from it when $L_h > L_g$.

5.2 DC Algorithm

DCAs are optimization algorithms for solving DC problems. To obtain the next iterate that linearly approximates the concave part $-h$ at the current iterate x_k and solves the resulting convex minimization problem:

$$\min_{x\in\mathbb{R}^d}\{g(x) - (h(x_k) + \langle \nabla h(x_k), x - x_k\rangle)\} \sim \min_{x\in\mathbb{R}^d}\{g(x) - \langle \nabla h(x_k), x\rangle\}, \tag{5.2}$$

where $\langle , \rangle$ denotes the Euclidean inner product. Several studies have exploited the convergence properties and shown the efficiency of a DCA under the assumption that we can obtain an exact or deterministically approximate solution of the sub-problem (5.2). However, there is no algorithm with convergence rate analysis for stochastic problems frequently encountered in machine learning. Thus in the next section, we propose a more suitable DCA for such problems.

5.3 Stochastic Proximal DC Algorithm

In the remainder of this chapter, we make the following stochastic assumption.

Assumption 6 (Stochastic Assumption)**.** *To solve DC problem (5.1), an optimization algorithm can use only the stochastic gradients of g and h.*

Note that although there are several problems such that a deterministic gradient of h can be computed and we can use ∇h, we also make stochastic assumption on h to handle some specific problems including general BMs. Here, we propose SPD, which is more suitable for this assumption. Let H_k denote the $d \times d$ positive definite matrix and $\|\cdot\|_{H_k}$ denote the Mahalanobis norm defined by H_k, i.e., for $v \in \mathbb{R}^d$, $\|v\|_{H_k} = \sqrt{\langle v, H_k v\rangle}$. Let $v_h(x)$ denote an unbiased estimator of $\nabla h(x)$ and σ_h^2 be an upper bound on the variance of v_h; $\mathbb{E}[v_h(x)] = \nabla h(x)$, $\mathbb{E}[\|v_h(x) - \nabla h(x)\|_2^2] \le \sigma_h^2$. Let x_k be the current iterate. To obtain the next iterate x_{k+1}, SPD solves the following proximal sub-problem inexactly by a stochastic method:

$$SP(k) : \min_{x\in\mathbb{R}^d}\{\phi_k(x) \overset{\text{def}}{=} g(x) + \frac{1}{2}\|x - x_k\|_{H_k}^2 - (h(x_k) + \langle v_h(x_k), x - x_k\rangle)\}. \tag{5.3}$$

The difference between sub-problems (5.2) and (5.3) is that the latter problem is a stochastic approximation and includes the proximal term $\frac{1}{2}\|x - x_k\|_{H_k}^2$, which forces the solution to stay close to x_k with respect to the norm $\|\cdot\|_{H_k}$. Practical choices for the metric H_k and our motivations are described in the next subsection. Since it is impractical to obtain an exact or deterministic approximation to the solution, we employ the following condition on expectation of a solution of the sub-problem:

$$\mathbb{E}[\phi_k(x_{k+1})|\mathcal{F}_k] \le \phi_k^* + \delta, \tag{5.4}$$

where $\mathcal{F}_k$ is the filtration for all information up to iterate x_k, ϕ_k^* is the optimal value of *SP(k)*, and $\delta > 0$. For many stochastic algorithms, e.g., stochastic gradient descent, the global convergence property in expectation is shown for convex problems. Therefore, we can use such algorithms as an underlying solver of SPD. Note that we can warm start to solve sub-problems, i.e., by running a stochastic algorithm from a previous solution x_k with sufficiently small learning rates, it is not particularly difficult to satisfy the above condition empirically. In Section 5.4, we demonstrate how condition (5.4) guarantees the convergence of SPD with better convergence rates to obtain an ϵ-accurate solution in expectation. Here, we briefly give a connection between SPD and mirror descent method. Let $x_{k+1}^* = \arg\min \phi_k(x), \psi_k(x) \overset{\text{def}}{=} g(x) + \frac{1}{2}\|x\|_{H_k}^2$, and assume $v_h(x_k) = \nabla h(x_k)$, then we have

$$\nabla f(x_k) = \nabla\psi_k(x_k) - \nabla\psi_k(x_{k+1}^*). \tag{5.5}$$

This equation means SPD can also be interpreted as an inexact variant of a stochastic mirror descent method using distance generating functions ψ_k for DC programming.

SPD runs for R iterations, where R is chosen uniformly at random from $\{1, 2, \dots, M\}$ for $M \in \mathbb{Z}_+$. This is a standard technique for nonconvex analysis (Ghadimi and Lan, 2013b). SPD is described in Algorithm 9.

Algorithm 9 SPD (Stochastic proximal DC algorithm)

```
Input: initial point $x_1$, the maximum number of iterations $M$, underlying solver $\mathcal{A}$ for solving $SP(k)$, the number of iterations $T$ for $\mathcal{A}$
Randomly pick up $R \in \{1, 2, \dots, M\}$
for $k = 1$ to $R - 1$ do
    Update the metric $H_k$
    Compute stochastic approximation $v_h(x_k)$ of $\nabla h(x_k)$
    $x_{k+1} \leftarrow$ Solve SP(k) by running $\mathcal{A}$ for $T$ iterations
end for
Return $x_R$
```

5.3.1 Metrics

There are two aims for including the proximal term $\frac{1}{2}\|x - x_k\|^2_{H_k}$ of ϕ_k in sub-problems. The first is to keep the next iterate x_{k+1} in a neighborhood of the current x_k where the linear approximation of the concave part $-h$ is sufficiently accurate. In the gradient-based optimization literature, it is well studied theoretically and empirically that the proximity induced by an appropriate metric at each iteration improves the convergence behavior, e.g., Natural Gradient (Amari, 1998) and AdaGrad (Duchi et al., 2011). The second is to enhance the effect of strong convexity, which makes the sub-problem *SP(k)* better conditioned and easier to solve.

Next, we give practical choices for the metric H_k. The first choice is a scalar matrix, i.e., $H_k = \mu I_d,\ \mu > 0$. As will be discussed in Section 5.4, this choice with $\mu = L_g$ or L_h, where L_g, L_h are smoothness parameters, gives a better convergence complexity according to our analysis. Second, when the concave part $-h$ is twice differentiable, we propose a diagonal approximation to the Hessian of h (Becker and Le Cun, 1988). In other words, we define H_k as follows:

$$H_k \leftarrow \left|\operatorname{diag}\left(\nabla^2 h(x_k)\right)\right| + \mu I_d, \tag{5.6}$$

where the absolute value operator $|\cdot|$ is applied element-wise to the diagonal of the Hessian and μ is a positive value that guarantees sufficiently strong convexity to improve the conditioning of the curvature of h. This metric makes the update take large steps in the direction of low curvature compared to that of highly curved directions.

5.3.2 AdaSPD

Here, we derive a specific form of the SPD described by Algorithm 9. For a metric H_k, we use a scalar matrix or the diagonal Hessian (5.6) as in the previous subsection. For an underlying solver, due to the simplicity of implementation and better empirical performance, we adopt AdaGrad using the proximal term $\frac{1}{2}\|x - x_k\|^2_{H_k}$ as the regularization in its update. Let $y_{k,t}$ and $v_{k,t}$ $(t = 1, 2, \dots)$ denote an inner iterate and a stochastic gradient of g at $y_{k,t}$, respectively, in outer iteration k. To adapt the step size to the geometry of the objective function, AdaGrad computes diagonal matrix $D_{k,t}$ as follows:

$$D_{k,t} \leftarrow \sqrt{\lambda I_d + \operatorname{diag}(\textstyle\sum_{i=1}^{t} s_{k,i} s_{k,i}^\top)},$$

where λ is a damping parameter for numerical stability and $s_{k,i}$ denotes $v_{k,i} - v_h(x_k)$. To obtain the next inner iterate $y_{k,t+1}$, we solve the following problem:

$$\arg\min_{y \in \mathbb{R}^d} \left\{ \langle s_{k,t}, y \rangle + \frac{1}{2} \|y - x_k\|_{H_k}^2 + \frac{1}{2\eta} \|y - y_{k,t}\|_{D_{k,t}}^2 \right\},$$

where η denotes the learning rate. Note that $D_{k,t}$ can be updated successively and $y_{k,t+1}$ can be computed in closed form. The algorithm *AdaSPD* is described in Algorithm 10.

Algorithm 10 AdaSPD

```
**Input:** initial point $x_1$, the maximum number of iterations $M$, (lower) scale $\mu$ of a metric $H_k$, the number of iterations $T$ for the inner loop, damping parameter $\lambda$ of $D_{k,t}$, learning rate $\eta > 0$, suffix averaging parameter $\alpha \in (0, 1)$ (assuming $\alpha T$ is an integer)
Randomly pick a $R \in \{1, 2, \dots, M\}$
**for** $k = 1$ **to** $R - 1$ **do**
  **scalar matrix option:**
    $H_k \leftarrow \mu I_d$
  **Diagonal Hessian option:**
    $H_k \leftarrow |\operatorname{diag}(\nabla^2 h(x_k))| + \mu I_d$
  $y_{k,1} \leftarrow x_k$
  $S_{k,0} \leftarrow O$
  **for** $t = 1$ **to** $T - 1$ **do**
    $v_{k,t} \leftarrow$ a stochastic gradient of $g$ at $y_{k,t}$
    $s_{k,t} \leftarrow v_{k,t} - v_h(x_k)$
    $S_{k,t} \leftarrow S_{k,t-1} + \operatorname{diag}(s_{k,t} s_{k,t}^\top)$
    $D_{k,t} \leftarrow \sqrt{\lambda I_d + S_{k,t}}$
    $y_{k,t+1} \leftarrow (\eta H_k + D_{k,t})^{-1}(\eta H_k x_k + D_{k,t} y_{k,t} - \eta s_{k,t})$
  **end for**
  $x_{k+1} \leftarrow \dfrac{\sum_{t=(1-\alpha)T+1}^{T} y_{k,t}}{\alpha T}$
**end for**
Return $x_R$
```

5.4 Analysis

In this section, we give convergence analyses of SPD and complexities to obtain an ϵ-accurate solution in expectation under several situations. Note that all proofs can be found in the supplement. For simplicity, we only consider the scalar matrix $\mu_k I_d$ for H_k. We first give the definition of Lipschitz smoothness needed for analyses.

Definition 3. *A function ϕ is Lipschitz smooth if there exists $L_\phi > 0$ such that $\forall x, \forall y \in \mathbb{R}^d$,*

$$\|\nabla\phi(x) - \nabla\phi(y)\| \le L_\phi \|x - y\|_2.$$

5.4.1 General Case

The following proposition shows the expected square norm of the gradient is upper-bounded by the expected reduction of the objective function per iteration up to δ and σ_h^2.

Proposition 4. *Consider Algorithm 9 under stochastic assumption 6. Suppose g is L_g-smooth and the expected condition (5.4) holds. Then, it follows that for $k = 1, 2, \ldots$*

$$\frac{\mu_k}{4}\mathbb{E}\left[\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k\right] + \frac{\|\nabla f(x_k)\|_2^2}{2(L_g + \mu_k)} \le \delta + \frac{\sigma_h^2}{\mu_k} + \mathbb{E}[f(x_k) - f(x_{k+1}) | \mathcal{F}_k].$$

Using Proposition 4, we derive a convergence theorem.

Theorem 9. *Make the same assumption as Proposition 4 and assume the optimal value f_* of f is bounded from below. Let $\mu_k = O(L_g)$ and ($\mu_k = \Omega(L_g)$ or $\sigma_h = 0$). Then it follows that*

$$\mathbb{E}[\|\nabla f(x_R)\|_2^2] \le O\left(L_g\delta + \sigma_h^2 + \frac{L_g(f(x_1) - f_*)}{M}\right).$$

We immediately obtain the following corollary.

Corollary 5. *Suppose the assumptions in Theorem 9 hold and $\sigma_h^2 = O(\epsilon)$. Set $\delta = O(\epsilon/L_g)$. Then, the complexity M to obtain an ϵ-accurate solution in expectation is $O(L_g/\epsilon)$.*

The readers might feel that the assumption $\sigma_h^2 = O(\epsilon)$ in the above corollary is unrealistic because the variance σ_h^2 of the stochastic gradient of h is assumed to be smaller than the solution accuracy ϵ. However, this is reasonable because the total complexity is unchanged even if we spend the same computational cost as that of solving $SP(k)$ to estimate ∇h and the variance σ_h^2 can be made sufficiently small by using a comparable number of samples in the mini-batch.

5.4.2 Smooth Concave Function

In this subsection, we give the convergence properties for problems having Lipschitz smooth h. To establish a complexity analysis, we slightly modify the algorithm: we choose R, the number of iterations of SPD, uniformly at random from $\{2, 3, \ldots, M+1\}$ instead of $\{1, 2, \ldots, M\}$ as before for $M \in \mathbb{Z}_+$. Then, we have the following proposition.

Proposition 5. *Suppose that g, h are L_g, L_h-smooth, respectively. Then, it follows that*

$$\mathbb{E}\left[\|\nabla f(x_{k+1})\|_2^2 | \mathcal{F}_k\right] \leq 8(L_g + \mu_k)\delta + 4\sigma_h^2 + 4(\mu_k^2 + L_h^2)\mathbb{E}\left[\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k\right].$$

By combining Proposition 4 and 5, we have the following proposition.

Proposition 6. *Make the same assumption as Proposition 5. Let $\mu_k = O(L_h)$ and $\mu_k = \Omega(L_h)$. Then, it follows that*

$$\mathbb{E}\left[\|\nabla f(x_{k+1})\|_2^2 | \mathcal{F}_k\right] \leq O((L_g + L_h)\delta + \sigma_h^2 + L_h \mathbb{E}\left[f(x_k) - f(x_{k+1}) | \mathcal{F}_k\right]).$$

From Proposition 6, we can obtain the convergence theorem that indicates that as L_h decrease, SPD has better convergence.

Theorem 10. *Make the same assumption as Proposition 6. We assume $L_h = O(L_g)$ and the optimal value f_* of f is bounded from below. Then it follows that*

$$\mathbb{E}[\|\nabla f(x_R)\|_2^2] \leq O\left(L_g\delta + \sigma_h^2 + \frac{L_h(f(x_1) - f_*)}{M}\right).$$

Theorem 10 implies the following complexity result which is better than Corollary 5 because of $L_h = O(L_g)$.

Corollary 6. *Suppose the assumptions in Theorem 10 hold and $\sigma_h^2 = O(\epsilon)$.We set $\delta = O(\epsilon/L_g)$. Then, the complexity M to obtain an ϵ-accurate solution in expectation is $O(L_h/\epsilon)$.*

5.4.3 Polyak-Łojasiewicz Condition

Here, we show a fast convergence of *Double-loop SPD* described in Algorithm 11 under Polyak-Łojasiewicz condition:

Definition 4. *A function ϕ satisfies Polyak-Łojasiewicz condition, i.e., $\exists C > 0$ such that $\forall x \in \mathbb{R}^d$*

$$\phi(x) - \min \phi \leq C\|\nabla\phi(x)\|_2^2. \tag{5.7}$$

Algorithm 11 Double-loop SPD

```
Input: initial point $y_1$, the maximum number of outer-iterations $N$, the options for Algorithm 9 $M, \mathcal{A}, T$
for $t = 1$ to $N - 1$ do
    $y_{t+1} \leftarrow$ Algorithm 9 $(y_t, M, \mathcal{A}, T)$
end for
Return $y_N$
```

Note that Algorithm 9 and 11 are essentially the same up to the returned point. Therefore, algorithm remain unchanged in practical implementations.

Let $\delta = O(\epsilon/L_g), M = O(CL_g/2)$ and assume $\sigma_h^2 = O(\epsilon)$. Using Theorem 9 and (5.7), we can easily show

$$\mathbb{E}[\|\nabla f(y_{t+1})\|_2^2] \leq \epsilon + \frac{\mathbb{E}[\|\nabla f(y_t)\|_2^2]}{2}.$$

This recurrence relation immediately implies $\mathbb{E}[\|\nabla f(y_{t+1})\|_2^2] \leq 2\epsilon + (\frac{1}{2})^t \|\nabla f(y_1)\|_2^2$. This mean that if we run Algorithm 11 for $N = O(\log 1/\epsilon)$ outer-iterations, we can obtain an ϵ-accurate solution. Thus, the following theorem holds.

Theorem 11. *Make the same assumption as Theorem 9 and assume Polyak-Łojasiewicz condition holds. Let δ, M and σ_h be as above. Then, the complexity including that of inner SPD to obtain a solution is $O(CL_g \log \frac{1}{\epsilon})$.*

5.4.4 Total Complexity

We consider the total complexity that includes the complexity of an underlying solver. In recent years, several stochastic optimization methods that can solve the sub-problem *SP(k)* have been developed. Note that the objective function of the sub-problem can be $\mu_k = O(L_g)$ or $O(L_h)$ strongly convex. Let us adopt SGD (Rakhlin et al., 2012) as an underlying solver. Noting that, SGD can solve the sub-problem with a complexity of $O\left(\frac{1}{\mu_k \delta}\right)$, we obtain the total complexities as shown in the middle row of Table 5.1. Although the complexity $O(L_g/\epsilon^2)$ is the same as that of RSG method (Ghadimi and Lan, 2013b) for solving Lipschitz smooth nonconvex problems, SPD has better practical performance for several reasons. Firstly, since we can warm start the underlying solver of SPD at the previous solution, it is enough to perform fewer iterations than suggested by the theory. By the strong convexity of the sub-problem, we get $\|\nabla f(x_k)\|_2^2 \geq 2\mu_k(\phi_k(x_k) - \phi_k^*)$, that is, as current iterate x_k is closer to a stationary point, initial objective gap of each sub-problem *SP(k)* also becomes small. Let us assume μ_k are uniformly upper and lower bounded by positive constants. Noting that smoothnesses $L_g + \mu_k$, strong convexities μ_k, and accuracy δ of sub-problems are the same order among all iterations, we find that as the initial objective gap of a sub-problem is smaller, we can easily solve it empirically. Secondly, although a performance of almost all of stochastic gradient based algorithms are affected by its variance, SPD reduces this effect by fixing an estimate of $\nabla h(x_k)$ in each inner loop.

Moreover, we can show improved convergence complexities by using an additional structure of the convex sub-problems such as a variance growth condition.

Variance Growth Condition

We first introduce the variance growth condition.

Definition 5. *A function ϕ satisfies the variance growth condition if there exist $\alpha, \beta > 0$ such that $\forall x \in \mathbb{R}^d$,*

$$\mathbb{V}[\Phi(x,\xi)] \le \alpha + \beta \|\nabla \phi(x)\|_2^2,$$

where $\Phi(x,\xi)$ denotes a stochastic gradient of ϕ at x.

This condition can be found in Bottou et al. (2016); Carlson et al. (2016) and a stronger condition called gradient growth condition is used in Schmidt and Roux (2013); Gürbüzbalaban et al. (2015). Note that the variance growth condition with $\alpha = 0$ is used in Carlson et al. (2016) to establish a convergence analysis of stochastic optimization method for the learning discrete graphical models including RBMs and this condition is controllable by mini-batching of gradient estimators.

Applying Theorem 4.6 in Bottou et al. (2016) to the sub-problem *SP(k)*, we immediately obtain a complexity to solve it as follows.

Proposition 7. *Let us assume that the objective function ϕ_k of SP(k) satisfies the variance growth condition with $\frac{\alpha}{(1+\beta)\mu_k} \le \delta$. Then, if we run SGD, with a constant learning rate $\eta = O(\frac{1}{L_g(1+\beta)})$, a δ-accurate solution of SP(k) can be obtained with a complexity of*

$$O\left(\frac{L_g(1+\beta)}{\mu_k} \log \frac{\phi_k(x_k) - \phi_k^*}{\delta}\right).$$

Since $\phi_k(x_k) - \phi_k^* \le \frac{1}{2\mu_k}\|\nabla f(x_k)\|_2^2$, if $\{\|\nabla f(x_k)\|_2\}_{k=1}^M$ are uniformly bounded and if we apply Proposition 7 with the same μ_k, δ as in Corollary 5, 6, or Theorem 11, we can find that the variance growth condition strictly improves the total complexities as shown in the last row of Table5.1.

5.5 Boltzmann Machines

Although we are mainly concerned with RBMs or DBMs rather than BMs, we describe an application to learn BMs because BMs are the general form of these models. The BM is a particular type of Markov random field with visible binary stochastic units $v \in \{0,1\}^D$ and hidden binary stochastic units $h \in \{0,1\}^M$. The negative energy of the state $\{v, h\}$ is

$$-E(v,h;\Theta) = v^\top b + h^\top c + v^\top U v + h^\top V h + v^\top W h,$$

where $\Theta = (b, c, U, V, W)$ are the model parameters, i.e., $b \in \mathbb{R}^D, c \in \mathbb{R}^M, U \in \mathbb{R}^{D\times D}, V \in \mathbb{R}^{M\times M}$, and $W \in \mathbb{R}^{D\times M}$. The diagonal elements of U and V are set to zeros. Note that special form of the Boltzmann machine with $U = 0$ and $V = 0$ is nothing else but RBMs. The joint distribution of v, h is defined as proportional to $\exp(-E(v,h;\Theta))$. Thus, the likelihood of BMs is

$$\begin{aligned} p(v|\Theta) &= \frac{1}{Z(\Theta)} \sum_h \exp(-E(v,h;\Theta)), \\ Z(\Theta) &= \sum_v \sum_h \exp(-E(v,h;\Theta)). \end{aligned}$$

Learning the BM is achieved by minimizing the average negative log-likelihood, i.e., for i.i.d. samples $\{v_i\}_{i=1}^N$:

$$\begin{aligned} \min_{\mathbb{R}^d} f(\Theta) &= -\frac{1}{N}\sum_{i=1}^N \log p(v_i|\Theta) = g(\Theta) - h(\Theta), \\ \text{where } g(\Theta) &= \log \sum_v \sum_h \exp(-E(v,h;\Theta)), \\ h(\Theta) &= \frac{1}{N}\sum_{i=1}^N \log \sum_h \exp(-E(v_i,h;\Theta)). \end{aligned}$$

Since a composite function of the convex *log-sum-exp* function and linear mapping is convex, training the BM is DC programming. The gradients of g and h are as follows: for the parameter $\theta \in \Theta$,

$$\begin{aligned} \nabla_\theta g(\Theta) &= -\mathbb{E}_{p(v,h;\Theta)}\left[\nabla_\theta E(v,h;\Theta)\right], \\ \nabla_\theta h(\Theta) &= -\mathbb{E}_{p(h|v;\Theta)p_0(v)}\left[\nabla_\theta E(v,h;\Theta)\right], \end{aligned}$$

where $p_0(v)$ is the empirical distribution $\frac{1}{N}\sum_{i=1}^n \delta(v = v_i)$. To run SPD with a diagonal Hessian approximation, we give $\operatorname{diag}(\nabla_\theta^2 h(\Theta))$ based on the formulation:

$$\nabla_{\theta_i}^2 h(\Theta) = \nabla_{\theta_i} h(\Theta) - (\nabla_{\theta_i} h(\Theta))^2,$$

whose derivation can be found in the supplement.

Although for RBMs the second expectation $\nabla_\theta h(\Theta)$ is tractable, the first $\nabla_\theta g(\Theta)$ is not because the expectation is taken with respect to v and h. Practically, contrastive divergence (CD) (Hinton, 2002) or persistent contrastive divergence (PCD) (Tieleman and Hinton, 2009) is used to obtain a stochastic approximation of $\nabla_\theta g(\Theta)$. Therefore, we can apply SPD with $\sigma_h^2 = 0$ to training RBMs. Note that, in each iteration of SPD, $\nabla h(\Theta)$ is computed at the cost of N evaluations; however, in practice, this cost is relatively small compared to that of CD / PCD used in an underlying solver. In fact, previous work (Salakhutdinov and Murray, 2008) has shown that to obtain a good approximation of the gradient, a sufficiently large number of Gibbs samples are required in the CD method.

It is intractable to compute both expectations $\nabla_\theta g(\Theta)$ and $\nabla_\theta h(\Theta)$ for general BMs. Therefore, we stochastically approximate these terms. We use persistent Gibbs sampling (Salakhutdinov and Hinton, 2009) to compute the model expectation term $\nabla_\theta g(\Theta)$. That is, we obtain new samples v, h in underlying solver by Gibbs sampling with few steps, initialized at the previous samples. This procedure is equivalent to PCD for RBMs. To estimate the data expectation $\nabla_\theta h(\Theta)$, we adopt self-normalized importance sampling using mean-field approximation: $q(h|\mu) = \prod_{j=1}^M q(h^j)$, with $q(h^j = 1) = \mu^j$, to the true distribution $p(h|v, \Theta)$. First, we perform following fixed-point iterations until convergence and obtain $q(h|\mu)$, where $\mu = (\mu^1, \dots, \mu^M)$, as done in Salakhutdinov and Hinton (2009),

$$\mu^j \leftarrow \sigma\left(c^j + \sum_i W_{ij} v^i + \sum_k J_{kj}\mu^k\right),$$

where σ is the sigmoid function. Next we draw samples $\{h_s\}_{s=1}^P$ from $q(h|\mu)$ and approximate $\nabla_\theta h(\Theta)$ as follows:

$$\nabla_\theta h(\Theta) \sim \mathbb{E}_{p_0(v)}\left[\frac{\sum_{s=1}^P -\nabla_\theta E(v, h_s|\Theta)\cdot\omega_s}{\sum_{s=1}^P \omega_s}\right],$$

where $\omega_s = \frac{\exp(-E(v,h_s;\Theta))}{q(h_s|\mu)}$ is a ratio between joint distribution and $q(h)$, so that it is computable. This estimate is asymptotically consistent (Owen, 2013). Using these approximations, we can run SPD for learning BMs. For simplicity of implementation, we may use $P = 1$ and the expectation μ instead of a sample h^1 to reduce the sampling variance, even though it may have a relatively large bias, and so the resulting approximation of ∇h is the same as the mean-field approximation.

5.5.1 SPD as The Extension of EM/MCEM Algorithms

In the following we describe the connection between EM, MCEM algorithms and SPD. Let Θ' be a current parameter of a BM, $V = \{v_i\}_{i=1}^N$, and $H = \{h_i\}_{i=1}^N$ be i.i.d data samples and corresponding hidden variables, respectively. At the E-step of the EM algorithm we computes the following expectation of the log-likelihood of the joint distribution:

$$Q(\Theta, \Theta') = \frac{1}{N}\int p(H|V, \Theta')\log p(V, H|\Theta)dH$$

$$= \mathbb{E}_{p_0(v)p(h|v,\Theta')}[-E(v,h;\Theta)] - \log Z.$$

In MCEM, the first term of the right hand side is approximated by a Monte Carlo method. This term is a linear mapping with respect to $\forall\theta \in \Theta$ and its gradient is nothing else but $\nabla_\theta h(\Theta')$. Combining the fact $g(\Theta) = \log Z$, we conclude that $Q(\Theta, \Theta')$ is the objective function of the sub-problem of SPD with $\mu = 0$ and the M-step in EM/MCEM corresponds to solving this sub-problem. Therefore, SPD can be recognized as an extension of the EM/MCEM algorithm for training BMs with better convergence analyses. Note that the proximal term of SPD with L_g or L_h convexity does not affect the convergence rate by Theorem 9 and 10, while it facilitates the optimization of the sub-problems by its strong convexity. Thus, SPD may be the more efficient method than EM/MCEM algorithms.

5.6 Numerical Experiments

In this section, we demonstrate the effectiveness of AdaSPD on training RBMs and DBMs with the weight decay. Our implementation is done using Theano (Bergstra et al., 2010; Bastien et al., 2012). We used the binarized MNIST (Salakhutdinov and Murray, 2008) which has 60,000 training and 10,000 test images (28$\times$28 pixels) of 10 handwritten digits (0-9) and used CalTech101 Silhouettes (Marlin et al., 2010) which has 6,364 training and 2,307 test images (28$\times$28 pixels) of 101 classes, representing object silhouettes.

Since computing the partition function of BMs is difficult (except for small RBMs), we used the annealed importance sampling (AIS) (Salakhutdinov and Murray, 2008) to estimate it with the settings: (i) for RBM, 500 temperatures spaced uniformly from 0 to 0.5, 4,000 temperatures spaced uniformly from 0.5 to 0.9, 10,000 temperatures spaced uniformly from 0.9 to 1, and 100 particles, (ii) for DBM, 20,000 temperatures spaced uniformly, and 1,000 particles. Theoretically, AdaSPD uses a random iteration count R to establish complexity results to solve problems; however, we always evaluate the model at the current iteration. The number of underlying solver iterations T and the suffix averaging parameter α were set as follows: $T = \lceil N/b \rceil, \alpha T = \lceil T/2 \rceil$, where N is the number of data points and b is a mini-batch size. All parameter settings of AdaSPD used in experiments can be found in the supplement.

5.6.1 Restricted Boltzmann Machines

We compare AdaSPD to SGD and AdaGrad on RBMs with 15, 25, and 500 hidden units. For metric H_k of AdaSPD, we tested the diagonal Hessian approximation and scalar matrices with $\mu \in \{10^{-1}, 10^{-3}, 10^{-5}\}$.

The results are shown in Figure 5.1. The top row represents the result for binarized MNIST dataset and the bottom row represents the result for CalTech101 Silhouettes. The vertical axis is the average (estimated) log-likelihood on training dataset. As can be seen

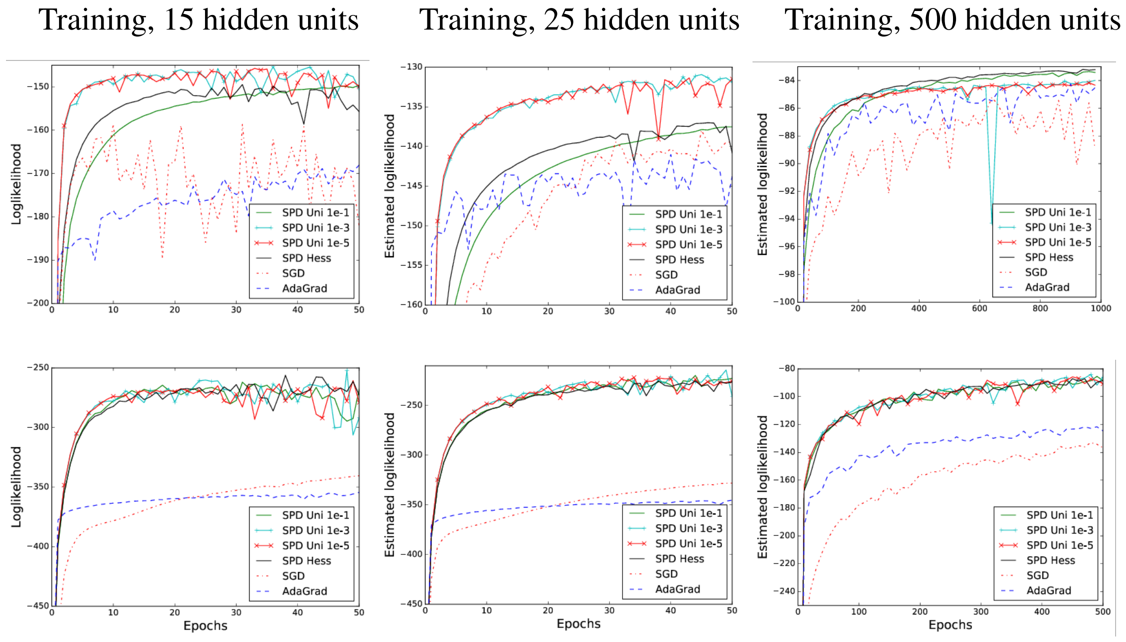

Figure 5.1: Comparison of algorithms on training RBMs with 15, 25, and 500 hidden units. The vertical axis is the average (estimated) log-likelihood on training dataset. Top row: MNIST, Bottom row: CalTech101 Silhouettes.

in the figure, AdaSPDs showed significantly fast convergence compared to the others, although it tends to over-fitting, especially for the 500-hidden RBM on CalTech101 Silhouettes dataset. For binarized MNIST, the best training log-likelihood of the 500-hidden RBM was -83.19 and test log-likelihood was -85.83 obtained by AdaSPD with the diagonal Hessian approximation. These results are comparable to those reported in Carlson et al. (2015); Salakhutdinov and Murray (2008). For CalTech101 Silhouettes, the best training log-likelihood of the 500-hidden RBM was -84.15 obtained by AdaSPD with the scalar matrix ($\mu = 10^{-3}$) and test log-likelihood was -109.95 obtained by AdaSPD with the scalar matrix ($\mu = 10^{-1}$).

5.6.2 Deep Boltzmann Machines

Next, we train two DBMs: one has three-hidden layers (500-500-1000 hidden units) and the other has four-hidden layers (500-500-500-1000 hidden units). The results are shown in Table 5.2. Stochastic approximation procedure (SAP) (Salakhutdinov and Hinton, 2009) is the standard method for training DBMs. We compare our method with SAP with and without pre-training schemes (Cho et al., 2013; Salakhutdinov and

Table 5.2: Comparison of estimated variational lower bound on the log-likelihood of MNIST dataset.

Algorithms	3-hidden layers DBM		4-hidden layers DBM	
	Train	Test	Train	Test
AdaSPD	-82.28	-85.17	-82.85	-85.15
SAP	-	-128.72	-	-128.70
Two-stage pre-training+SAP	-	-81.84	-	-83.25
Pre-Training+SAP	-84.49	-85.10	-	-

Hinton, 2009) We should point out that AdaSPD and SAP were run without any sophisticated pre-training such as Salakhutdinov and Hinton (2009); Cho et al. (2013). Although AdaSPD showed a little bit worse score than the method using the best pre-training strategy (Cho et al., 2013), it outperformed SAP and was comparable to or better than the other pre-training methods proposed in (Salakhutdinov and Hinton, 2009; Cho et al., 2013). Thus, our experiments showed the possibility that no pre-training methods can lead to a better BM model. Figure 5.2 shows the learning curves for AdaSPD. From this figure we observe efficiency of our method on DBMs.

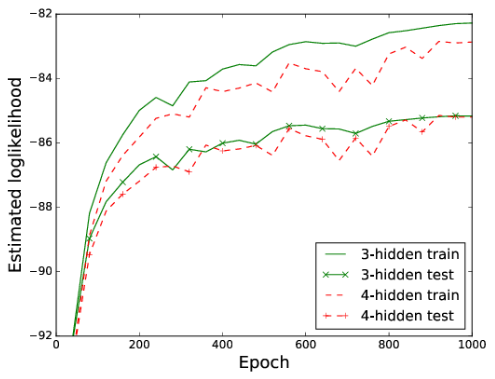

Figure 5.2: Learning curves for AdaSPD on DBMs.

5.7 Appendix

5.7.1 Proofs

We give proofs of convergence analyses. We first prove the Proposition 4.

Proof of Proposition 4. Since ϕ_k is $(L_g + \mu_k)$-smooth function, we have

$$\phi_k(x) \leq \phi_k(x_k) + \langle \nabla \phi_k(x_k), x - x_k \rangle + \frac{L_g + \mu_k}{2} \|x - x_k\|_2^2.$$

By minimizing both sides of the above inequality,

$$\phi_k^* \leq \phi_k(x_k) - \frac{1}{2(L_g + \mu_k)} \|\nabla \phi_k(x_k)\|_2^2.$$

Noting that $\phi_k(x_k) = f(x_k)$ and $\mathbb{E}_{v_h(x_k)}[\|\nabla \phi_k(x_k)\|_2^2 | \mathcal{F}_k] \geq \|\nabla f(x_k)\|_2^2$, we have

$$\mathbb{E}_{v_h(x_k)}[\phi_k^* | \mathcal{F}_k] \leq f(x_k) - \frac{1}{2(L_g + \mu_k)} \|\nabla f(x_k)\|_2^2.$$

Using $\mathbb{E}[\phi_k(x_{k+1}) | \mathcal{F}_k] \leq \phi_k^* + \delta$ and the above inequality, we have

$$\mathbb{E}[\phi_k(x_{k+1}) | \mathcal{F}_k] \leq \delta + f(x_k) - \frac{1}{2(L_g + \mu_k)} \|\nabla f(x_k)\|_2^2.$$

Thus, it follows that

$$\begin{aligned}
&\mathbb{E}[f(x_{k+1}) + \frac{\mu_k}{2} \|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k] \\
&\quad \leq \mathbb{E}[g(x_{k+1}) - (h(x_k) + \langle \nabla h(x_k), x_{k+1} - x_k \rangle) + \frac{\mu_k}{2} \|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k] \\
&\quad = \mathbb{E}[\phi_k(x_{k+1}) - \langle \nabla h(x_k) - v_h(x_k), x_{k+1} - x_k \rangle | \mathcal{F}_k] \\
&\quad \leq \mathbb{E}[\phi_k(x_{k+1}) | \mathcal{F}_k] + \mathbb{E}[\frac{1}{\mu_k} \|\nabla h(x_k) - v_h(x_k)\|_2^2 + \frac{\mu_k}{4} \|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k] \\
&\quad \leq \delta + f(x_k) - \frac{1}{2(L_g + \mu_k)} \|\nabla f(x_k)\|_2^2 + \frac{\sigma_h^2}{\mu_k} + \frac{\mu_k}{4} \mathbb{E}[\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k],
\end{aligned}$$

where for the first inequality we used convexity of h and for the second inequality we used Young's inequality. This finishes the proof of Proposition 4. □

Next, let us prove Theorem 9.

Proof of Theorem 9. Summing up the inequality of Proposition 4 over indices $k = 1, \dots, M$ and taking the expectation, we have

$$\sum_{k=1}^{M} \frac{1}{2(L_g+\mu_k)} \mathbb{E}[\|\nabla f(x_k)\|_2^2] \le M\delta + \sum_{k=1}^{M} \frac{\sigma_h^2}{\mu_k} + \mathbb{E}[f(x_1) - f(x_{M+1})].$$

Since $\mu_k = O(L_g) \wedge (\mu_k = \Omega(L_g) \vee \sigma_h = 0)$ and $f(x_1) - f(x_{M+1}) \le f(x_1) - f_*$,

$$\sum_{k=1}^{M} \mathbb{E}[\|\nabla f(x_k)\|_2^2] \le O(L_g M\delta + M\sigma_h^2 + L_g(f(x_1) - f_*)).$$

Noting that

$$\mathbb{E}[\|\nabla f(x_R)\|_2^2 | \mathcal{F}_M] = \frac{1}{M} \sum_{k=1}^{M} \|\nabla f(x_k)\|_2^2,$$

we can conclude the proof of Theorem as follows,

$$\begin{aligned}\mathbb{E}[\|\nabla f(x_R)\|_2^2] = \mathbb{E}\left[\mathbb{E}[\|\nabla f(x_R)\|_2^2 | \mathcal{F}_M]\right] &= \frac{1}{M}\sum_{k=1}^{M} \mathbb{E}[\|\nabla f(x_k)\|_2^2] \\ &\le O\left(L_g\delta + \sigma_h^2 + \frac{L_g(f(x_1)-f_*)}{M}\right).\end{aligned}$$

□

Below is the proof of Proposition 5.

Proof of Proposition 5. It follows that

$$\begin{aligned}&\mathbb{E}[\|\nabla f(x_{k+1})\|_2^2 | \mathcal{F}_k] \\ &\quad = \mathbb{E}[\|\nabla \phi_k(x_{k+1}) - (\nabla h(x_k) - v_h(x_k)) + \nabla h(x_k) - \nabla h(x_{k+1}) - \mu_k(x_{k+1} - x_k)\|_2^2 | \mathcal{F}_k] \\ &\quad \le 4\mathbb{E}[\|\nabla \phi_k(x_{k+1})\|_2^2 + \|\nabla h(x_k) - v_h(x_k)\|_2^2 + \|\nabla h(x_k) - \nabla h(x_{k+1})\|_2^2 + \mu_k^2\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k] \\ &\quad \le 4\sigma_h^2 + 4\mathbb{E}[\|\nabla \phi_k(x_{k+1})\|_2^2 + (\mu_k^2 + L_h^2)\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k],\end{aligned}$$

where for the first inequality we used $\|\sum_{j=1}^d \alpha_j\|_2^2 \le d\sum_{j=1}^d \|\alpha_j\|_2^2$ and for the second inequality we used Lipschitz smoothness of h. Since ϕ_k is $(L_g + \mu_k)$-smooth,

$$\frac{1}{2(L_g+\mu_k)} \mathbb{E}[\|\nabla \phi_k(x_{k+1})\|_2^2 | \mathcal{F}_k] \le \mathbb{E}[\phi_k(x_{k+1}) - \phi_k^* | \mathcal{F}_k] \le \delta.$$

Thus, we conclude

$$\mathbb{E}[\|\nabla f(x_{k+1})\|_2^2 | \mathcal{F}_k] \le 4\sigma_h^2 + 8(L_g+\mu_k)\delta + 4(\mu_k^2 + L_h^2)\mathbb{E}[\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k].$$

□

By combining Proposition 4 and 5, we prove Proposition 6.

Proof of Proposition 6. Noting that $\mu_k = O(L_h) \wedge \mu_k = \Omega(L_h)$, we have

$$\begin{aligned}\mathbb{E}[\|\nabla f(x_{k+1})\|_2^2|\mathcal{F}_k] &\leq O\left((L_g + L_h)\,\delta + \sigma_h^2 + L_h^2\mathbb{E}[\|x_{k+1} - x_k\|_2^2|\mathcal{F}_k]\right)\\ &\leq O\left((L_g + L_h)\,\delta + \sigma_h^2 + L_h\mathbb{E}[f(x_k) - f(x_{k+1})|\mathcal{F}_k]\right),\end{aligned}$$

where for the first and second inequality we used Proposition 4 and 5, respectively. □

We give the proof of Theorem 10.

Proof of Theorem 10. Using Proposition 6 and $L_h = O(L_g)$, it follows that

$$\mathbb{E}[\|\nabla f(x_{k+1})\|_2^2|\mathcal{F}_k] \leq O\left(L_g\delta + \sigma_h^2 + L_h\mathbb{E}[f(x_k) - f(x_{k+1})|\mathcal{F}_k]\right).$$

This inequality resemble Proposition 4 up to the term $\mathbb{E}[\|x_{k+1} - x_k\|_2^2|\mathcal{F}_k]$, so that we can show the theorem in the same manner as Theorem 9. □

5.7.2 The derivation of diagonal hessian approximation

To run AdaSPD with a diagonal hessian approximation for training BMs, we give $\mathrm{diag}(\nabla_\theta^2 h(\Theta))$, where h is the concave part of the log-likelihood of BMs. We only consider a parameter W_{ij} connecting a visible unit v^i and a hidden unit h^j because for the other parameters it can be shown in the same manner.

$$\begin{aligned}\nabla_{W_{ij}}^2 h(\Theta) &= \nabla_{W_{ij}}^2 \log\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h};\Theta))\\ &= \nabla_{W_{ij}}\left(\frac{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))v^ih^j}{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))}\right)\\ &= \frac{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))(v^ih^j)^2}{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))} - \left(\frac{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))v^ih^j}{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))}\right)^2\\ &= \frac{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))v^ih^j}{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))} - \left(\frac{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))v^ih^j}{\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))}\right)^2\\ &= \nabla_{W_{ij}}\log\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta)) - \left(\nabla_{W_{ij}}\log\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\Theta))\right)^2\\ &= \nabla_{W_{ij}}h(\Theta) - (\nabla_{W_{ij}}h(\Theta))^2,\end{aligned}$$

where we used $(v^ih^j)^2 = v^ih^j$ derived from the fact that v^i and h^j are binary units $\{0, 1\}$.

5.7.3 Parameter settings for training RBMs and DBMs

In our experiments, we optimized L_2-penalized log-likelihoods of RBMs and DBMs. Here, we give all parameter settings used for AdaSPD. The damping parameter λ was fixed to 10^{-4}. The scales of metrics were set to $\mu = 10^{-4}$ for diagonal hessian approximation and $\mu \in \{10^{-5}, 10^{-3}, 10^{-1}\}$ for scalar matrix. The number of underlying solver iterations T and the suffix averaging parameter α were set as follows: $T = \lceil N/b \rceil, \alpha T = \lceil T/2 \rceil$, where N is the number of data points and b is a mini-batch size. The other parameters are listed in Table 5.3 for binarized MNIST dataset and Table 5.4 for CalTech101 Silhouettes.

Table 5.3: Parameter settings for binarized MNIST

Model	Minibatch b	PCD-k	Mean-field iter.	L_2-penalty	η
RBM-15	32	1	-	0	10^{-1}
RBM-25	32	3	-	0	10^{-1}
RBM-500	128	10	-	5×10^{-4}	10^{-1}
DBM-500-500-1000	128	10	10	3×10^{-4}	10^{-2}
DBM-500-500-500-1000	128	10	10	5×10^{-4}	10^{-2}

Table 5.4: Parameter settings for CalTech101 Silhouettes

Model	Minibatch-size b	PCD-k	Mean-field iter.	L_2-penalty	η
RBM-15	32	1	-	0	10^{-2}
RBM-25	32	3	-	0	10^{-2}
RBM-500	64	10	-	10^{-3}	10^{-2}

Part II

New Machine Learning Methods using Functional Gradient

Chapter 6

Functional Gradient Descent

Most optimization methods on Euclidean spaces can be generalized to those on Hilbert spaces. For instance, functional gradient methods (Kivinen et al., 2004; Dai et al., 2014; Zhu and Xu, 2015; Dieuleveut and Bach, 2016) in Hilbert spaces are the most straightforward extensions. When a given functional class approximates the true function well and it can be modeled efficiently by fewer parameters, parametric models are quite useful in providing better results. One of the typical successful model is the convolutional neural network commonly used in computer vision tasks. In contrast, when the task is rather complicated and we do not have enough information about it, finding such a good parameterization is difficult due to the limitation of computational resources. In this case, an infinite-dimensional modeling with a sophisticated regularization is reasonable to solve the problem efficiently. Kernel method (Smola and Schölkopf, 1998) is an example of such a method, which is formalized as L_2-regularized empirical risk minimization problems in a reproducing kernel Hilbert space consisted of functions. There are various methods for solving this problem and many functional gradient based methods (Kivinen et al., 2004; Dai et al., 2014) in a reproducing kernel Hilbert space were also proposed. On the other hand, in boosting methods (Freund and Schapire, 1997; Schapire et al., 1998; Friedman, 2001; Mason et al., 1999; Schapire and Freund, 2012), complicated functions are constructed as large as necessary from a simple function by adding a weak learner at each iteration and generalization ability of an obtained function is commonly guaranteed by the early stopping. We note both approaches are none other than optimization methods in functional spaces.

The methods proposed in subsequent chapters are similar to boosting methods. Those are constructed based on the observation that several machine learning tasks can be regarded as optimization problems of probability measures. In this chapter, we first explain that such a problem can be (approximately) solved through transport maps using the simplest form of the problem. We next introduce functional gradient methods for the optimization problem of transport maps and explain convergence property of the method. Finally, we explain the connection between residual networks (He et al., 2016) which is the state-of-the-art model in computer vision and the functional gradient method, and explain how powerful compared to methods on finite-dimensional spaces, intuitively and theoretically. Recently, this connection is considered to be prominent for explaining the reason of superior performance of residual networks. For instance, utilizing this connection, Bartlett et al. (2018) showed the global convergence property of functional gradient method for training residual networks for regression tasks under suitable assumptions. In Huang et al. (2017a), a boosting method for learning residual networks, which iteratively trains residual-layer, was proposed. This method is also similar to our method introduced in Chapter 9. Another interesting aspect of functional gradient methods is Wasserstein geometry view which will be explained in Chapter 8. The methods proposed in Chapter 7 and 8 for ensemble learning and generative model learning are closely related to this viewpoint. We note that very similar methods (Chizat and Bach, 2018; Johnson and Zhang, 2018) to ours based on the same viewpoints have been proposed with good theoretical analyses.

6.1 Problem Setting

In this section, we provide several notations and describe the simplest form of problems considered in Part II. Let $\mathcal{X} = \mathbb{R}^d$ be a space of interest. We denote by μ a general Borel probability measure on $\mathcal{X}$ which may have continuous, discrete, or finite support. we also denote by $\mathbb{E}_\mu$ the expectation with respect to a random variable according to μ, by $L_2(\mu)$ the space of square-integrable functions with respect to μ, and by $L_2^q(\mu)$ the product space of $L_2(\mu)$ equipped with $\langle \cdot, \cdot \rangle_{L_2^q(\mu)}$-inner product: for $\forall \xi, \forall \zeta \in L_2^q(\mu)$,

$$\langle \xi, \zeta \rangle_{L_2^q(\mu)} \stackrel{\text{def}}{=} \mathbb{E}_\mu[\xi(X)^\top \zeta(X)] = \mathbb{E}_\mu\left[\sum_{j=1}^{q} \xi_j(X)\zeta_j(X)\right].$$

We also use the following norm: for $\forall \xi \in L_2^q(\mu)$, $\|\xi\|_{L_1^q(\mu)} \stackrel{\text{def}}{=} \mathbb{E}_\mu[\|\xi(X)\|_2] = \mathbb{E}_\mu\left[\sqrt{\sum_{j=1}^{q} \xi_j^2(X)}\right]$. We note that the notation of q in $L_1^q(\mu)$ may be omitted if it is not needed.

As seen in subsequent chapters later, several problems in machine learning can be formalized as optimization problems with respect to probability measure. We note that when a base probability measure μ_0 is given, this problem is (approximately) solved by optimizing transport maps in $L_2(\mu_0)$ as explained below. In general, for a probability measure μ, the push-forward measure $\phi_\sharp\mu$ by a map $\phi \in L_2(\mu)$ is defined as follows: for a Borel measurable set $\mathcal{A} \subset \Theta$,

$$\phi_\sharp\mu(\mathcal{A}) \stackrel{\text{def}}{=} \mu(\phi^{-1}(\mathcal{A})). \tag{6.1}$$

For a probability measure μ_q having a continuous density function q, i.e., $d\mu_q = q(\theta)d\theta$, the push-forward measure is described as follows: by the change of variables formula

$$d\phi_\sharp\mu_q(\theta) = q(\phi^{-1}(\theta))|\det \nabla\phi(\theta)^{-1}|d\theta.$$

When we use maps in $L_2(\mu)$ to push-forward probability measures, we call these as transport maps. We notice that a vast space of probability measures may be reached through transport maps from a given base probability measure. Such a property is investigated well in the optimal transport theory, see Villani (2008); Ambrosio et al. (2008).

From the above discussion, considering the optimization problems with respect to transport maps is justified. Especially, we treat the following simple problem to explain basic form of functional gradient method and its property.

$$\min_{\phi\in L_2^d(\mu)} \left\{ \mathcal{L}(\phi) \overset{\mathrm{def}}{=} \mathbb{E}_\mu[r(\phi(X))] \right\}, \tag{6.2}$$

where r is a sufficiently smooth function. Note that problems considered in subsequent chapters are not interpreted as this simplified problem correctly, but it is useful in explaining a property and an advantage of the method and leads to a deeper understanding.

Connections to concrete problems. We here briefly introduce the formulation of the problems treated in subsequent chapters. In Chapter 7, we propose an ensemble method for classification tasks. Since an ensemble is composed of weak (base) learners by taking an expectation on the space of them, the problem turns out to be equivalent to optimizing probability measures to obtain a highly accurate classifier. Thus, the discussion in this chapter is quite useful for understanding the method for ensemble learning problems. In Chapter 8, we consider the generative adversarial networks (GANs) (Goodfellow et al., 2014) which is the most successful model for generating realistic images. In the training procedure of GANs, the discriminator and the generator are simultaneously trained in an adversarial way. Namely, the discriminator is trained to classify between training images and generated images drawn from the generator, while the generator is trained to generate realistic images from a noise distribution to mimic real images. Thus, GANs training can be formalized as min-max problems. Under the assumption where each discriminator is completely optimized in the procedure, the problems can be recognized as minimizing the objective with respect to generator to increase the classification error of a current discriminator. Noting that generators are none other than transport maps, the GAN training is also accomplished by optimizing transport maps. Moreover, we give an interesting viewpoint that the functional gradient method is a discretization procedure of a gradient flow in the space of probability measures in terms of Wasserstein geometry. In Chapter 9, we propose a new gradient boosting method for classification tasks. In our method, a classifier consisting of the feature extraction and the linear classifier grows by greedily adding a residual-layer on the top of the feature extraction based on functional gradient boosting theory. While GANs training is the min-max problem, this problem is the min-min problem with respect to the linear classifier and the feature extraction. By regarding feature extractions as transport maps from input data distribution in order to obtain a ditribution in which each example can be linearly separated into corresponding classes, we notice that the proposed method turns out to be the method for optimizing transport maps when a linear classifiers is completely optimized in each iteration. Thus, this problem is also basically special case of the problem treated in this chapter.

6.2 Functional Gradient Descent

In this section, we introduce functional gradient descent to solve (6.2). The key notion used in the method is the functional gradient in function spaces. Since they are taken in some function spaces, we first introduce Fréchet differential in general Hilbert spaces.

Definition 6. *Let $(\mathcal{H}, \langle , \rangle_{\mathcal{H}})$ be a Hilbert space and h be a function on $\mathcal{H}$. For $\xi \in \mathcal{H}$, we call that h is Fréchet differentiable at ξ in $\mathcal{H}$ when there exists an element $\nabla_\xi h(\xi) \in \mathcal{H}$ such that*

$$h(\zeta) = h(\xi) + \langle \nabla_\xi h(\xi), \zeta - \xi \rangle_{\mathcal{H}} + o(\|\xi - \zeta\|_{\mathcal{H}}).$$

Moreover, for simplicity, we call $\nabla_\xi h(\xi)$ Fréchet differential or functional gradient.

If the objective $\mathcal{L}(\phi)$ is differential, then we can obtain $\nabla \mathcal{L}(\phi) = \nabla_z r \circ \phi$, where z stands for the input variable of r. Therefore, a functional gradient is a vector field of gradients $\nabla_z r$ at $\phi(x)$ ($x \in \mathcal{X}$). A functional gradient descent is an iterative optimization method using this functional gradient. The whole procedure is described in Algorithm 12.

Algorithm 12 Functional Gradient Descent

```
Input: the initial transport map $\phi_0 \in L_2^d(\mu_0)$ and the learning rate $\eta$.
for $k = 0$ to $T - 1$ do
    $\phi_{k+1} \leftarrow \phi_k - \eta \nabla \mathcal{L}(\phi_k)$
end for
Return the function: $\phi_T$.
```

That is, the functional gradient is the method to move particles $\{\phi_k(x)\}_{x \in \mathcal{X}}$ along negative gradients $\{\nabla \mathcal{L}(\phi_k(x))\}_{x \in \mathcal{X}}$ of decreasing the objective.

As in the finite-dimensional case, smoothness property of objective function is also useful in infinite-dimensional problems, hence, we here introduce Lipschitz smoothness in Hilbert spaces.

Definition 7. *Let h be a function on a Hilbert space $(\mathcal{H}, \langle , \rangle_{\mathcal{H}})$. We call that h is L-Lipschitz smooth if h is differentiable and it follows that $\forall \xi, \zeta \in \mathcal{H}$.*

$$\|\nabla h(\xi) - \nabla h(\zeta)\|_{\mathcal{H}} \leq L \|\xi - \zeta\|_{\mathcal{H}}.$$

Under the smoothness assumption, we can show the convergence to a stationary point by naturally extending the proof of Theorem 1.

Theorem 12. *Let us assume that $\mathcal{L}$ is L-Lipschitz smooth. Consider Algorithm 12 with constant learning rate $\eta \leq 1/L$. Then we have for $T \in \mathbb{Z}_+$*

$$\min_{k \in \{0,\ldots,T-1\}} \|\nabla_\phi \mathcal{L}(\phi_k)\|^2_{L^2(\mu_0)} \leq \frac{2}{\eta T}(\mathcal{L}(\phi_0) - \mathcal{L}_*),$$

where $\mathcal{L}_* = \inf_\phi \mathcal{L}(\phi)$.

Since the proof of the theorem is essentially included in subsequent chapters, though their have specific forms according to the problem and require additional assumption, we omit it here. Note that the convergence rate $O(1/T)$ is the same as the gradient descent method for smooth objective in the finite-dimensional one. This means that even though the optimization is executed in the infinite-dimensional space, we do not suffer from the infinite dimensionality in terms of the convergence.

As explained in the previous section, concrete examples introduced in subsequent chapters are also formalized as optimization problems of transport maps. Thus, the proposed methods for these problems and theoretical analyses are given based on the above idea although we need several adaptations depending on problems.

6.3 Powerful Optimization Ability and Connection to Residual Networks

In this section, we explain that functional gradient methods exhibit an excellent performance for optimizing $\mathcal{L}$ compared to the gradient method in a finite-dimensional space. If $\mathcal{L}$ is Fréchet differentiable, the functional gradient is represented as $\nabla_\phi \mathcal{L}(\phi)(\cdot) = \nabla_z r(\phi(\cdot))$ as shown in the previous section. Therefore, the negative functional gradient indicates the direction of decreasing the objective r at each point $\phi(x)$. An iteration of the functional gradient method with a learning rate η is described as

$$\phi_{t+1} \leftarrow \phi_t - \eta \nabla_z r \circ \phi_t = (id - \eta \nabla_z r) \circ \phi_t.$$

We can immediately notice that this iterate makes ϕ_t one level deeper by stacking a residual network-type layer $id - \eta \nabla_z r$ (He et al., 2016), and data representation is refined through this layer. Starting from a simple feature extraction ϕ_0 and running the functional gradient method for T-iterations, we finally obtain a residual network:

$$\phi_T = (id - \eta \nabla_z r) \circ \cdots \circ (id - \eta \nabla_z r) \circ \phi_0.$$

Therefore, feature extraction ϕ gradually grows by using the functional gradient method to optimize $\mathcal{L}$. This is a huge advantage of the functional gradient method because stationary points in $L_2^d(\mu_0)$ are potentially significant better than those of finite-dimensional spaces. Note that this formulation explains the optimization view (Jastrzebski et al., 2017) of ResNet mathematically.

We now briefly explain how powerful the functional gradient method is compared to the gradient method in a finite-dimensional space, for optimizing $\mathcal{L}$. Let us consider any parameterization of $\phi_t \in L_2^d(\mu_0)$. That is, we assume that ϕ_t is contained in a family of parameterized feature extractions $\mathcal{M} = \{g_\theta : \mathcal{X} \to \mathcal{X} \mid \theta \in \Theta \subset \mathbb{R}^m\} \subset L_2^d(\mu_0)$, i.e., $\exists \theta' \in \Theta \; s.t. \; \phi_t = g_{\theta'}$. Typically, the family $\mathcal{M}$ is given by neural networks. If $\mathcal{L}(g_\theta)$ is differentiable at θ', we get $\nabla_\theta \mathcal{L}(g_\theta)|_{\theta=\theta'} = \langle \nabla_\phi \mathcal{L}(\phi_t), \nabla_\theta g|_{\theta=\theta'} \rangle_{L_2^d(\mu_0)}$ according to the chain rule of derivatives. Note that $\nabla_\phi \mathcal{L}(\phi_t)$ dominates the norm of gradients. Namely, if ϕ_t is a stationary point in $L_2^d(\mu_0)$, ϕ_t is also a stationary point in $\mathcal{M}$ and there is no room for improvement using gradient-based methods. This result holds for any family $\mathcal{M}$, but the inverse relation does not always hold. This means that gradient-based methods may fail to optimize $\mathcal{L}$ in the function space, while the functional gradient method exceeds such a limit by making a feature extraction ϕ_t deeper. For detailed descriptions and related work in this line, we refer to Ambrosio et al. (2008); Daneri and Savaré (2010) and subsequent chapters. As for the representational ability of residual networks, we also refer the reader to Bartlett et al. (2018) which showed the global convergence property of functional gradient methods for regression problems under suitable conditions.

Chapter 7

Stochastic Particle Gradient Descent for Infinite Ensembles

We propose a new method for ensemble learning problems with the L_1-regularization in an infinite-dimensional space of base classifiers; in other words, optimization problems of probability measures. Several optimization methods for this problem have been proposed due to better statistical performance of this problem. Most of these methods adopt the boosting strategy, namely adding base classifiers iteratively in a greedy fashion because of the difficulty of handling L_1-regularization in an infinite-dimensional space. As a result, it causes the requirement of solving nonconvex optimization problems which may make it difficult to apply to large scale models. In contrast, our proposed method adopts an easily executable sampling strategy for approximating a probability measure, which optimizes a transport map for sampling base classifiers, and we are free from adjusting early stopping time. In the theoretical analysis, we show the convergence property to a stationary point with a convergence rate and an *interior optimality property* of the method, that is, the optimization proceeds as long as better probability measures than a current probability measure exist in its support.

This chapter is based on the work *Stochastic Particle Gradient Descent for Infinite Ensembles*, A. Nitanda and T. Suzuki, preprint, 2017 (Nitanda and Suzuki, 2017b).

7.1 Overview

The goal of the binary classification problem is to find a measurable function, called a classifier, from the feature space to the range $[-1, 1]$, which is required to minimize the expected classification error. The ensemble methods, including boosting and bagging, are powerful scheme for solving this problem; constructing a complex classifier by combining base classifiers. It is well-known empirically that such a classifier attains good generalization performance in experiments and applications (Bauer and Kohavi (1999);

Dietterich (2000); Viola and Jones (2001)).

Especially boosting methods have been thoroughly analyzed due to their excellent performance. The first important result was presented by Schapire et al. (1998), where the margin theory for convex combinations of classifiers was introduced. The tighter generalization bounds were given in Koltchinskii and Panchenko (2002); Bartlett et al. (2006) by using the complexities of the function class such as the covering numbers and Rademacher complexity. Moreover, several studies have shown the convergence property and consistency of boosting methods (Mason et al. (2000); Zhang (2003); Mannor et al. (2003); Blanchard et al. (2003); Lugosi and Vayatis (2004); Zhang and Yu (2005); Bartlett and Traskin (2007)). At the same time, many boosting-type methods have been proposed, e.g., AdaBoost (Freund and Schapire (1997)), LogitBoost (Friedman et al. (2000)), Arc-gv (Breiman (1999)), AdaBoost$^*_\nu$ (Rätsch and Warmuth (2005)), α-Boost (Friedman (2001)), and AnyBoost (Mason et al. (2000)). These methods are essentially based on the strategy of coordinate descent methods or functional gradient methods in an infinite-dimensional space of base classifiers; classifiers are added iteratively in greedy fashion with their weights in each iteration. As for the regularization, L_1-regularization in an infinite-dimensional space is adopted in these methods by restricting the model to convex combinations of base classifiers or imposing early stopping with a small learning rate (Friedman (2001); Rosset et al. (2004)) to prevent the rapid growth of its L_1-norm. The former formalization have also appeared in convex neural networks (Bengio et al. (2006); Bach (2014)) and boosting-like methods have been proposed. Moreover, Bach (2014) has studied the generalization ability of this problem and Rosset et al. (2007) has shown that the sparsity property still holds even in the case of an infinite-dimensional space.

In spite of the success of boosting methods in data analysis, it seems to be difficult to apply these methods for taking an ensemble of large size base classifiers such as neural networks because of the requirement of solving nonconvex subproblems for adding base classifiers, which can become intractable. In this chapter, we propose a new ensemble learning method, called *Stochastic Particle Gradient Descent* (SPGD) by adopting a completely different strategy based on easily executable sampling method for approximating a current probability measure. The SPGD performs in a space of probability measures on a set of continuously parameterized base classifiers and constructs an ensemble by the expectation with respect to the obtained probability measure. In other words, the SPGD method handles the L_1-constraint naturally, hence we are free from adjusting early stopping time and a complex ensemble can be found quickly. Such a procedure is realized by constructing a transport map for sampling base classifiers and this transport map iteratively grows by stacking layers on neural networks to optimize the objective function. This strategy is in opposition to those of existing methods that successively increase the number of basis to be combined.

In the theoretical analysis, we show the convergence of the proposed method to a stationary point with a convergence rate as fast as that of a stochastic optimization method for finite-dimensional nonconvex problems. Moreover, we show the *interior optimality property* of the method, where the optimization proceeds as long as better probability measures than a current probability measure exist in its support. This property is inherent in problems with respect to probability measures and its proof mainly relies on partial differential equation theory.

Furthermore, we provide two practical variants of SPGD method. One is a natural approximation of a transport map in SPGD using finite particles, and we note this approximation forms a residual-type network (He et al. (2016)). The other is a more practical variant without resampling of particles, and we show this variant can be regarded as well-initialized SGD for the nonweighted voting classification problem, that is, we can say it is an extension of the vanilla SGD to the method for optimizing a general probability measure.

Related Work

Ensemble learning with infinite models have been received a lot of attention due to their superior performance and many optimization methods have been exploited. Representative methods are boosting methods (Schapire et al. (1998)), convex (continuous) neural networks (Bengio et al. (2006); Le Roux and Bengio (2007); Rosset et al. (2007)), and Bayesian neural networks (MacKay (1992, 1995); Neal (2012)). Kernel methods using shift-invariant kernels (Rahimi and Recht (2007)) also combine a basis, although a base probability measure to sample base functions is pre-determined. Most of these methods are based on optimization problems in infinite-dimensional spaces with some regularization, for instance, boosting methods and convex neural networks use the L_1-regularization as mentioned earlier, that is, combinations by probability measures, and kernel methods with shift-invariant kernels use RKHS-norm regularization which is written by the L_2-regularization using an associated probability measure like infinite-layer networks (Livni et al. (2017)). L_2-regularized problems in kernel methods can be efficiently solved by the functional gradient descent (Kivinen et al. (2004); Dai et al. (2014); Zhu and Xu (2015); Dieuleveut and Bach (2016)) or methods using explicit random features (Rahimi and Recht (2007); Livni et al. (2017)). Note that the random kitchen sinks (Rahimi and Recht (2009)) adopt the L_∞-constraint rather than the L_2-regularization. As for the L_1-regularization, combining its good generalization performance (Koltchinskii and Panchenko (2002); Koltchinskii et al. (2003); Bach (2014)) with the fact that the L_1-ball always includes the L_2 (L_∞)-ball, superior classification performance is expected, especially when base classifiers have high representational ability. However, solving L_1-regularized problems are more challenging than L_2 (L_∞)-regularized problems because handling L_1-regularization is difficult from the optimization perspective, indeed boosting methods which are representative for this problem require to solve the nonconvex subproblems for adding base classifiers.

From the perspective of optimizing the probability measure, gradient-based Bayesian inference methods (Welling and Teh (2011); Dai et al. (2016); Liu and Wang (2016)) are related to ours. Especially, stein variational gradient descent (SVGD) proposed in Liu and Wang (2016) is most related to our work, which has a similar flavor to our method. Convergence analysis and gradient flow perspective were given in Liu (2017) and further analysis was provided in Chen and Zhang (2017). However, while SVGD is a method specialized to minimize the Kullback–Leibler-divergence based on Stein's identity technique, our method does not require special structure of a loss function; hence, our method can be applied to a wider class of problems and theoretical results hold in the more general setting, though we focus our study only on the ensemble learning. We would like to remark an interesting point of our method compared to the normalizing flow (Rezende and Mohamed (2015)) that approximates Bayes posterior through deep neural networks. In our method, a transport map is obtained by stacking residual-type layers (He et al. (2016)) iteratively, hence a residual network to output an infinite ensemble is built naturally.

7.2 Problem Setting

In this section, we explain our problem setting for classification tasks. We first define the space of base classifiers. Let us denote the Borel measurable feature space and the label set by $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Y} = \{-1, 1\}$, respectively. Let $\Theta \subset \mathbb{R}^d$ be a Borel set and $\mathcal{H} = \{h_\theta : \mathcal{X} \to [-1,1];\ \mathrm{Borel\ measurable} \mid \theta \in \Theta\}$ be a subset of base binary classifiers that is parameterized by $\theta \in \Theta$. We sometimes use the notation $h(\theta, x)$ for $h_\theta(x)$ when it is regarded as a function with respect to θ and x. We assume that $h(\theta, x)$ is Borel measurable on $\Theta \times \mathcal{X}$ and continuous with respect to $x \in \mathcal{X}$.

Example 1 (Linear Classifier)**.** *For $\theta \in \Theta$, we define the linear classifier as follows:*

$$h_\theta(x) = \tanh(\theta^\top x),$$

which separates the feature space $\mathcal{X}$ linearly using a hyperplane with normal vector θ.

Example 2 (Neural Network)**.** *Let $\{l_s\}_{s=0}^{L}$ be the sizes of layers, where $l_0 = n$, $l_L = 1$. We set $d = \sum_{s=0}^{L-1} l_s l_{s+1}$ and $\Theta = \prod_{s=0}^{L-1} \Theta_s$, where $\Theta_s \subset \mathbb{R}^{l_s l_{s+1}}$. For $\theta = \{\theta_s\}_{s=0}^{L-1} \in \Theta$, we define the classifier, that is, an L-layer neural network:*

$$h_\theta(x) = \tanh(\theta_L^\top \sigma(\theta_{L-1}^\top \sigma(\cdots \sigma(\theta_1^\top x)\cdots))),$$

where σ is a continuous activation function.

Let us denote the set of all Borel probability measures on Θ by $\mathcal{P}$. For $\mu \in \mathcal{P}$, we define the infinite ensemble (ensemble by the probability measure) $h_\mu : \mathcal{X} \to [-1,1]$ as follows: for $x \in \mathcal{X}$,

$$h_\mu(x) \stackrel{\mathrm{def}}{=} \mathbb{E}_\mu[h(\theta, x)],$$

where $\mathbb{E}_\mu$ is the expectation with respect to μ, and predict the label of x by $\mathrm{sign}(h_\mu(x))$. Let $\mathcal{G}$ be the set of all infinite ensembles: $\mathcal{G} = \{h_\mu \mid \mu \in \mathcal{P}\}$. The goal of the classification problem under our setting is to minimize the empirical risk on $\mathcal{P}$.

Let $l : \mathbb{R} \to \mathbb{R}$ be a loss function such as the exponential loss. Let us consider solving the following problem:

$$\min_{\mu \in \mathcal{P}} \mathcal{L}_S(\mu) \overset{\text{def}}{=} \frac{1}{N} \sum_{j=1}^{N} l\left(-y_j h_\mu(x_j)\right). \tag{7.1}$$

Noting that this problem can be cast as L_1-regularization problem in an infinite-dimensional problem, good statistical properties of the problem are well known (Bach (2014)). Moreover, since, this problem have an optimal solution of finite support (Rosset et al. (2007)), several generalization guarantees (Koltchinskii and Panchenko (2002); Blanchard et al. (2003); Bartlett et al. (2006)) for convex combinations can be applied to the problem (7.1). Therefore, we focus on how to solve this problem efficiently.

One way to make this infinite dimensional optimization problem computationally tractable is to parameterize its subspace locally by a space of actions, which may also be infinite-dimensional manifold. Basically, our proposed method sequentially updates a Borel probability measure on Θ based on the theory of transportation. That is, the current probability measure μ_k is updated through pushing-forward by a transport map having the form $id + \xi_k$ toward a direction reducing the objective function $\mathcal{L}_S$. Repeating this procedure, we finally obtain a composite transport map $\phi_T = (id + \xi_{T-1}) \circ \cdots (id + \xi_0)$ from an initial probability measure μ_0 and the corresponding probability measure μ_T is obtained by pushing-forward μ_0 by ϕ_T. In practice, the final probability measure μ_T is approximated by samples obtained through ϕ_T as $\phi_T(\theta_i) \sim \mu_T$ where $\theta_i \sim \mu_0$ and this approximation makes the method feasible. The resulting problem is how to choose ξ_k to optimize (7.1) and an answer to this question is to perform the functional gradient method with respect to ξ. To explain our proposed method correctly, we should describe the optimization domain with its properties.

7.3 Optimization Domain and Optimality Condition

In this section, we specify the optimization domain where the proposed method performs. We set $\Theta = \mathbb{R}^d$. Let μ denote any Borel probability measure on Θ with finite second moment $\mathbb{E}_\mu[\|\theta\|^2] < +\infty$ and $\mathcal{P}_2$ denote the set of such probability measures. We denote by $L_2(\mu)$ the space of $L_2(\mu)$-integrable maps from $\mathrm{supp}(\mu) \subset \Theta$ into Θ, equipped with $\langle \cdot, \cdot \rangle_{L_2(\mu)}$-inner product: for $\forall \xi, \forall \zeta \in L_2(\mu)$,

$$\langle \xi, \zeta \rangle_{L_2(\mu)} = \mathbb{E}_\mu[\xi(\theta)^\top \zeta(\theta)].$$

As defined in Chapter 6, we denote by $\phi_\sharp\mu$ the push-forward measure of μ by ϕ. When we use maps in $L_2(\mu)$ to push-forward probability measures, we call these as transport maps. We can clearly see $\phi_\sharp\mu$ is also contained in $\mathcal{P}_2$ when $\mu \in \mathcal{P}_2$ and id is contained in $L_2(\mu)$ for arbitrary $\mu \in \mathcal{P}_2$. Let us consider approximately solving the problem (7.1) on $\mathcal{P}_2$ by updating transport maps iteratively. To discuss the local behavior of the problem, we must specify the topology of $\mathcal{P}$; hence, we need to introduce more notions.

7.3.1 Integral Probability Metric on $\mathcal{P}$

In this chapter, we adopt a kind of integral probability metrics (Müller (1997)) on $\mathcal{P}$. For a positive constant $C > 0$, let f be a function on Θ such that it is uniformly bounded $|f(\theta)| \leq C$ and $f(\theta)$ is C-Lipschitz continuous on Θ with respect to the Euclidean norm. We denote by $\mathcal{F}_C$ the set of such functions and the subscript will be omitted for simplicity. This set of functions is used for defining the norm $\|\cdot\|_{\mathcal{F}}$ on the space of linear functionals on $\mathcal{F}$, which includes $\mathcal{P}$. Specifically, $\|\mu\|_{\mathcal{F}} = \sup_{f\in\mathcal{F}} |\mu(f)|$ for a finite signed measure μ on Θ, where we denote the integral of a function f with respect to $\mu \in \mathcal{P}$ by $\mu(f)$. Thus, $\mathcal{P}$ is a metric space with respect to the uniform distance $d_{\mathcal{F}}(\mu,\nu) = \|\mu - \nu\|_{\mathcal{F}}$ for $\mu, \nu \in \mathcal{P}$. The convergence $d_{\mathcal{F}}(\mu_t, \mu) \to 0$ is none other than the uniform convergence of integrals $\mu_t(f) \to \mu(f)$ on $\mathcal{F}$. Note that this norm defines the same topology as the Dudley metric (Dudley (1968)).

To investigate the local behavior of $\mathcal{L}_S(\mu)$, we need to clarify the continuity of several quantities depending on $\mu \in \mathcal{P}$ and $h(\cdot, x)$. Especially, $-l'(-yh_\mu(x))y\nabla_\theta h(\theta, x)$ is really important because it is used to describe an optimality condition and performs as the stochastic gradient in the function space. For simplicity, we use the notation

$$s_\mu(\theta, x, y) = -l'(-yh_\mu(x))y\nabla_\theta h(\theta, x). \tag{7.2}$$

The following proposition gives a sufficient condition for the continuity of these quantities.

Proposition 8. *Suppose sets $\{h(\cdot, x) \mid x \in \mathcal{X}\}$ and $\{\|\mathbb{E}_S[s_\mu(\cdot, x, y)]\|_2^2 \mid \mu \in \mathcal{P}\}$ are included in the set $\mathcal{F}_C$. Then, $h_\mu(x)$, $\mathcal{L}_S(\mu)$, and $\|\mathbb{E}_S[s_\mu(\theta, x, y)]\|_{L_2(\mu)}$ are continuous as a function of μ on $\mathcal{P}$ with respect to $\|\cdot\|_{\mathcal{F}}$.*

The next proposition supports the validity of the assumption in this proposition for Example 1 of the linear classifier.

Proposition 9. *Let the loss function l be a C^1-class function. If $h(\cdot, x)$ is two times continuously differentiable and $h(\cdot, x), \nabla_\theta h(\cdot, x)$ are Lipschitz continuous with respect to θ with the same constant for all $x \in \mathcal{X}$, then for sufficiently large $C > 0$, the assumption in Proposition 8 is satisfied.*

Note that, if we use the set of all 1-Lipschitz continuous functions instead of $\mathcal{F}$, it derives the Kantorovich and Rubinstein dual form of 1-Wasserstein distance. Although continuous functions with respect to $d_{\mathcal{F}}$ are also continuous with respect to larger distances including p-Wasserstein distances with $p \geq 1$, we prefer $d_{\mathcal{F}}$ in this work because it has sufficient size to contain the integrands $h(\cdot, x)$.

7.3.2 Local Optimality Condition

In this subsection, we establish the local optimality condition for the problem over $\mathcal{P}_2$. To achieve this goal, we need not only the continuity propositions, but also the counterpart of Taylor's formula in the space of probability measures, giving the intuition to construct and analyze an optimization method for solving the problem. Thus, we show such a proposition under the following assumption.

Assumption 7 (Smoothness and boundedness). *Let l be a $\mathcal{C}^2$-function and let h be a $\mathcal{C}^2$-function with respect to θ. Moreover, we assume $\nabla_\theta h(\theta, x)$, $\nabla_\theta^2 h(\theta, x)$, and the eigenvalues of the latter matrix are uniformly bounded on $\Theta \times \mathcal{X}$.*

Note that this assumption also holds for Example 1 of the linear classifier under the compactness of $\mathcal{X}$ and this assumption is essentially stronger than the assumption in Proposition 8. The following is the counterpart of Taylor's formula.

Proposition 10. *Suppose Assumption 7 holds. For $\forall \mu \in \mathcal{P}_2$ and $\forall \xi \in L_2(\mu)$, $\mathcal{L}_S((id + \xi)_\sharp \mu)$ can be represented as follows:*

$$\mathcal{L}_S((id + \xi)_\sharp \mu) = \mathcal{L}_S(\mu) + \mathbb{E}_\mu[\mathbb{E}_S[s_\mu(\theta, x, y)]^\top \xi(\theta)] + \frac{1}{2} H_\mu(\xi) + o(\|\xi\|_{L_2(\mu)}^2), \quad (7.3)$$

where $H_\mu(\xi) = O(\|\xi\|_{L_2(\mu)}^2)$ is described as follows: for θ', which is a convex combination of θ and $\theta + \xi(\theta)$ depending also on x, $H_\mu(\xi)$ is defined by

$$H_\mu(\xi) = -\mathbb{E}_S[yl'(-yh_\mu(x))\mathbb{E}_\mu\|\xi(\theta)\|^2_{\nabla_\theta^2 h(\theta', x)}] + \mathbb{E}_S[l''(-yh_\mu(x))\mathbb{E}_\mu[\nabla_\theta h(\theta, x)^\top \xi(\theta)]^2].$$

Proof Sketch. A perturbation of the probability measure μ can be translated into that on the parameter space θ in the integral with respect to μ by the change of variables formula: $\mathbb{E}_{(id+\xi)_\sharp \mu}[h(\theta, x)] = \mathbb{E}_\mu[h(\theta + \xi, x)]$. Therefore, applying Taylor's formula to h and l, we get

$$\begin{aligned}
\mathbb{E}_S[l(-yh_{(id+\xi)_\sharp \mu}(x))] &= \mathbb{E}_S[l(-y\mathbb{E}_\mu[h(\theta + \xi, x)])] \\
&\simeq \mathbb{E}_S[l(-y\mathbb{E}_\mu[h(\theta, x) + \nabla h(\theta, x)^\top \xi(\theta)])] \\
&\simeq \mathbb{E}_S[l(-yh_\mu(x)) + \mathbb{E}_\mu[s_\mu(\theta, x, y)^\top \xi(\theta)]],
\end{aligned}$$

where we omitted the second- and higher-order terms for simplicity. A complete proof will be given in the Appendix. □

Using this proposition, we can immediately derive a necessary local optimality condition over $\overline{\mathcal{P}}_2$ in a similar way to the finite-dimensional case, where $\overline{\mathcal{P}}_2$ is a closure of $\mathcal{P}_2$ in $\mathcal{P}$ with respect to $d_{\mathcal{F}}$. Note that from Assumption 7, $s_\mu(\cdot, x, y)$ is contained in $L_2(\mu)$ for any $\mu \in \mathcal{P}$.

Theorem 13 (Necessary local optimality condition)**.** *Suppose Assumption 7 holds. Let $\mu_* \in \overline{\mathcal{P}}_2$ be the local minimum of $\mathcal{L}_S(\mu)$ with respect to $d_{\mathcal{F}}$. Then we have*

$$\|\mathbb{E}_S[s_{\mu_*}(\theta, x, y)]\|_{L_2(\mu_*)} = 0. \tag{7.4}$$

7.3.3 Interior Optimality Property

In this section, we present the *interior optimality property* of the condition (7.4). When the loss function l is convex, the optimization problem (7.1) is also convex with respect to $\mu \in \mathcal{P}$ in terms of affine geometry, and hence the following holds: for a Borel measure $\forall\tau$ such that $\int d\tau(\theta) = 0$,

$$\mathcal{L}_S(\mu) + \int \nabla_\mu \mathcal{L}_S(\mu)(\theta) d\tau(\theta) \leq \mathcal{L}_S(\mu + \tau),$$

where $\nabla_\mu \mathcal{L}_S(\mu)$ is Fréchet derivative with respect to μ: $\nabla_\mu \mathcal{L}_S(\mu) = \mathbb{E}_S[-l'(-yh_\mu(x))yh(\cdot, x)]$. Thus, the equation, $\int \nabla_\mu \mathcal{L}_S(\mu)(\theta) d\tau(\theta) = 0$ (for $\forall\tau$ s.t. $\int d\tau(\theta) = 0$), is the global optimality condition. In general, this condition and the local optimality condition (7.4) are different. Indeed, in the set of Dirac measures, there may exist some local minima as finite-dimensional optimization problems but the global optimality condition is not satisfied. However, we can show the interior optimality property of local optimum by using the global optimality condition.

Theorem 14. *Suppose that h is a $\mathcal{C}^1$-function with respect to θ and the loss function l is a $\mathcal{C}^1$-convex function. Let $\mu_* \in \mathcal{P}$ be a probability measure having a continuous density function. If* $\mathrm{supp}(\mu_*)$ *is a compact $\mathcal{C}^\infty$-manifold with boundary and μ_* satisfies the local optimality condition (7.4), then there is neither measure μ having a continuous density such that* $\mathrm{supp}(\mu) \subset \mathrm{supp}(\mu_*)$ *and $\mathcal{L}_S(\mu) < \mathcal{L}_S(\mu_*)$ nor μ not having a continuous density such that* $\mathrm{supp}(\mu)$ *is contained in the interior of* $\mathrm{supp}(\mu_*)$ *and $\mathcal{L}_S(\mu) < \mathcal{L}_S(\mu_*)$.*

This theorem states that the optimization proceeds as long as there exists a better probability measure in support of a current measure μ satisfying the same assumptions on μ_* in Theorem 14 except for condition (7.4).

So far, we have discussed the local optimality conditions and we have confirmed that $\|\mathbb{E}_S[s_\mu(\theta, x, y)]\|_{L_2(\mu)}$ for $\mu \in \mathcal{P}_2$ can be regarded as the local optimality quantity for the problem due to its continuity and the above theorems. Therefore, the goal of an optimization method for the problems is to output a sequence $\{\mu_t\}_{t=1}^\infty \subset \mathcal{P}_2$ such that $\|\mathbb{E}_S[s_{\mu_t}(\theta, x, y)]\|_{L_2(\mu_t)}$ converges to zero.

The following proposition ensures the existence of an accumulation point of such a sequence satisfying the local optimality condition under the tightness assumption on generated probability measures.

Proposition 11. *We assume a sequence $\{\mu_t\}_{t=1}^{\infty}$ in $\mathcal{P}$ is tight, that is, for arbitrary $\epsilon > 0$, there exists a compact subset $\mathcal{A} \subset \Theta$ such that $\mu_t(\mathcal{A}) \geq 1 - \epsilon$ for $\forall t \in \mathbb{N}$. Then, this sequence has a convergent subsequence with respect to $\|\cdot\|_{\mathcal{F}}$.*

Therefore, by taking a subsequence if necessary, we can obtain a sequence $\{\mu_t\}_{t=1}^{\infty}$ to converge to a stationary point $\mu_* \in \mathcal{P}$ by an appropriate method. Note that this convergence implies the uniform convergence $\mu_t(h(\cdot,x)) \to \mu_*(h(\cdot,x))$ for all $x \in \mathcal{X}$ by Proposition 8.

7.4 Stochastic Particle Gradient Descent

In this section, we introduce a stochastic optimization method for solving problem (7.1) on $\mathcal{P}_2$ and present its convergence analysis. The proposed method is essentially the same as the functional gradient descent described in Chapter 6, but we describe it by another perspective. We first present an overview of our method again. Let $\mu_0 \in \mathcal{P}_2$ be an initial probability measure and suppose a current probability measure μ is obtained by pushing-forward μ_0 by $\phi \in L_2(\mu_0)$. Then, ϕ and μ are updated along $\xi \in L_2(\mu)$ as $\phi^+ \leftarrow (id + \xi) \circ \phi$ and $\mu^+ \leftarrow (id + \xi)_\sharp \mu$. The resulting problem is how to obtain ξ to locally minimize the objective function $\mathcal{L}_S((id+\xi)_\sharp\mu)$ on $L_2(\mu)$. We can find that by Taylor's formula (7.3), this objective is Fréchet differentiable with respect to $\xi \in L_2(\mu)$ and its differential is represented by $\mathbb{E}_S[s_\mu(\cdot,x,y)]$ via the $L_2(\mu)$-inner product. Thus, this differential performs in function space with this inner-product like the usual gradient in a finite-dimensional space and it is expected to reduce the objective value. We next provide a more detailed description below.

Let us denote by $B_r(\mu)$ the r-neighborhood of the origin in $L_2(\mu)$; $B_r(\mu) \overset{\text{def}}{=} \{\xi \in L_2(\mu) \mid \|\xi\|_{L_2(\mu)} < r\}$. Since the higher-order term $H_\mu(\xi) + o(\|\xi\|^2_{L_2(\mu)})$ in (7.3) is $O(\|\xi\|^2_{L_2(\mu)})$, it can be locally upper bounded by the quadratic form at $\xi = 0$. Thus, we can assume that there exists a positive-definite smooth (d,d)-matrix $A_\mu(\theta)$ such that for all $\xi \in B_r(\mu)$,

$$\frac{1}{2}H_\mu(\xi) + o(\|\xi\|^2_{L_2(\mu)}) \leq \frac{1}{2}\mathbb{E}_\mu\|\xi\|^2_{A_\mu(\theta)}. \tag{7.5}$$

Note that we can choose scalar matrix cI_d as A_μ with $c > 0$ that does not depend on μ under Assumption 7. By Proposition 10, the following quadratic function with respect to ξ is a local upper bound on $\mathcal{L}_S((id+\xi)_\sharp\mu)$ at $\mu \in \mathcal{P}_2$:

$$\mathcal{L}_S(\mu) + \mathbb{E}_\mu[\mathbb{E}_S[s_\mu(\theta,x,y)]^\top \xi(\theta)] + \frac{1}{2}\mathbb{E}_\mu\|\xi\|^2_{A_\mu(\theta)}. \tag{7.6}$$

Thus, minimizing (7.6) as a surrogate function, we can obtain $\xi \in L_2(\mu)$ to reduce the objective $\mathcal{L}_S$ and we can make an update $\mu^+ \leftarrow (id+\xi)_\sharp\mu$ and an update $\phi^+ \leftarrow (id+\xi)\circ\phi$ for the corresponding transport map from the initial probability measure. Practically, such a solution ξ is obtained by minimizing the following stochastic approximation to (7.6): for randomly chosen (x', y') from S,

$$\min_{\xi \in B_r(\mu)} \mathbb{E}_\mu[s_\mu(\theta, x', y')^\top \xi(\theta)] + \frac{1}{2}\mathbb{E}_\mu\|\xi\|^2_{A_\mu(\theta)}. \tag{7.7}$$

Note that under Assumption 7 and uniformly boundedness assumption on A_μ, a positive constant η_0 exists such that for $\forall\mu \in \mathcal{P}$ and $\forall(x', y') \in \mathcal{X} \times \mathcal{Y}$,

$$\eta_0\|A_\mu(\theta)^{-1}s_\mu(\theta, x', y')\|_{L_2(\mu)} < r.$$

Thus, we can choose the step $-\eta A_\mu(\theta)^{-1}s_\mu(\theta, x', y')$ $(0 < \eta < \eta_0)$ as an approximate solution to (7.7) and Lemma 11 shows the reduction of the objective function by this step. Moreover, if η_0 is sufficiently small, we can find this step produces a diffeomorphism (see Appendix), which preserves good properties of the initial probability measure such as the manifold structure, and it may lead to good exploration of the proposed method by an intuition from Theorem 14.

Lemma 11 (Descent Lemma). *Suppose Assumption 7 holds and suppose* $0 \prec \lambda_A I_d \preceq A_\mu(\theta) \preceq \Lambda_A I_d$. *We set* $\zeta(\theta) = -A_\mu(\theta)^{-1}s_\mu(\theta, x', y')$ *Then, there exist* $G > 0$ *and* $\eta_0 > 0$, *depending on the smoothness, the boundedness of* l, h, A_μ, *and the radius* r, *such that for* $0 < \forall\eta < \eta_0$, $\eta\zeta$ *is contained in* $B_r(\mu)$ *and it leads to a reduction:*

$$\mathbb{E}_S[\mathcal{L}_S((id - \eta\zeta)_\sharp\mu)] \leq \mathcal{L}_S(\mu) - \frac{\eta}{\Lambda_A}\|\mathbb{E}_S[s_\mu(\theta, x, y)]\|^2_{L_2(\mu)} + \eta^2 G.$$

This lemma means that for sufficiently small learning rates $\eta > 0$, the iterate $\mu_+ \leftarrow (id + \eta\xi)_\sharp\mu$ strictly reduces the objective function $\mathcal{L}_S$ in the expectation when μ does not satisfy the local optimality condition (7.4). Here, we propose an algorithm called SPGD in Algorithm 13 to solve problem (7.1) based on the above analyses. Note that Algorithm 13 is the ideal one, and hence a practical variant will be described later.

Depending on the choice of A_μ, we can derive several specific algorithms as in the traditional (stochastic) optimization literature, e.g. steepest descent method, natural gradient method, and quasi-Newton method. Thus, Algorithm 13 can be regarded as the simplest form, where $A_\mu = cI_d$ $(c > 0)$, of SPGD. We can obtain the convergence theorem for Algorithm 13 by the inequality of Lemma 11.

Theorem 15 (Convergence Theorem). *Let us make the same assumptions as in Lemma 11. For* $\epsilon > 0$, *let* $\eta > 0$ *be a constant satisfying* $\eta \leq \min\{\eta_0, \frac{\epsilon}{2G}\}$. *Then an* ϵ*-accurate solution in the expectation, i.e.,* $\mathbb{E}[\|\mathbb{E}_S[s_{\mu_k}(\theta, x, y)]\|^2_{L_2(\mu_k)}] \leq \epsilon$, *where the outer expectation is taken with respect to the history of sample data used in learning, can be obtained*

Algorithm 13 SPGD

```
**Input:** dataset $S$, initial distribution $\mu_0$, the maximum number of iterations $T$, learning rates $\{\eta_k\}_{k=0}^{T-1}$
$\phi_0 \leftarrow id$
**for** $k = 0$ **to** $T-1$ **do**
    Randomly choose a sample $(x', y')$ from $S$
    $\phi_{k+1} \leftarrow (id - \eta_k s_{\mu_k}(\cdot, x', y')) \circ \phi_k$
    $\mu_{k+1} \leftarrow (id - \eta_k s_{\mu_k}(\cdot, x', y'))_\sharp \mu_k$
**end for**
Return $\phi_T$
```

at the most

$$\frac{2(\mathcal{L}_S(\mu_0) - \inf_{\mu \in \mathcal{Q}} \mathcal{L}_S(\mu))}{\epsilon \eta} \tag{7.8}$$

iterations of Algorithm 13 with learning rate $\eta_k = \eta$.

Running Algorithm 13, we obtain the transport map ϕ_T. If we choose a tractable distribution as the initial distribution μ_0, we can obtain i.i.d. particles $\{\theta_i^0\}_{i=1}^M$ from μ_0. By the construction of ϕ_k, we find that $\{\phi_k(\theta_i^0)\}_{i=1}^M$ are regarded as i.i.d. particles from the distribution $\mu_k = \phi_{k\sharp}\mu_0$. However, note that Algorithm 13 is impractical because we cannot compute exact value of $h_{\mu_k}(x')$ required to get $s_{\mu_k}(\cdot, x', y')$. Thus, we estimate it using sample average $h_k \sim \frac{1}{M}\sum_{i=1}^M h_{\theta_i^k}$. where $\theta_i^k = \phi_k(\theta_i^0)$. The overall procedure is summarized in Algorithm 14. Because of the form of $id + \eta_k l'(-y'h_k)y'\nabla_\theta h(\cdot, x')$, we notice that Algorithm 14 iteratively stacks residual-type layers (He et al. (2016)) and so a residual network to output an infinite ensemble is built naturally. We can also derive more practical variant Algorithm 15 without resampling in Algorithm 14, that is, using the same seeds $\{\theta_i^0\}_{i=1}^M$ over all iterations.

7.4.1 Extension of Vanilla Stochastic Gradient Descent

Let us explain how the proposed method can be regarded as an extension of vanilla SGD in a finite-dimensional space. If we adopt the sum of Dirac measures as the initial distribution μ_0, then Algorithm 13 and 15 become the same method by initializing particles $\{\theta_i^0\}_{i=1}^M$ to be the support of μ_0. Moreover, we can see that the step of Algorithm 15 is the same as that of vanilla SGD for the nonweighted voting problem: $\min_{\{\theta_i\}\in\Theta^M} \mathbb{E}_S[l(-\frac{1}{M}\sum_{i=1}^M yh(\theta_i, x))]$. Specifically, we can say that the vanilla SGD for learning a base classifier is the method to optimize a Dirac measure and is none other than Algorithm 13 with a Dirac measure μ_0. In other words, Algorithm 13 is an extension of the vanilla SGD to the method for optimizing a general probability measure.

Algorithm 14 SPGD - building residual network -

```
Input: dataset $S$, initial distribution $\mu_0$, the maximum number of iterations $T$, the number of particles $M$, learning rates $\{\eta_k\}_{k=0}^{T-1}$
$\phi_0 \leftarrow id$
for $k = 0$ to $T-1$ do
    Independently draw particles $\{\theta_i^0\}_{i=1}^M$ from $\mu_0$
    $\{\theta_i^k\}_{i=1}^M \leftarrow \{\phi_k(\theta_i^0)\}_{i=1}^M$
    Randomly choose a sample $(x', y')$ from $S$
    $h_k \leftarrow \frac{1}{M}\sum_{i=1}^M h_{\theta_i^k}(x')$
    $\phi_{k+1} \leftarrow (id + \eta_k l'(-y'h_k)y'\nabla_\theta h(\cdot, x')) \circ \phi_k$
end for
Return $\{\theta_i^T\}_{i=1}^M$
```

Algorithm 15 SPGD - practical variant -

```
Input: dataset $S$, initial distribution $\mu_0$, the maximum number of iterations $T$, the number of particles $M$, learning rates $\{\eta_k\}_{k=0}^{T-1}$
Independently draw particles $\{\theta_i^0\}_{i=1}^M$ from $\mu_0$
for $k = 0$ to $T-1$ do
    Randomly choose a sample $(x', y')$ from $S$
    $h_k \leftarrow \frac{1}{M}\sum_{i=1}^M h_{\theta_i^k}(x')$
    $\{\theta_i^{k+1}\}_{i=1}^M \leftarrow \{\theta_i^k + \eta_k l'(-y'h_k)y'\nabla_\theta h(\theta_i^k, x')\}_{i=1}^M$
end for
Return $\{\theta_i^T\}_{i=1}^M$
```

From this viewpoint of Algorithm 15, we can introduce some existing techniques and extensions to our method. For instance, we can use accelerating techniques such as Nesterov's momentum method (Nesterov (2004)), which is also used in our experiments to accelerate the convergence.

Moreover, we can extend Algorithm 15 to the multiclass classification problems. Let us consider the c-classes classification problem. We denote the binary vector for the class by $\mathbf{y}$, that is, for the i-th class, only the i-th element y_i is one and the other elements are zeros. The output of the classifier $\mathbf{h}$ is extended to the range $[0,1]^c$, which represent the confidences of each class such as the softmax function. Then, the SGD for the problem $\min_{\{\theta_i\}\in\Theta^M} \mathbb{E}_S[l(-\frac{1}{M}\sum_{i=1}^M \mathbf{y}^\top \mathbf{h}(\theta_i, x))]$ is the extension of Algorithm 15 to the multiclass problem.

7.5 Numerical Experiments

7.5.1 Synthetic Data

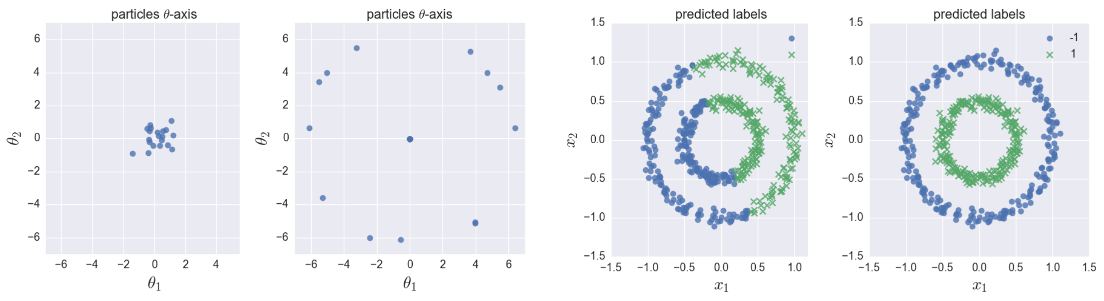

Figure 7.1: Toy example of the SPGD method (upper-left: weights of the initial particles; upper-right: weights of the final particles; bottom-left: predicted labels by the initial particles; bottom-right: predicted labels by the final particles).

We first present how our method behaves by using toy data: two-dimensional double circle data. We ran Algorithm 15 for Example 1 of a binary linear model with exponential loss;

$$\min_{\theta_i \in \mathbb{R}^2, b_i \in \mathbb{R}} \mathbb{E}_S \left[\exp\left(-\frac{1}{M} \sum_{i=1}^{M} Y \tanh(\theta_i^\top X + b_i) \right) \right].$$

The number of particles was set to be 20. The behavior of the method is shown in Figure 7.1. The upper-left part shows weights θ of the initial particles and the upper-right part shows θ of the final particles. The bottom row represents predicted labels by the initial particles (left) and the final particles (right). It can be seen that the data are well classified by the locations of the particles using this method.

7.5.2 Real Data

Next, we present the results of experiments on binary and multiclass classification tasks in a real dataset. We ran Algorithm 15 with momentum for logistic regression and three-layer perceptrons where we set the number of hidden units to be the same as the input dimension and we used sigmoid activation for the output of the hidden layer. For the last layer of multilayer perceptrons, we used softmax output with the exponential loss or the logarithmic loss function. The number of particles was set to be 10 or 30. Each element of initial particles was sampled from the normal distribution μ_0 with zero mean and standard deviation of 0.01 to bias parameters and of 1 to weight parameters. To

Table 7.1: Test classification accuracy on binary and multiclass classification.

Dataset	LogReg	SPGD(logreg)	MLP(exp)	SPGD(exp)	MLP(log)	SPGD(log)
breastcancer	**0.966** **(0.0187)**	0.965 (0.0177)	0.965 (0.0210)	**0.971** **(0.0174)**	0.968 (0.0110)	**0.971** **(0.0174)**
diabetes	0.755 (0.0464)	**0.761** **(0.0435)**	**0.764** **(0.0366)**	0.756 (0.0447)	0.738 (0.0524)	**0.757** **(0.0400)**
german	**0.769** **(0.0406)**	0.763 (0.0390)	0.738 (0.0178)	**0.769** **(0.0381)**	0.724 (0.0393)	**0.775** **(0.0356)**
ionosphereo	**0.892** **(0.0400)**	0.886 (0.0383)	0.914 (0.0512)	**0.937** **(0.0274)**	0.923 (0.0339)	**0.937** **(0.0274)**
glass	0.566 (0.0655)	**0.622** **(0.0692)**	0.477 (0.1127)	**0.616** **(0.0595)**	0.619 (0.1144)	**0.659** **(0.1033)**
segment	**0.934** **(0.0148)**	0.913 (0.0143)	0.717 (0.1104)	**0.953** **(0.0100)**	0.961 (0.0082)	**0.970** **(0.0089)**
vehicle	0.771 (0.0422)	**0.780** (0.0248)	0.759 (0.0372)	**0.838** **(0.0451)**	0.794 (0.0525)	**0.829** **(0.0370)**
wine	0.968 (0.0321)	**0.978** (0.0377)	0.949 (0.0519)	**0.974** **(0.0414)**	0.963 (0.0552)	**0.984** **(0.0246)**
covertype	0.720 (0.0071)	**0.738** **(0.0056)**	**0.772** **(0.0269)**	0.763 (0.0255)	0.772 (0.0271)	**0.806** **(0.0247)**

evaluate the performance of the SPGD, we also ran logistic regression and multilayer perceptron, whose structure is the same as used for SPGD.

We used the UCI datasets: breast-cancer, diabetes, german, and ionosphere for binary classification; glass, segment, vehicle, wine, and covertype for multiclass classification. We used the following experimental procedure as in Cortes et al. (2014); we first divided each dataset into 10 folds. For each run $i \in \{1, \ldots, 10\}$, we used fold i for validation, used fold $i + 1 \pmod{10}$ for testing, and used the other folds for training. We performed each method on the training dataset with several hyper-parameter settings and we chose the best parameter on the validation dataset. Finally, we evaluated it on the testing dataset.

The mean classification accuracy and the standard deviation are presented in Table 7.1. Notations SPGD(LOGREG), SPGD(EXP), and SPGD(LOG) stand for SPGD for logistic regression, multilayer perceptrons with exponential loss, and with logarithmic loss function, respectively. Although SPGD did not improve logistic regression on some datasets, it showed overall improvements over base models on the other settings. Thus, we confirmed the effectiveness of our method.

7.6 Appendix

7.6.1 Topological Properties and Optimality Conditions

In this section, we prove statements about the optimization problem for majority vote classifiers.

Proof of Proposition 9 . By the assumption, uniform boundedness, Lipschitz continuity of $h(\cdot, x)$ and uniform boundedness of $\|\mathbb{E}_S[s_\mu(\theta, x, y)]\|_2^2$ are clear. Thus, it is sufficient to show uniform Lipschitz continuity of the latter functions. Let us define functions $\phi_\alpha(z) = \|\sum_{i=1}^N \alpha_i z_i / N\|_2^2$ (where $z_i \in \mathbb{R}^d$, $\exists K$, $\alpha \in [-K, K]^N$) and mappings $\psi_S(\theta) = (\nabla_\theta h(\theta, x_i))_{i=1}^N$. By the boundedness assumption there is a constant $C > 0$ such that $\|\nabla_\theta h(\theta, x)\|_2 \leq C$. Note that $\phi_\alpha|_{[-C,C]^{dN}}$ and ψ_S are Lipschitz continuous with the uniformly bounded constant. Thus, composite functions of these; $\{\phi_\alpha \circ \psi_S\}_{\alpha \in [-C,C]^{dN}, S}$ are also Lipschitz continuous with the uniformly bounded constant. Clearly, functions $\|\mathbb{E}_S[s_\mu(\theta, x, y)]\|_2^2$ is an element of these composite functions, so this finishes the proof. □

We now give propositions needed in our analysis. The first statement in the following proposition shows that the distance between ϕ and $\phi + \xi \circ \phi$ with respect to $L_2(\mu)$ is the norm of ξ with respect to $L_2(\phi_\sharp \mu)$. The second statement gives a sufficient condition for a vector to define a diffeomorphism that preserves good properties if the base probability measures possesses these, for instance, the absolute continuity with respect to Lebesgue measure and the manifold structure of the support of itself which are sometimes useful from the Wasserstein geometry or partial differential equation perspective.

Proposition 12. *For $\mu \in \mathcal{P}_2$, the following statements are valid:*
(i) $\|(id + \xi) \circ \phi - \phi\|_{L_2(\mu)} = \|\xi\|_{L_2(\phi_\sharp \mu)}$ for $\phi \in L_2(\mu), \xi \in L_2(\phi_\sharp \mu)$;
(ii) Let $\xi \in L_2(\mu)$ be the $\mathcal{C}^1$-mapping from the convex hull of $\mathrm{supp}(\mu)$ *to Θ. We denote by Λ an upper bound on maximum singular values of $\nabla \xi(\theta)$ as the (d, d)-matrix on the convex hull of* $\mathrm{supp}(\mu)$. *Then $id + \eta\xi$ is a diffeomorphism on* $\mathrm{supp}(\mu)$ *for* $0 \leq \forall \eta < 1/\Lambda$.

Proof. We set $\mu = \phi_\sharp \mu_0$ for $\phi \in L_2(\mu_0)$. Then we have that for $\xi \in L_2(\mu)$

$$\begin{aligned}
\|\xi\|_{L_2(\mu)}^2 &= \int \|\xi(\theta)\|_2^2 d\mu(\theta) \\
&= \int \|\xi(\theta)\|_2^2 d\phi_\sharp \mu_0(\theta) \\
&= \int \|\xi(\phi(\theta))\|_2^2 d\mu_0(\theta) = \|\xi \circ \phi\|_{L_2(\mu_0)}^2,
\end{aligned}$$

where we used the variable transformation for the third equality. This finishes the proof of (i).

If we assume $(id + \eta\xi)(\theta) = (id + \eta\xi)(\theta')$, then it follows that $\|\theta - \theta'\|_2 = \eta\|\xi(\theta) - \xi(\theta')\|_2 \leq \eta\|\nabla\xi(\theta'')^T(\theta - \theta')\|_2 < \eta\Lambda\|\theta - \theta'\|_2$, where θ'' is a convex combination of θ and θ'. Since $\eta\Lambda < 1$, we have $\theta = \theta'$, i.e., $id + \eta\xi$ is an injective mapping. By the same argument, we find that $\nabla(id+\eta\xi)(\theta) = I_d + \eta\nabla\xi(\theta)$ also defines an injective linear mapping for $\theta \in \mathrm{supp}(\mu)$ and $\eta\Lambda < 1$, so that this matrix is invertible. Thus, we conclude the proof of (ii) by using the invertible mapping theorem. $\square$

Here, we present the proof of Proposition 8 and the continuity of the parameterization via transport maps in the following propositions, which will be used to show a local optimality condition theorem.

Proof of Proposition 8 . Continuity of $h_\mu(x)$ and $\mathcal{L}_S(\mu)$ with respect to μ are clear. Let $\{\mu_t\}_{t=1}^\infty$ be a sequence converging to $\mu \in \mathcal{P}$. In the following, we denote $s_\mu(\theta, x, y)$ by s_μ for simplicity. By triangle inequality, we have

$$\begin{aligned}\left|\|\mathbb{E}_S[s_{\mu_t}]\|_{L_2(\mu_t)}^2 - \|\mathbb{E}_S[s_\mu]\|_{L_2(\mu)}^2\right| \leq{}& \left|\mu_t(\|\mathbb{E}_S[s_{\mu_t}]\|_2^2) - \mu_t(\|\mathbb{E}_S[s_\mu]\|_2^2)\right| \\ &+ \left|\mu_t(\|\mathbb{E}_S[s_\mu]\|_2^2) - \mu(\|\mathbb{E}_S[s_\mu]\|_2^2)\right|.\end{aligned}$$

Since $\|\mathbb{E}_S[s_\mu]\|_2^2 \in \mathcal{F}$, the latter term converges to zero. In order to show that the former converges to zero, it is sufficient to see the uniform convergence $\|\mathbb{E}_S[s_{\mu_t}]\|_2^2 \to \|\mathbb{E}_S[s_\mu]\|_2^2$. By the boundedness and the triangle inequality, we have

$$\begin{aligned}\left|\|\mathbb{E}_S[s_{\mu_t}]\|_2^2 - \|\mathbb{E}_S[s_\mu]\|_2^2\right| &\leq 2\sqrt{C}\left|\|\mathbb{E}_S[s_{\mu_t}]\|_2 - \|\mathbb{E}_S[s_\mu]\|_2\right| \\ &\leq 2\sqrt{C}\|\mathbb{E}_S[s_{\mu_t}] - \mathbb{E}_S[s_\mu]\|_2.\end{aligned}$$

This upper bound converges to zero. Indeed, each element in expectation: $s_{\mu_t}(\cdot, x, y)$ uniformly converges to $s_\mu(\cdot, x, y)$ as seen in the following:

$$\|l'(-y h_{\mu_t}(x))\nabla h(\theta, x) - l'(-y h_\mu(x))\nabla h(\theta, x)\|_2 \leq C|l'(-y h_{\mu_t}(x)) - l'(-y h_\mu(x))| \to 0.$$

This finishes the proof. $\square$

Proposition 13. *For $\forall\mu \in \mathcal{P}_2$ and $\forall\xi \in L_2(\mu)$, it follows that $d_\mathcal{F}((id + \xi)_\sharp\mu, \mu) \leq C\|\xi\|_{L_2(\mu)}$.*

Proof of Proposition 13 . Noting that Lipschitz continuity of $\forall f \in \mathcal{F}$, we have that for $\forall\xi \in L_2(\mu)$,

$$\begin{aligned}d_\mathcal{F}((id + \xi)_\sharp\mu, \mu) &= \sup_{f\in\mathcal{F}} |((id + \xi)_\sharp\mu)(f) - \mu(f)| \\ &= \sup_{f\in\mathcal{F}} \left|\int f(\theta)d(id + \xi)_\sharp\mu(\theta) - \int f(\theta)d\mu(\theta)\right| \\ &= \sup_{f\in\mathcal{F}} \left|\int (f(\theta + \xi(\theta)) - f(\theta))\, d\mu(\theta)\right|\end{aligned}$$

$$\leq C \int \|\xi(\theta)\|_2 d\mu(\theta) \leq C\|\xi\|_{L_2(\mu)},$$

where we used Hölder's inequality for the last inequality. □

As noted in the paper, the continuity in Proposition 8 also holds with respect to p-Wasserstein distance ($p \geq 1$) and Proposition 13 holds for 1-Wasserstein distance with $C = 1$.

We now give the proof of the counterpart of Taylor's formula.

Proof of Proposition 10 . By the variable transformation, we have

$$\int h(\theta, x) d(id + \xi)_\sharp \mu(\theta) = \int h(\theta + \xi(\theta), x) d\mu(\theta).$$

Using Taylor's formula, we obtain

$$h_{\psi\sharp\mu_0}(x) = h_\mu(\theta) + \mathbb{E}_\mu[\nabla_\theta h(\theta, x)^T \xi(\theta) + \|\xi(\theta)\|^2_{\nabla^2_\theta h(\theta', x)}],$$

where $\|\cdot\|_{\nabla_\theta h(\theta', x)}$ is Mahalanobis norm, and

$$l(a + b) = l(a) + l'(a)b + \frac{1}{2}l''(a)b^2 + o(b^2) \quad (a, b \in \mathbb{R}).$$

Noting that by Hölder's inequality and Assumption 7, $\mathbb{E}_\mu[\nabla_\theta h(\theta, x)^T \xi(\theta)] = O(\|\xi\|_{L_2(\mu)})$ and $\mathbb{E}_\mu[\|\xi(\theta)\|^2_{\nabla_\theta h(\theta', x)}] = o(\|\xi\|_{L_2(\mu)})$, we get

$$l(-yh_{\psi_\sharp\mu_0}(x)) = l(-yh_\mu(x)) + \mathbb{E}_\mu[s_\mu(\theta, x, y)^T \xi(\theta)] + H_\mu(\xi, x, y) + o(\|\xi\|^2_{L_2(\mu)}),$$

where $H_\mu(\xi, x, y)$ is the integrand in $H_\mu(\xi)$. Therefore, by taking the expectation $\mathbb{E}_S$, we finish the proof. □

Using facts and propositions presented in the paper, we prove the theorem of a necessary optimality condition.

Proof of Theorem 13 . We denote $\zeta_\mu = \mathbb{E}_S[s_\mu(\cdot, x, y)]$ and denote the δ-ball centered at μ_* by $B^{\mathcal{F}}_\delta(\mu_*)$ with respect to $d_{\mathcal{F}}$. We assume μ_* is a minimum on $B^{\mathcal{F}}_\delta(\mu_*)$. By Assumption 7 and Proposition 13, there exists $\eta_0 > 0$ such that $(id \pm \eta\zeta_\mu)_\sharp \mu \in B^{\mathcal{F}}_\delta(\mu_*) \cap \mathcal{P}_2$ for $0 < \forall\eta < \eta_0$ and $\forall\mu \in B^{\mathcal{F}}_{\delta/2}(\mu_*) \cap \mathcal{P}_2$. Let $\epsilon > 0$ be an arbitrary constant. Here, we can choose a sequence $\{\mu_t\}_{t=1}^\infty$ in $B^{\mathcal{F}}_{\delta/2}(\mu_*) \cap \mathcal{P}_2$ satisfying $\mu_t \to \mu_*$ and $\mathcal{L}_S(\mu_t) \leq \mathcal{L}_S(\mu_*) + \epsilon/t$ by the continuity of $\mathcal{L}_S$. Then, using Proposition 10, we have

$$-\frac{\epsilon}{t} \leq \mathcal{L}_S\left(\left(id - \frac{\eta_0}{t}\zeta_{\mu_t}\right)_\sharp \mu_t\right) - \mathcal{L}_S(\mu_t) = -\frac{\eta_0}{t}\|\mathbb{E}_S[s_{\mu_t}]\|^2_{L_2(\mu_t)} + \frac{\eta_0^2}{t^2} O(\|\mathbb{E}_S[s_{\mu_t}]\|^2_{L_2(\mu_t)}),$$

where we denote $s_{\mu_t} = s_{\mu_t}(\theta, x, y)$ for simplicity. Note that Assumption 7 is essentially stronger than the assumption in Proposition 8 and the continuity of $\mathcal{L}_S(\mu)$ and $\|\mathbb{E}_S[s_\mu]\|^2_{L_2(\mu)}$ with respect to μ are valid by Proposition 8. Thus, multiplying t, taking the limit as $t \to \infty$, and using continuity, we have $\eta_0 \|\mathbb{E}_S[s_{\mu_*}]\|^2_{L_2(\mu_*)} \leq \epsilon$. Since ϵ is taken arbitrary and ϵ, η_0 are independent of each other, we get $\|\mathbb{E}_S[s_{\mu_*}]\|^2_{L_2(\mu_*)} = 0$ □

We next provide the proof of Proposition 11.

Proof of Proposition 11 . For $\forall \epsilon > 0$, let $\mathcal{A}$ be a compact subset in Θ such that $\mu_t(\mathcal{A}) \geq 1 - \epsilon$ Let $\mathcal{P}(\mathcal{A})$ denote the set of Borel probability measures on $\mathcal{A}$ and $\mu'_t \in \mathcal{P}(\mathcal{A})$ be the rescaled probability measure of $\mu_t|_{\mathcal{A}}$, i.e., $\mu'_t = \mu_t|_{\mathcal{A}}/\mu_t(\mathcal{A})$. Note that measures in $\mathcal{P}(\mathcal{A})$ are naturally extended to the whole space Θ and the distance between μ_t and μ'_t is bounded as follows: for $\forall f \in \mathcal{F}$,

$$\begin{aligned}
|\mu_t(f) - \mu'_t(f)| &= \left| \int f(\theta) d\mu_t - \int f(\theta) d\mu'_t \right| \\
&\leq \left| \int_{\mathcal{A}} f(\theta) d\mu_t - \int_{\mathcal{A}} f(\theta) d\mu'_t \right| + \left| \int_{\mathcal{A}^c} f(\theta) d\mu_t \right| \\
&\leq \left| \int_{\mathcal{A}} f(\theta)(1 - 1/\mu_t(\mathcal{A})) d\mu_t|_{\mathcal{A}} \right| + C\epsilon \\
&\leq C\left(\frac{1}{\mu_t(\mathcal{A})} - 1 + \epsilon \right) \leq C\epsilon \left(1 + \frac{1}{1-\epsilon} \right).
\end{aligned}$$

Since $\mathcal{P}(\mathcal{A})$ is a compact with respect to the topology of weak convergence, we can take a convergent subsequence of $\{\mu'_t\}_{t=1}^{\infty}$. Let us denote this by $\{\mu'_{t_k}\}_{k=1}^{\infty}$ and its limit by $\mu \in \mathcal{P}(\mathcal{A})$. Using Ascoli-Arzela theorem, we can see $\mathcal{F}|_{\mathcal{A}}$ is relatively compact in the set of continuous functions on $\mathcal{A}$ with uniform norm $\|\cdot\|_\infty$. Noting that the set of the continuous functions on a compact set is complete and that relatively compactness and totally boundedness are equivalent, we can conclude $\mathcal{F}|_{\mathcal{A}}$ is totally bounded. Thus, for $\forall \epsilon$, we have a ϵ-ball covering; $\{B^\infty_{f_i}(\epsilon)\}_{i=1}^{l}$ $(f_i \in \mathcal{F}|_{\mathcal{A}})$, where $B^\infty_f(\cdot)$ denote a ball centered at f with respect to the uniform norm. Hence, for any $f \in \mathcal{F}|_{\mathcal{A}}$, there is $i \in \{1, \ldots, l\}$ such that $f \in B^\infty_{f_i}(\epsilon)$.

Therefore, we have that for $\forall f \in \mathcal{F}$,

$$\begin{aligned}
|\mu_{t_k}(f) - \mu(f)| &= \left| \int f(\theta) d\mu_{t_k} - \int f(\theta) d\mu \right| \\
&\leq \left| \int f(\theta) d\mu'_{t_k} - \int f(\theta) d\mu \right| + \left| \int f(\theta) d\mu_{t_k} - \int f(\theta) d\mu'_{t_k} \right| \\
&\leq \left| \int f(\theta) d\mu'_{t_k} - \int f_i(\theta) d\mu'_{t_k} \right| + \left| \int f_i(\theta) d\mu'_{t_k} - \int f_i(\theta) d\mu \right|
\end{aligned}$$

$$\begin{aligned}&+\left|\int f_i(\theta)d\mu-\int f(\theta)d\mu\right|+O(\epsilon)\\&\leq O(\epsilon)+\left|\int f_i(\theta)d\mu'_{t_k}-\int f_i(\theta)d\mu\right|.\end{aligned}$$

Since, μ'_{t_k} weakly converges to μ in $\mathcal{P}(\mathcal{A})$, there is $k_0 \in \mathbb{N}$ such that $\forall k \geq k_0$, $\max_{j\in\{1,\ldots,l\}} |\mu'_{t_k}(f_j)-\mu(f_j)| \leq \epsilon$. Thus, we can conclude $\sup_{f\in\mathcal{F}} |\mu_{t_k}(f)-\mu(f)| \leq \exists C'\epsilon$ for $k \geq k_0$. This implies $\mu_{t_k} \to \mu$ with respect to $d_{\mathcal{F}}$. □

7.6.2 Interior Optimality Property

To prove Theorem 14, we now introduce the notion of the smoothing of probability measures as Schwartz distribution. We denote by χ a $\mathcal{C}^\infty$-class probability density function on $\Theta = \mathbb{R}^d$ with $\mathrm{supp}(\chi) = \{\theta \in \Theta \mid \|\theta\|_2 \leq 1\}$ and write $\chi_\epsilon(\theta) = \epsilon^{-d}\chi(\theta/\epsilon)$ for $\epsilon > 0$. For a probability measure $\mu \in \mathcal{P}$, it can be approximated by a smooth probability density function defined by the following:

$$(\mu * \chi_\epsilon)(\theta) = \int_\Theta \chi_\epsilon(\theta-\theta')d\mu(\theta').$$

It is well known that $\mu * \chi_\epsilon$ is $\mathcal{C}^\infty$-class on Θ and converges as Schwartz distribution to μ as $\epsilon \to 0$ (Hörmander (1963)). Moreover, if μ possesses a L_p-integrable probability density function $q \in L_p(\Theta)$ with $p \geq 1$, then $\mu * \chi_\epsilon$ converges to q with respect to $L_p(\Theta)$-norm. Let μ_ϵ denote a probability measure induced by $\mu * \chi_\epsilon$. When $\mathrm{supp}(\mu_\epsilon)$ is compact in Θ, $\mu_\epsilon(f)$ converges to $\mu(f)$ for arbitrary continuous function f on Θ. This can be confirmed by constructing a $\mathcal{C}^\infty$-function g that uniformly approximates f on $\mathrm{supp}(\mu_\epsilon)$ and takes the value zero outside of sufficiently large compact set. Clearly, we see that $\mathrm{supp}(\mu_\epsilon)$ is contained in the closed ϵ-neighborhood of $\mathrm{supp}(\mu)$. Thus, if $\mathrm{supp}(\mu)$ is compact, then $\{\mu_\epsilon\}_{\epsilon\in(0,1)}$ is tight, so that we can find μ_ϵ converges to μ with respect to $\|\cdot\|_{\mathcal{F}}$ by the proof of Proposition 11, that is, $\mu_\epsilon(f)$ converges uniformly to $\mu(f)$ on $\mathcal{F}$.

Note that if $\mathrm{supp}(\mu)$ is the compact submanifold in Θ, $\mathrm{supp}(\mu)$ and the closed ϵ-neighborhood of $\mathrm{supp}(\mu)$ coincide for sufficiently small $\epsilon > 0$ and these sets possess a manifold structure as can be seen by the following auxiliary lemma.

Lemma 12. *Let $\mathcal{M}$ be a l-dimensional compact $\mathcal{C}^\infty$-submanifold ($l < d$) or a d-dimensional compact $\mathcal{C}^\infty$-submanifold with boundary in $\mathbb{R}^d$. If $\epsilon > 0$ is sufficiently small, then closed ϵ-neighborhood of $\mathcal{M}$ in $\mathbb{R}^d$ is a d-dimensional compact $\mathcal{C}^\infty$-submanifold with boundary.*

Proof. We only prove the case where $\mathcal{M}$ is a compact $\mathcal{C}^\infty$-submanifold since we can give a proof for a d-dimensional compact $\mathcal{C}^\infty$-submanifold with boundary in a similar fashion. Let $\mathcal{M}^\epsilon$ denote an open ϵ-neighborhood of $\mathcal{M}$ in $\Theta = \mathbb{R}^d$. By the ϵ-neighborhood theorem (Guillemin and Pollack (1974)), if ϵ is sufficiently small, then $\forall\theta \in \mathcal{M}^\epsilon$ possesses a unique closest point $\pi(\theta)$ in $\mathcal{M}$ and the map $\pi : \mathcal{M}^\epsilon \to \mathcal{M}$ is a submersion. Moreover, for each $\theta_0 \in \mathcal{Y}$, we can see that there is a local coordinate system $z = (z^1, \ldots, z^d) = \phi(\theta)$ on an open subset $U \subset \Theta$ such that $\phi(\theta_0) = (0, \ldots, 0)$, $\phi(\mathcal{M} \cap U) = \{z \in \phi(U) \mid z^{l+1} = \cdots = z^d = 0\}$, and the submersion π can be written as $\pi(\phi^{-1}(z)) = \phi^{-1}(z^1, \ldots, z^l, 0, \ldots, 0)$ for $z \in \phi(\mathcal{M}^\epsilon \cap U)$. Since, $\mathcal{M}$ is compact, the closed ϵ-neighborhood $\overline{\mathcal{M}^\epsilon}$ is covered by a finite number of such local coordinate systems for sufficiently small $\epsilon > 0$. We redefine (U, ϕ) to be one of such local coordinate system. The Euclidean distance to $\mathcal{M}$ from $\phi^{-1}(z) \in U$ is $f(z) = d(\phi^{-1}(z), \mathcal{M}) = \|\phi^{-1}(z) - \phi^{-1}(z^1, \ldots, z^l, 0, \ldots, 0)\|_2$ and $\overline{\mathcal{M}^\epsilon} \cap U$ is represented as $\{\phi^{-1}(z) \mid z \in \phi(U), f(z) \le \epsilon\}$. Since, $f(\cdot)$ is a $\mathcal{C}^\infty$-function and $df \neq 0$ on a neighborhood of $\partial\overline{\mathcal{M}^\epsilon}$ in U, $\overline{\mathcal{M}^\epsilon} \cap U$ is a d-dimensional compact $\mathcal{C}^\infty$-submanifold with boundary in Θ. □

Let U be a bounded domain with smooth boundary in $\Theta = \mathbb{R}^d$, that is, $\overline{U}$ is a d-dimensional $\mathcal{C}^\infty$-manifold with boundary. We denote by $H^1(U)(= W^{1,2}(U))$ the Sobolev space and we denote by $V(U)$ a linear subspace $\{f \in H^1(U) \mid \int_U f d\theta = 0\}$. We equip $H^1(U)$ with the Sobolev inner product $\langle u, v\rangle_{H^1(U)} = \int_U u(\theta)v(\theta)d\theta + \int_U \nabla u(\theta)^\top \nabla v(\theta) d\theta$ and we equip $V(U)$ with the inner product $\langle u, v\rangle_{V(U)} = \int_U \nabla u(\theta)^\top \nabla v(\theta)d\theta$, $(u, v \in V(U))$. The non-degeneracy and the completeness of $\langle , \rangle_{V(U)}$ on $V(U)$ can be checked as follows. We denote $\overline{u} = \int_U u(\theta)d\theta/|U|$, where $|U|$ is the Lebesgue measure of U. Since $\overline{u} = 0$ for $u \in V(U)$, we get from the Poincaré-Wirtinger inequality that there exists $C_U > 0$ such that

$$\|u\|_{L_2(U)} = \|u - \overline{u}\|_{L_2(U)} \le C_U \|\nabla u\|_{L_2(U)} = C_U \|u\|_{V(U)}. \tag{7.9}$$

Thus, we have

$$\|u\|_{V(U)} \le \|u\|_{H^1(U)} = \sqrt{\|u\|^2_{L_2(U)} + \|\nabla u\|^2_{L_2(U)}} \le (1 + C_U)\|u\|_{V(U)}.$$

This inequality means that these two norms introduce the same topology to $V(U)$ and it immediately implies the non-degeneracy and also the completeness of $\|\cdot\|_{V(U)}$ on $V(U)$ because $V(U)$ is the closed subspace in the Sobolev space $H^1(U)$ with respect to $\|\cdot\|_{H^1(U)}$. Therefore, $V(U)$ with $\langle , \rangle_{V(U)}$ is actually Hilbert space.

Although the Poincaré constant C_U depends on a region U, it is known that for any $R > 0$ there exists $C_R > 0$ such that if U is an ϵ-open neighborhood of a connected set $K \subset B_R(0) = \{\theta \in \Theta \mid \|\theta\|_2 < R\}$ for some constant $\epsilon > 0$, then C_U can be taken as it is upper bounded by C_R (Ruiz (2012)).

In our analysis, we need an estimation of the norm of a solution to the problem where for $f \in V(U)$, the task is to find $u \in V(U)$ satisfying the following equation:

$$\int_U \nabla u(\theta)^\top \nabla v(\theta) d\theta = -\int_U f(\theta)v(\theta)d\theta \ \ for\ any\ v \in V(U). \tag{7.10}$$

This is the weak formulation of the *Neumann problem*: to find $u \in V(U)$ such that $\Delta u = f$ in U and $\partial u/\partial n = 0$ on ∂U, where n is the outward pointing unit normal vector of ∂U. An upper bound on the norm of a solution is given by the following lemma which can be proven in the standard way in partial differential equation theory.

Lemma 13. *Let U be a bounded domain in $\Theta = \mathbb{R}^d$. Then for any $f \in V(U)$, a solution $u_* \in V(U)$ to the problem (7.10) exists and its norm is bounded as follows:*

$$\|u_*\|_{V(U)} \leq \|\alpha_f\|_{V(U)^*}, \tag{7.11}$$

where α_f is a linear functional $\alpha_f(u) = \int_U f(\theta)u(\theta)d\theta$ $(u \in V(U))$ and $\|\cdot\|_{V(U)^*}$ denote the dual of the norm $\|\cdot\|_{V(U)}$.

Proof. We denote $\beta(u,v) = \int_U \nabla u(\theta)^\top \nabla v(\theta)d\theta$ for $u, v \in V(U)$. Clearly, $\beta(\cdot,\cdot)$ is bilinear function. The boundedness with respect to $\|\cdot\|_{V(U)}$ are shown as follows. Using Hölder's inequality and the inequality (7.9), we have that for $u, v \in V(U)$,

$$|\beta(u,v)| = \left| \int_U \nabla u(\theta)^\top \nabla v(\theta)d\theta \right| \leq \|u\|_{L_2(U)}\|v\|_{L_2(U)} \leq C_U^2 \|u\|_{V(U)}\|v\|_{V(U)}.$$

Moreover, $\beta(\cdot,\cdot)$ is 1-coercive because $\beta(u,u) = \|u\|_{V(U)}^2$. We can also see that $\alpha_f(\cdot)$ is a bounded linear functional in the same manner: for $u \in V(U)$,

$$|\alpha_f(u)| = \left| \int_U f(\theta)u(\theta)d\theta \right| \leq \|f\|_{L_2(U)}\|u\|_{L_2(U)} \leq C_U \|f\|_{L_2(U)}\|u\|_{V(U)}.$$

Thus, by the Lax-Milgram theorem, there is a unique solution $u_* \in V(U)$ and we have $\|u_*\|_{V(U)} \leq \|\alpha_f\|_{V(U)^*}$. □

We now give the proof of Theorem 14 that gives an interior optimality property of the local optimality condition.

Proof of Theorem 14 . We denote $\Omega = \mathrm{supp}(\mu_*)$ and let q_* be a continuous probability density function of μ_*. We assume that there exists $\mu' \in \mathcal{P}$ that possesses a continuous probability density function q' and satisfies $\mathrm{supp}(\mu') \subset \Omega$, $\mathcal{L}_S(\mu') < \mathcal{L}_S(\mu_*)$. By smoothing μ_* and μ' with sufficiently small $\epsilon > 0$, we can obtain $d\mu'_\epsilon = q'_\epsilon(\theta)d\theta$ and $d\mu_{*\epsilon} = q_{*\epsilon}(\theta)d\theta$ where $q'_\epsilon, q_{*\epsilon}$ are $\mathcal{C}^\infty$-density functions satisfying $\mathrm{supp}(\mu'_\epsilon) \subset \mathrm{supp}(\mu_{*\epsilon})$. As stated above, $q'_\epsilon, q_{*\epsilon}$ converge to q', q_* in $L_2(\Theta)$.

Let us denote $\Omega_\epsilon = \mathrm{supp}(\mu_{*\epsilon})$ Since $q'_\epsilon - q_{*\epsilon}$ is $\mathcal{C}^\infty$-function and $\int_{\Omega_\epsilon}(q'_\epsilon - q_{*\epsilon})d\theta = 0$, there is a $\mathcal{C}^\infty$-function ψ_ϵ on Ω_ϵ that solves the Neumann problem (Hörmander (1963)):

$$\Delta\psi_\epsilon = q'_\epsilon - q_{*\epsilon} \;\; in \;\; \Omega_\epsilon, \;\; \partial\psi_\epsilon/\partial n = 0 \;\; on \;\; \partial\Omega_\epsilon,$$

where $\partial\Omega_\epsilon$ is the boundary of Ω_ϵ and n is the outward pointing unit normal vector of $\partial\Omega_\epsilon$. By adding a constant, we assume $\int_{\Omega_\epsilon}\psi_\epsilon(\theta)d\theta = 0$, i.e., $\psi_\epsilon|_{\Omega_\epsilon^i} \in V(\Omega_\epsilon^i)$, where Ω_ϵ^i is the interior of Ω_ϵ. Therefore, we have

$$\begin{aligned}\int_{\Omega_\epsilon} \nabla_\mu \mathcal{L}_S(\mu_{*\epsilon})(\theta)d(\mu'_\epsilon - \mu_{*\epsilon}) &= \int_{\Omega_\epsilon} \nabla_\mu \mathcal{L}_S(\mu_{*\epsilon})(\theta)\Delta\psi_\epsilon(\theta)d\theta \\ &= -\int_{\Omega_\epsilon} \nabla_\theta\nabla_\mu \mathcal{L}_S(\mu_{*\epsilon})(\theta)^\top \nabla_\theta\psi_\epsilon(\theta)d\theta \\ &\quad + \int_{\partial\Omega_\epsilon} \nabla_\mu \mathcal{L}_S(\mu_{*\epsilon})(\theta)\frac{\partial\psi_\epsilon(\theta)}{\partial n}d\partial\Omega_\epsilon \\ &= -\int_{\Omega_\epsilon} \mathbb{E}_S[s_{\mu_{*\epsilon}}(\theta,x,y)]^\top \nabla_\theta\psi_\epsilon(\theta)d\theta, \end{aligned} \tag{7.12}$$

where for the second equality we used Green's formula and for the last equality we used $\partial\psi_\epsilon/\partial n = 0$. By the convexity of $\mathcal{L}_S$ with respect to μ in terms of Affine geometry and $\mathcal{L}_S(\mu') < \mathcal{L}_S(\mu_*)$, we have

$$\lim_{\epsilon\to 0}\int_{\Omega_\epsilon} \nabla_\mu \mathcal{L}_S(\mu_{*\epsilon})(\theta)d(\mu'_\epsilon - \mu_{*\epsilon}) \le \lim_{\epsilon\to 0}\mathcal{L}_S(\mu'_\epsilon) - \mathcal{L}_S(\mu_{*\epsilon}) = \mathcal{L}_S(\mu') - \mathcal{L}_S(\mu_*) < 0. \tag{7.13}$$

By the boundedness of Ω, we can assume it is contained in a ball with radius $R > 0$ centered around 0. Since, ψ_ϵ solves (7.10) with $U = \Omega_\epsilon^i$ and $f = q'_\epsilon - q_{*\epsilon}$, we get that by Lemma 13,

$$\begin{aligned}\lim_{\epsilon\to 0}\|\psi_\epsilon\|_{V(\Omega_\epsilon^i)} &\le \lim_{\epsilon\to 0}\sup_{\|u\|_{V(\Omega_\epsilon^i)}\le 1}\left|\int_{\Omega_\epsilon^i}(q'_\epsilon - q_{*\epsilon})(\theta)u(\theta)d\theta\right| \\ &\le \lim_{\epsilon\to 0}\sup_{\|u\|_{V(\Omega_\epsilon^i)}\le 1}\|q'_\epsilon - q_{*\epsilon}\|_{L_2(\Omega_\epsilon^i)}\|u\|_{L_2(\Omega_\epsilon^i)} \\ &\le \lim_{\epsilon\to 0}\|q'_\epsilon - q_{*\epsilon}\|_{L_2(\Omega_\epsilon^i)}\sup_{\|u\|_{V(\Omega_\epsilon^i)}\le 1} C_{\Omega_\epsilon^i}\|u\|_{V(\Omega_\epsilon^i)} \\ &\le C_R\|q' - q_*\|_{L_2(\Theta)},\end{aligned}$$

where we used the Poincaré-Wirtinger inequality (7.9) and uniform boundedness of $C_{\Omega_\epsilon^i}$. Thus, the limit as $\epsilon \to 0$ in the right hand side of (7.12) is lower-bounded by

$$-\lim_{\epsilon\to 0}\|\mathbb{E}_S[s_{\mu_{*\epsilon}}(\theta,x,y)]\|_{L_2(\Omega_\epsilon)}C_R\|q' - q_*\|_{L_2(\Theta)} = -\|\mathbb{E}_S[s_{\mu_*}(\theta,x,y)]\|_{L_2(\Omega)}C_R\|q' - q_*\|_{L_2(\Theta)}. \tag{7.14}$$

Combining (7.13) and (7.14), we find $\mathbb{E}_S[s_{\mu_*}(\theta,x,y)] \not\equiv 0$ on Ω, so μ_* does not satisfy the local optimality condition (7.4). This finishes the proof of the theorem.

For the case where μ does not have a continuous density, we can show the same result in a similar way by smoothing μ with as its support is contained in Ω. □

7.6.3 Convergence Analysis

In this section, we prove the convergence theorem of the proposed method.

Proof of Lemma 11 . Putting $\eta\zeta$ into (7.3) and (7.5), we obtain

$$\mathcal{L}_S((id + \eta\zeta)_\sharp\mu) \leq \mathcal{L}_S(\mu) - \eta\mathbb{E}_\mu[\mathbb{E}_S[s_\mu(\theta, x, y)]^T A_\mu(\theta)^{-1} s_\mu(\theta, x', y')] + \frac{\eta^2}{2}\mathbb{E}_\mu[\|s_\mu(\theta, x', y')\|^2_{A_\mu(\theta)^{-1}}].$$

Note that by the assumption, there exists $G > 0$ such that $\mathbb{E}_\mu[\|s_\mu(\theta, x', y')\|^2_{A_\mu(\theta)^{-1}}] < G$. Moreover, using the bound on $A_\mu(\theta)^{-1}$ and taking the expectation with respect to (x', y') (i.e., $\mathbb{E}_S$), we can finish the proof. □

Proof of Theorem 15 . Using the Lemma 11, we can see the updates of Algorithm 13 decreases the objective value as follows:

$$\mathbb{E}_S[\mathcal{L}_S(\mu_{k+1})] \leq \mathcal{L}_S(\mu_k) - \eta\|\mathbb{E}_S[s_{\mu_k}(\theta, x, y)]\|^2_{L_2(\mu_k)} + \eta^2 G.$$

Taking an expectation of the history of sample, summing up $k \in \{1, \ldots, t-1\}$, and dividing by $t\eta$, we have

$$\frac{1}{t}\sum_{k=1}^{t}\mathbb{E}[\|\mathbb{E}_S[s_{\mu_k}(\theta, x, y)]\|^2_{L_2(\mu_k)}] \leq \frac{\mathcal{L}_S(\mu_0) - \inf_{\mu\in Q}\mathcal{L}_S(\mu)}{\eta t} + \eta G.$$

Thus, if $t \geq \frac{2(\mathcal{L}_S(\mu_0) - \inf_q \mathcal{L}_S(\mu))}{\epsilon\eta}$, then $\frac{1}{t}\sum_{k=1}^{t}\mathbb{E}[\|\mathbb{E}_S[s_{\mu_k}(\theta, x, y)]\|^2_{L_2(\mu_k)}] \leq \epsilon$. This means the method can find ϵ-accurate solution with respect to the expectation, up to t iterations. □

7.6.4 Functional Gradient Aspect of SPGD

In this section, we provide the functional gradient method perspective of SPGD, that is, we describe a connection between SPGD and the functional gradient method in $L_2(\mu_0)$ where μ_0 is the fixed initial probability measure in the method.

Though, we have introduced our method to optimize a probability measure, it also can be readily recognized as the method to optimize a transport map in $L_2(\mu_0)$ if we fix the initial distribution $\mu_0 \in \mathcal{P}_2$. Indeed, since a composite function $\phi \circ \psi$ is contained in $L_2(\mu_0)$ when $\psi \in L_2(\mu_0)$ and $\phi \in L_2(\psi_\sharp\mu_0)$, so obtained transport maps by Algorithm 13 also belong to $L_2(\mu_0)$. Thus objective function can be translated to the form of $\mathcal{L}_S(\phi) = \mathcal{L}_S(\phi_\sharp\mu_0)$ with respect to $\phi \in L_2(\mu_0)$. Note that in general, since an initial distribution is usually variable in several trials, such a translation does not make sense.

In a similar manner to the proof of Proposition 10, we can obtain the following formula: for $\phi, \tau \in L_2(\mu_0)$,

$$\mathcal{L}_S(\phi + \tau) = \mathcal{L}_S(\phi) + \mathbb{E}_{\mu_0}[\mathbb{E}_S[s_\mu(\phi(\theta), x, y)]^\top \tau(\theta)] + H_\phi(\tau) + o(\|\tau\|^2_{L_2(\mu_0)}),$$

where $\mu = \phi_\sharp \mu_0$ and $H_\phi(\tau) = O(\|\tau\|^2_{L_2(\mu_0)})$. Thus, this formula indicates $\mathcal{L}_S(\phi)$ is Fréchet differentiable with respect to ϕ. We can see its differential is represented by $\mathbb{E}_S[s_\mu(\phi(\theta), x, y)]$ and $s_\mu(\phi(\theta), x, y)$ is the stochastic gradient via $L_2(\mu_0)$-inner product. Therefore, we can perform a stochastic variant of the functional gradient method (Luenberger (1969)) to minimize $\mathcal{L}_S(\phi)$ on $L_2(\mu_0)$ and its update rule becomes as follows:

$$\phi^+ \leftarrow \phi - \eta s_\mu(\phi(\cdot), x, y) = (id - \eta s_\mu(\cdot, x, y)) \circ \phi.$$

We immediately notice the equivalence between this update and Algorithm 13, so SPGD method is nothing but the stochastic functional gradient method if the initial distribution μ_0 is fixed. However, we note that to consider the problem with respect to a probability measure μ is important because it can lead to a much better understanding of the problem as seen before.

Chapter 8

Enhancing the Convergence of Adversarial Training

We propose a new technique that boosts the convergence of training generative adversarial networks. Generally, the rate of training deep models reduces severely after multiple iterations. A key reason for this phenomenon is that a deep network is expressed using a highly nonconvex finite-dimensional model, and thus the parameter gets stuck in a local optimum. Because of this, methods often suffer not only from degeneration of the convergence speed but also from limitations in the representational power of the trained network. To overcome this issue, we propose an additional layer called the *gradient layer* to seek a descent direction in an *infinite-dimensional space*. Because the layer is constructed in the infinite-dimensional space, we are not restricted by the specific model structure of finite-dimensional models. As a result, we can get out of the local optima in finite-dimensional models and move towards the global optimal function more directly. In this chapter, this phenomenon is explained from the functional gradient method perspective of the gradient layer. Interestingly, the optimization procedure using the gradient layer naturally constructs the deep structure of the network. Moreover, we demonstrate that this procedure can be regarded as a discretization method of the gradient flow that naturally reduces the objective function. Finally, the method is tested using several numerical experiments, which show its fast convergence.

This chapter is based on the work *Gradient Layer: Enhancing the Convergence of Adversarial Training for Generative Models*, A. Nitanda and T. Suzuki, Artificial Intelligence and Statistics, 2018 (Nitanda and Suzuki, 2018b).

8.1 Overview

Generative adversarial networks (GANs) (Goodfellow et al., 2014) are a promising scheme for learning generative models. GANs are trained by a discriminator and a generator in an adversarial way. Discriminators are trained to classify between real samples and fake samples drawn from generators, whereas generators are trained to mimic real samples. Although training GANs is quite difficult, adversarial learning succeeded in generating very impressive samples (Radford et al., 2016), and there are many subsequent studies (Larsen et al., 2016; Salimans et al., 2016; Nowozin et al., 2016; Chen et al., 2016; Zhang et al., 2017). Wasserstein GANs (WGANs) (Arjovsky et al., 2017) are a variant to remedy the mode collapse that appears in the standard GANs by using the Wasserstein distance (Villani, 2008), although they also sometimes generate low-quality samples or fail to converge. Moreover, an improved variant of WGANs was also proposed (Gulrajani et al., 2017) and it succeeded in generating high-quality samples and stabilizing WGANs. Although these attempts have provided better results, there is still scope to improve the performance of GANs further.

One reason for this difficulty stems from the limitation of the representational power of the generator. If the discriminator is optimized for the generator, the behavior is solely determined by the samples produced from that generator. In other words, for a generator with a poor representational power, the discriminator terminates its learning in the early stage and consequently results in having low discriminative power. However, for a finite-dimensional parameterized generator, the ability to generate novel samples to cheat the discriminators is limited. In addition, the highly nonconvex structure of the deep neural network for the generator prevents us from finding a direction for improvement. As a result, the trained parameter gets stuck in a local optimum and the training procedure does not proceed any more.

In this study, we propose a new learning procedure to overcome the issues of limited representational power and local optimum by introducing a new type of layer called a *gradient layer*. The gradient layer finds a direction for improvement in an infinite-dimensional space by computing the *functional gradient* (Luenberger, 1969) instead of the ordinary gradient induced by a finite-dimensional model. Because the functional gradient used for the gradient layer is not limited in the tangent space of a finite-dimensional model, it has much more freedom than the ordinary finite-dimensional one. Thanks to this property, our method can break the limit of the local optimum induced by the strong non-convexity of a finite-dimensional model, which gives much more representational power to the generator. We theoretically justify this phenomenon from the functional gradient method perspective and rigorously present a convergence analysis. Interestingly, one iteration of the method can be recognized as inserting one layer into the generator and the total number of iterations is the number of inserted layers. Therefore, our learning procedure naturally constructs the deep neural network architecture by inserting gradient layers. Although gradient layers can be inserted into an arbitrary layer, they are typically stacked on top of the generator in the final training phase to improve the generated sample quality.

Moreover, we provide another interesting perspective of the gradient layer, i.e., dis-

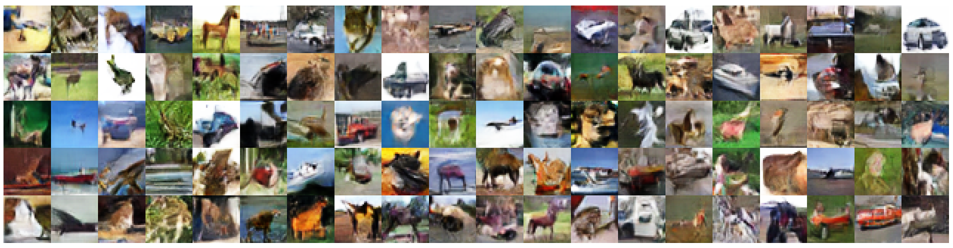

Figure 8.1: Random samples drawn from the generator trained by Algorithm 16 on the CIFAR-10 dataset.

cretization of the gradient flow in the space of probability measures. In Euclidean space, the steepest descent which is the typical optimization method, can be derived by discretizing the gradient flow that naturally produces a curve to reduce the objective function. Because the goal of GANs is to generate a sequence of probability measures moving to the empirical distribution by training samples, it is natural to consider a gradient flow in the space of probability measures defined by a distance between generated distribution and the empirical distribution and to discretize it in order to construct practical algorithms. We show that the functional gradient method for optimizing the generator in the function space is such a discretization method; in other words, the gradient flow can be tracked by stacking gradient layers successively.

The recently proposed SteinGAN (Wang and Liu, 2016) is closely related to our work and has a similar flavor, but it is based on another strategy to track gradient flow. That is, since that discretization is mimicked by a fixed-size deep neural network in SteinGAN, it may have the same limitation as typical GANs. By contrast, our method directly tracks the gradient flow in the final phase of training GANs to break the limit of the finite-dimensional generator.

8.2 Brief Review of Wasserstein GANs

In this section, we introduce WGANs and their variants. Although our proposed gradient layer is applicable to various models, we demonstrate how it performs well for the training of generative models; in particular, we treat Wasserstein GANs as a main application in this chapter. Let us start from briefly reviewing WGANs.

WGAN is a powerful generative model based on the 1-Wasserstein distance, defined as the L^1 minimum cost of transporting one probability distribution to the other. Let $\mathcal{X} \subset \mathbb{R}^v$ and $\mathcal{Z} \subset \mathbb{R}^h$ be a compact convex data space and a hidden space, respectively. A typical example of $\mathcal{X}$ is the image space $[0,1]^v$. For a noise distribution μ_n on $\mathcal{Z}$, WGAN learns a data generator $g : \mathcal{Z} \to \mathcal{X}$ to minimize an approximation to the 1-Wasserstein distance between the data distribution μ_D and the push-forward distribution $g_\sharp \mu_n$, which is a distribution that the random variable $g(z)$ follows when $z \sim \mu_n$ (in other words, the distribution obtained by applying a coordinate transform g to $z \sim \mu_n$). That is, WGAN can be described as the following $\min\max$ problem by using a Kantrovich-Rubinstein duality form of the 1-Wasserstein distance:

$$\min_{g \in \mathcal{G}} \max_{f \in \mathcal{F}} \mathcal{L}(f, g) \overset{\text{def}}{=} \mathbb{E}_{x \sim \mu_D}[f(x)] - \mathbb{E}_{z \sim \mu_n}[f \circ g(z)],$$

where $\mathcal{G}$ is the set of generators and $\mathcal{F}$ is an approximate set to the set of 1-Lipschitz continuous functions called *critic*. In WGANs, $\mathcal{G}, \mathcal{F}$ are parameterized neural networks $\{g_\theta\}, \{f_\tau\}$ and the problem is solved by alternate optimization: maximizing and minimizing $\mathcal{L}(f_\tau, g_\theta)$ with respect to τ and θ, alternately.

In practice, to impose the Lipschitz continuity on critics f_τ, penalization techniques were explored. For instance, the original WGANs (Arjovsky et al., 2017) use weight clipping $\|\tau\|_\infty \leq c$, which implies the upper-bound on the norm of $\nabla_\tau f_\tau$ and makes it Lipschitz continuous. However, it was pointed out in a subsequent study (Gulrajani et al., 2017) that such a restriction seems to be unnatural and sometimes leads to a low-quality generator or a failure to converge. In the same study, an improved variant of WGANs called WGAN-GP was proposed, which succeeded in stabilizing the optimization process and generating high-quality samples. WGAN-GP (Gulrajani et al., 2017) adds the gradient penalty $(\|\nabla_{\tilde{x}} f_\tau(\tilde{x})\|_2 - 1)^2$ to the objective function in the training phase of critics, where $\tilde{x}$ is a random interpolation between a training example $x \sim \mu_D$ and a generated sample $g(z) \sim g_{\theta\sharp}\mu_n$, i.e., $\tilde{x} \leftarrow \epsilon x + (1 - \epsilon) g(z)$ ($\epsilon \sim U[0, 1]$: uniform distribution). DRAGAN (Kodali et al., 2017) is a similar method to WGAN-GP, although it is based on a different motivation. DRAGAN also uses the gradient penalty, but the penalty is imposed on a neighborhood of the data manifold by a random perturbation of a training example.

WGAN and its variants are learned by alternately optimizing f_τ and g_θ, as stated above. We can regard this learning procedure as a problem of minimizing $\mathcal{L}(g_\theta) \overset{\text{def}}{=} \max_\tau\{\mathcal{L}(f_\tau, g_\theta) - \lambda R_\tau\}$, where R_τ is a penalty term. Let $\mathcal{L}(f_\tau, g_\theta) - \lambda R_\tau$ attain its maximum value at τ_* for g_θ. Then, the gradient $\nabla_\theta \mathcal{L}(g_\theta)$ is the same as $-\mathbb{E}_{\mu_n}[\nabla_\tau f_{\tau_*}^\top \nabla_\theta g_\theta(z)]$ by the envelope theorem (Milgrom and Segal, 2002) when both terms are well-defined. The differentiability of $\mathcal{L}(g_\theta)$ with respect to θ almost everywhere is proved in Arjovsky et al. (2017) under a reasonable assumption. Hence, we can apply the gradient method to this problem by approximating this gradient with finite particles generated from μ_n. However, because it is difficult to obtain f_{τ_*}, we run the gradient method for several iterations on training a critic instead of exactly computing f_{τ_*} at each g_θ. We can notice that this learning procedure is quite similar to that of the standard GAN (Goodfellow et al., 2014).

8.3 Gradient Layer

In the usual training procedure of WGANs, though more general maps are admissible for the original purpose, generators are parameterized by finite-dimensional space as described in the previous section, and the parameter may get stuck in a local optimum induced by this restriction, or the speed of convergence may reduce. In this work, we propose a gradient layer that accelerates the convergence and breaks the limit of finite-dimensional models. This layer is theoretically derived by the infinite-dimensional optimization method. We first explain the high-level idea of the gradient layer that strictly improves the ability of generator and why our method enhances the convergence of training WGANs.

8.3.1 High-level idea of gradient layer

Here, we explain gradient layer with intuitive motivation. It is inserted into the generator g in WGANs. We now focus on minimizing $\mathcal{L}(f, g)$ with respect to g under a fixed critic f, that is, we consider the problem $\min_g \mathcal{L}_f(g) \overset{\text{def}}{=} \mathbb{E}_{\mu_n}[-f(g(z))]$. Let us split g into two neural networks $g = g_1 \circ g_2$ at arbitrary layer where a new layer is to be inserted. Our purpose is to specify the form of layer ϕ that reduces the objective value by perturbations of inputs $g_2(z)$, i.e., $\mathcal{L}_f(g_1 \circ \phi \circ g_2) \leq \mathcal{L}_f(g)$. Since $\mathcal{L}_f(g_1 \circ \phi \circ g_2)$ is regarded as the integral $\mathbb{E}_{z' \sim g_{2\sharp}\mu_n}[-f(g_1(\phi(z')))]$ with respect to the push-forward distribution $g_{2\sharp}\mu_n$, this purpose is achieved by transporting the input distribution of ϕ along the gradient field $\nabla_{z'} f(g_1(z'))$. Therefore, we propose a gradient layer G_η with one hyperparameter $\eta > 0$ as a map that transforms an input z' to

$$G_\eta(z') = z' + \eta \nabla_{z'} f(g_1(z')). \tag{8.1}$$

Because the gradient layer depends on the parameters τ, θ of the upper layers f, g_1, we specify the parameter as $G_\eta^{\tau,\theta}$ if needed.

Applying the gradient layer recursively, it further progresses and achieves a better objective. The computation of the gradient layer is quite simple. Actually, simply taking the derivative is sufficient, which can be efficiently executed. Because too many gradient layers would lead to overfitting to the critic f, we stop stacking the gradient layer after an appropriate number of steps. Indeed, if $f \circ g_1$ is Lipschitz continuous, $id + \eta f \circ g_1$ for sufficiently small η is an injection because $(id + \eta f \circ g_1)(z) = (id + \eta f \circ g_1)(z')$ implies $\|z - z'\|_2 \leq \eta \mathcal{L}_{f \circ g_1} \|z - z'\|_2$ where $\mathcal{L}_{f \circ g_1}$ is the Lipschitz constant. Thus, a topology of $\mathrm{supp}(g_{2\sharp}\mu_n)$ is preserved and early stopping is justified. Then, this layer efficiently generates high-quality samples for the critic and the overall adversarial training procedure can be also boosted.

8.3.2 Algorithm description

The overall algorithm is described in this subsection. We adopt WGAN-GP as the base model to which gradient layer is applied. Let us denote by $R_{f_\tau}(\tilde{x})$ a gradient penalty term. In a paper on the improved WGANs (Gulrajani et al., 2017), the use of a two-sided penalty $(\|\nabla_{\tilde{x}} f_\tau(\tilde{x})\|_2 - 1)^2$ is recommended. However, we also allow the use of the one-sided variant $(\max(\|\nabla_{\tilde{x}} f_\tau(\tilde{x})\|_2 - 1, 0))^2$. As for the place in which the gradient layer is inserted, we can propose several possibilities, e.g., inserting the gradient layer into (i) the top and (ii) the bottom of the layers of the generator. The latter usage is described in the appendix.

The first usage is stacking gradient layers on the top of the generator, except for normalization to fine-tune the generator in the final phase. Although a normalization term such as $\tanh$ is commonly stacked on generators to bound the output range of the generators, gradient layers are typically applied before the normalization layer. Since $\tanh$ is a fixed function, it is no problem to combine $\tanh$ with critics by reinterpreting $\mathcal{F}$ and $\mathcal{X}$. The gradient layer directly handles the generated samples, so that it may significantly improve the sample quality. Because the gradient $\nabla_x f_\tau(x)$ of the critic with respect to data variables provides the direction to improve the quality of the current generated samples, it is expected that we can obtain better results by tracking the gradient iteratively. To compute the output from the gradient layer for a completely new input, we need to reproduce the computation of the gradient layers, which can be realized by saving the history of the parameters of critics and stacking the gradient layers using these parameters. The concrete procedure is described in Algorithm 16. When executing Algorithm 16, the parameter of g_θ is fixed, so that the push-forward measure $g_{\theta\sharp}\mu_n$ is treated as a base probability measure and we denote it by μ_g. Because the gradient layers depend on the history of the parameters in this case, we specify the parameter to be used: G_η^τ. For the parameter τ and the gradient v, we denote by $\mathcal{A}(\tau, v)$ one step of a gradient-based method such as SGD with momentum, Adam (Kingma and Ba, 2015), and RMSPROP (Tieleman and Hinton, 2012). From the optimization perspective, we show that Algorithm 16 can be regarded as an approximation to the functional gradient method. From this perspective, we show fast convergence of the method under appropriate assumptions where the objective function is smooth and the critics are optimized in each loop. This theoretical justification is described later. Although Algorithm 16 has a great optimization ability, applying the algorithm to large models is difficult because it requires the memory to register parameters; thus, we propose its usage for fine-tuning in the final phase of training a WGAN-GP. After the execution of Algorithm 16, we can generate samples by using the history of critics, the learning rate, and the base distribution as described in Algorithm 17.

Algorithm 16 Finetuning WGAN-GP

```
Input: The base distribution $\mu_g = g_\sharp \mu_n$, the minibatch size $b$, the number of iterations $T$, the initial parameters $\tau_0$ of the critic, the number of iterations $T_0$ for the critic, the regularization parameter $\lambda$, and the learning rate $\eta$ for gradient layers.
for $k = 0$ to $T - 1$ do
    $\tau \leftarrow \tau_k$
    for $k_0 = 0$ to $T_0 - 1$ do
        $\{x_i\}_{i=1}^b \sim \mu_D^b$, $\{z_i\}_{i=1}^b \sim \mu_g^b$, $\{\epsilon_i\}_{i=1}^b \sim U[0,1]^b$
        $\{z_i\}_{i=1}^b \leftarrow \{G_\eta^{\tau_k} \circ \cdots \circ G_\eta^{\tau_1}(z_i)\}_{i=1}^b$
        $\{\tilde{x}_i\}_{i=1}^b \leftarrow \{\epsilon_i x_i + (1-\epsilon_i) z_i\}_{i=1}^b$
        $v \leftarrow \nabla_\tau \frac{1}{b} \sum_{i=1}^b [f_\tau(z_i) - f_\tau(x_i) + \lambda R_{f_\tau}(\tilde{x}_i)])$
        $\tau \leftarrow \mathcal{A}(\tau, v)$
    end for
    $\tau_{k+1} \leftarrow \tau$
end for
Return $(\tau_1, \ldots, \tau_T)$.
```

Algorithm 17 Data Generation for Algorithm 16

```
Input: the seed drawn from base measure $z \sim \mu_g = g_\sharp \mu_n$, the history of parameters $\{\tau_k\}_{k=1}^T$, and the learning rate $\eta$ for gradient layers.
Return the sample $G_\eta^{\tau_T} \circ \cdots \circ G_\eta^{\tau_1}(z)$.
```

8.4 Functional Gradient Method

In this section, we provide mathematically rigorous derivation from the functional gradient method perspective under the Fréchet differentiable (functional differentiable) assumption on $\mathcal{L}$. That is, we consider an optimization problem with respect to a generator in an infinite-dimensional space. For simplicity, we focus on the case where the gradient layer is stacked on top of a generator g and we treat $g_\sharp \mu_n$ as the base measure μ_g. Thus, in the following we omit the notation g in $\mathcal{L}(f, \phi \circ g)$. Let $L^2(\mu_g)$ be the space of $L^2(\mu_g)$-integrable maps from $\mathbb{R}^v$ to $\mathbb{R}^v$, equipped with the $\langle \cdot, \cdot \rangle_{L^2(\mu_g)}$-inner product: for $\forall \phi_1, \forall \phi_2 \in L^2(\mu_g)$,

$$\langle \phi_1, \phi_2 \rangle_{L^2(\mu_g)} = \mathbb{E}_{\mu_g}[\phi_1(z)^\top \phi_2(z)].$$

To learn WGAN-GP, we consider the infinite-dimensional problem:

$$\min_{\phi \in L^2(\mu_g)} \max_{f_\tau \in \mathcal{F}} \mathcal{L}(f_\tau, \phi) - \lambda R_{f_\tau},$$

where R_{f_τ} is a gradient penalty term. To achieve this goal, we take a Gâteaux derivative along a given map $v \in L^2(\mu_g)$, i.e., a directional derivative along v. Let us denote

$\max_{f_\tau \in \mathcal{F}}\{\mathcal{L}(f_\tau, \phi) - \lambda R_f\}$ by $\mathcal{L}(\phi)$ and $\arg\max_{f_\tau \in \mathcal{F}}\{\mathcal{L}(f_\tau, \phi) - \lambda R_{f_\tau}\}$ by f_ϕ^* and the corresponding parameter by τ_ϕ^*, i.e., $f_\phi^* = f_{\tau_\phi^*}$ for $\phi \in L^2(\mu_g)$. If every $f \in \mathcal{F}$ is Lipschitz continuous and differentiable, we can find that by the envelope theorem and Lebesgue's convergence theorem this derivative takes the form:

$$\frac{d}{dt}\mathcal{L}(\phi + tv)\bigg|_{t=0} = -\mathbb{E}_{\mu_g}[\nabla_x f_\phi^*(x)|_{x=\phi(z)}^\top v(z)].$$

Therefore, $-\nabla_x f_\phi^*(x)|_{x=\phi(\cdot)}$ can be regarded as a Fréchet derivative (functional gradient) in $L^2(\mu_g)$ and we denote it by $\nabla_\phi \mathcal{L}(\phi)$, which performs like the usual gradient in Euclidean space. Using this notation, the optimization of $\mathcal{L}(\phi)$ can be accomplished by Algorithm 18, which is a gradient descent method in a function space. Because the functional gradient has the form $-\nabla_x f_\phi^* \circ \phi$, each iteration of the functional gradient method with respect to ϕ is $\phi \leftarrow \phi + \eta\nabla_x f_\phi^* \circ \phi = (id + \eta\nabla_x f_\phi^*) \circ \phi$, where η is the learning rate. We notice here that this iteration is the composition of a perturbation map $id + \eta\nabla_x f_\phi^*$ and a current map ϕ and is nothing but stacking a gradient layer $G_\eta^{\tau_*}$ on $\phi(z)$. In other words, the functional gradient method with respect to ϕ, i.e., Algorithm 18, is the procedure of building a deep neural network by inserting gradient layers, where the total number of iterations is the number of layers. Moreover, we notice that if we view $\nabla_x f_\phi^*$ as a perturbation term, this layer resembles that of *residual networks* (He et al., 2016) which is one of the state-of-the-art architectures in supervised learning tasks.

However, executing Algorithm 18 is difficult in practice because the exact optimization with respect to a critic f to compute $\mathcal{L}(\phi)$ is a hard problem. Thus, we need an approximation and we argue that Algorithm 16 is such a method. This point can be understood as follows. Roughly speaking, it maximizes $\mathcal{L}(f, \phi)$ with respect to f in the inner loop under fixed $\phi = G_\eta^{\tau_k} \circ G_\eta^{\tau_{k-1}} \circ \cdots \circ G_\eta^{\tau_1}$ to obtain an approximate solution τ_{k+1} to τ_* and minimizes that with respect to ϕ in the outer loop by stacking $G_\eta^{\tau_{k+1}}$, which is an approximation to $G_\eta^{\tau_*}$. Thus, Algorithm 16 is an approximated method, but we expect it to achieve fast convergence owing to the powerful optimization ability of the functional gradient method, as shown later. In particular, it is more effective to apply the algorithm in the final phase of training WGAN-GP to fine-tune it, because the optimization ability of parametric models are limited.

Algorithm 18 Functional Gradient Descent

```
Input: the initial generator $g$ and the learning rate $\eta$.
$\phi_0 \leftarrow g$
for $k = 0$ to $T - 1$ do
    $\phi_{k+1} \leftarrow \phi_k - \eta\nabla_\phi \mathcal{L}(\phi_k)$
end for
Return the function: $\phi_T$.
```

8.5 Convergence Analysis

Let us provide convergence analysis of Algorithm 18 for the problem of the general form: $\min_\phi \mathcal{L}(\phi)$. We note that the Algorithm 18 is the sames as that in Chapter 6. The convergence can be shown in an analogous way to that for the finite-dimensional one. To prove this, we make a smoothness assumption on the loss function. We now describe a definition of the smoothness on a Hilbert space whose counterpart in finite-dimensional space is often assumed for smooth nonconvex optimization methods.

Definition 8. *Let h be a function on a Hilbert space $(\mathcal{Z}, \langle, \rangle_{\mathcal{Z}})$. We call that h is L-smooth at z in U if h is differentiable at z and it follows that $\forall z' \in U$.*

$$|h(z') - h(z) - \langle \nabla_z h(z), z' - z\rangle_{\mathcal{Z}}| \leq \frac{L}{2}\|z' - z\|_{\mathcal{Z}}^2.$$

The following definition and proposition provide one condition leading to Lipschitz smoothness of $\mathcal{L}$. Let us denote by $\|\cdot\|_{L^\infty(\mu_g)}$ the sup-norm $\|\psi\|_{L^\infty(\mu_g)} = \sup_{\mathrm{supp}(\mu_g)}\|\psi(z)\|_2$ and by $B_r^\infty(\phi)$ a ball of center ϕ and radius r. Let $\hat{\mathcal{L}}(f,\psi) = \mathcal{L}(f,\psi) - \lambda R_f$. In the following we assume f_ψ^* is uniquely defined for $\psi \in L^2(\mu_g)$ and L-smoothness with respect to the input x.

Definition 9. *For positive values r and L, we call that $\mathcal{L}$ is (r, L)-regular at ϕ when the following condition is satisfied; For $\forall\psi \in B_r^\infty(\phi)$, $\hat{\mathcal{L}}(f_{\psi'}^*, \psi)$ is L-smooth at ψ with respect to ψ' in $B_r^\infty(\psi)$.*

Proposition 14. *If $\mathcal{L}$ is (r, L)-regular at ϕ, then $\mathcal{L}$ is $2L$-smooth at ϕ in $B_r^\infty(\psi)$.*

We now show the convergence of Algorithm 18. The following theorem gives the rate to converge to the stationary point.

Theorem 16. *Let us assume the norm of the gradient $\|\nabla_x f_\phi^*(x)\|_2$ is uniformly bounded by α and assume $\mathcal{L}$ is L-smooth at ϕ in $B_r^\infty(\phi)$ for $\forall\phi \in L^2(\mu_g)$. Suppose we run Algorithm 18 with constant learning rate $\eta \leq \min\{1/L, r/\alpha\}$. Then we have for $T \in \mathbb{Z}_+$*

$$\min_{k\in\{0,\ldots,T-1\}} \|\nabla_\phi \mathcal{L}(\phi_k)\|_{L^2(\mu_g)}^2 \leq \frac{2}{\eta T}(\mathcal{L}(\phi_0) - \mathcal{L}_*),$$

where $\mathcal{L}_* = \inf_\phi \mathcal{L}(\phi)$.

Note that the convergence rate $O(1/T)$ is the same as the gradient descent method for smooth objective in the finite-dimensional one. This means that even though the optimization is executed in the infinite-dimensional space, we do not suffer from the infinite dimensionality in terms of the convergence.

The following rough argument indicates that Algorithm 18 matches with learning WGANs. Let W_1 denote the 1-Wasserstein distance with respect to the Euclidean distance on a compact base space $\mathcal{X} \subset \mathbb{R}^v$. The following proposition is immediately shown by combining existing results (Ambrosio, 2003; Sudakov, 1979).

Proposition 15. *Let μ_g be a Borel probability measure on $\mathcal{X}$ and assume μ_g is absolutely continuous with respect to the Lebesgue measure. Then, there exists an optimal transport ψ and it follows that $W_1(\psi_{t\sharp}\mu_g, \mu_D) = (1-t)W_1(\mu_g, \mu_D)$, where $\psi_t = (1-t)id + t\psi$.*

The notion of the optimal transport is briefly introduced in Appendix. By this proposition, there exists a curve ψ_t strictly reducing distance, i.e., $dW_1(\psi_{t\sharp}\mu_g, \mu_D)/dt < 0$ if $\mu_g \neq \mu_D$. Because $\mathcal{L}$ approximates W_1, it is expected that $d\mathcal{L}(\psi_t)/dt < 0$ when μ_g differs from μ_D. Noting that $d\mathcal{L}(\psi_t)/dt = \langle \nabla_\phi \mathcal{L}(\psi_t), \psi - id \rangle_{L^2(\mu_g)}$, the functional gradient $\nabla_\phi \mathcal{L}(\psi_t) \neq 0$ does not vanish and the objective $\mathcal{L}$ may be strictly reduced by Algorithm 18.

8.6 Gradient Flow Perspective

In Euclidean space, the step of the steepest descent method for minimizing problems can be derived by the discretization of the gradient flow $d\gamma(t)/dt = -\nabla_x F(\gamma(t))$ where F is an objective function on Euclidean space. Because our goal is to move μ_g closer to μ_D, we should consider a gradient flow in the space of probability measures. To make this argument rigorously, we need the continuity equation that characterizes a curve of probability measures and the tangent space where velocities of curves should be contained (c.f., Ambrosio et al. (2008)). When these notions are provided, the gradient flow is defined immediately and it is quite natural to discretize this flow to track it well. In this section, we show that Algorithm 18 is such a natural discretization; in other words, building a deep neural network by stacking gradient layers is a discretization procedure of the gradient flow. We refer to Ambrosio et al. (2008) for detailed descriptions on this subject, and also refer to Otto (2001) for an original method developed by Otto.

8.6.1 Continuity Equation and Discretization

We denote by $\mathcal{P}$ the set of probability measures on $\mathbb{R}^v$. For $\mu \in \mathcal{P}$, let $\{\phi_t\}_{t\in[0,\delta]}$ be a curve in $L^2(\mu)$ that solves the following ordinary differential equation: for an $L^2(\phi_{t\sharp}\mu)$-integrable vector field v_t on $\mathbb{R}^v$,

$$\phi_0 = id, \ \ \frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \ for \ \forall x \in \mathbb{R}^v.$$

Then, this equation derives the curve $\nu_t = \phi_{t\sharp}\mu$ in $\mathcal{P}$, which can be characterized by .

$$\frac{d}{dt}\nu_t + \nabla \cdot (v_t \nu_t) = 0. \tag{8.2}$$

In other words, the following equation is satisfied

$$\int_I \int_{\mathbb{R}^v} (\partial_t f(x,t) + \nabla_x f(x,t)^\top v_t) d\nu_t dt = 0,$$

for $\forall f \in \mathcal{C}_c^\infty(\mathbb{R}^v \times I)$ where $\mathcal{C}_c^\infty(\mathbb{R}^v \times I)$ is the set of $\mathcal{C}^\infty$-functions with compact support in $\mathbb{R}^v \times I$. Conversely, a *narrowly* continuous family of probability measures ν_t solving equation (8.2) can be obtained by transport map ϕ_t satisfying $\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x))$ (Ambrosio et al., 2008). Thus, equation (8.2) indicates that v_t drifts the probability measures ν_t. Indeed, v_t can be recognized as the tangent vector of the curve ν_t as discussed below.

Here, we focus on curves in the subset $\mathcal{P}_2 \subset \mathcal{P}$ composed of probability measures with finite second moment. Noting that there is freedom in the choice of v_t modulo divergence-free vector fields $w \in L^2(\nu_t)$ (i.e., $\nabla \cdot (w\nu_t) = 0$), it is natural to consider the equivalence class of $v \in L^2(\nu_t)$ modulo divergence-free vector fields. Moreover, there exists a unique $\Pi(v)$ that attains the minimum $L^2(\nu_t)$-norm in this class: $\Pi(v) = \arg\min_{w\in L^2(\nu_t)}\{\|w\|_{L^2(\nu_t)} \mid \nabla\cdot((v-w)\nu_t) = 0\}$. Thus, we here introduce the definitions of the tangent space at $\mu \in \mathcal{P}_2$ as follows:

$$T_\mu\mathcal{P}_2 \overset{\text{def}}{=} \{\Pi(v) \mid v \in L^2(\mu)\}. \tag{8.3}$$

The following proposition shows that $T_\mu\mathcal{P}_2$ has the property of the tangent space on the space of probability measures, that is, a perturbation using $v_t \in T_\mu\mathcal{P}_2$ can discretize an absolutely continuous curve ν_t and v_t locally approximates *optimal transport maps*. We denote the 2-Wasserstein distance by W_2.

Proposition 16 (Ambrosio et al. (2008))**.** *Let $\nu_t : I \to \mathcal{P}_2$ be an absolutely continuous curve satisfying the continuity equation with a Borel vector field v_t that is contained in $T_{\nu_t}\mathcal{P}_2$ almost everywhere $t \in I$. Then, for almost everywhere $t \in I$ the following property holds:*

$$\lim_{\delta\to 0}\frac{W_2(\nu_{t+\delta}, (id+\delta v_t)_\sharp\nu_t)}{|\delta|} = 0.$$

In particular, for almost everywhere $t \in I$ such that ν_t is absolutely continuous with respect to the Lebesgue measure, we have

$$\lim_{\delta\to 0}\frac{1}{\delta}(\mathbf{t}_{\nu_t}^{\nu_{t+\delta}} - id) = v_t \quad in\ L^2(\nu_t),$$

where $\mathbf{t}_{\nu_t}^{\nu_{t+\delta}}$ is the unique optimal transport map between ν_t and $\nu_{t+\delta}$.

This proposition suggests the update $\mu^+ \leftarrow (id+v)_\sharp\mu$ for discretizing an absolutely continuous curve in $\mathcal{P}_2$. Note that when $\mu = \phi_\sharp\nu$, $(\nu \in \mathcal{P}_2, \phi \in L^2(\nu))$, the corresponding map to μ^+ is obtained by $\phi_\sharp^+\nu = \mu^+$ where ϕ^+ is a composition as follows:

$$\phi^+ \leftarrow (id+v)\circ\phi = \phi + v\circ\phi. \tag{8.4}$$

So far, we have introduced the property of continuous curves in $\mathcal{P}_2$ and a method of their discretization. We notice that the above update resembles the update of Algorithm 18. Indeed, we show that the functional gradient method is nothing but a discretization method of the gradient flow derived by the functional gradient $\nabla_\phi\mathcal{L}(\phi)$.

8.6.2 Discretization of Gradient Flow

We here introduce the gradient flow, which is one of the most straightforward ways to understand Algorithm 18. We have explained that an absolutely continuous curve $\{\nu_t\}_{t\in I}$ in $\mathcal{P}_2$ is well characterized by the continuity equation (8.2) and we have seen that $\{v_t\}_{t\in I}$ in (8.2) corresponds to the notion of the velocity field induced by the curve. Such a velocity points in the direction of the particle flow. Moreover, the functional gradient $\nabla_\phi \mathcal{L}(\phi)(\cdot)$ points in an opposite direction to reduce the objective $\mathcal{L}$ at each particle. Thus, these two vector fields exist in the same space and it is natural to consider the following equation:

$$v_t = -\nabla_\phi \mathcal{L}(\phi_t). \tag{8.5}$$

This equation for an absolutely continuous curve is called the gradient flow (Ambrosio et al., 2008) and a curve satisfying this will reduce the objective $\mathcal{L}$. Indeed, we can find by the chain rule such a curve $\{\nu_t = \phi_{t\sharp}\mu_g\}_{t\in I}$ that also satisfies the following:

$$\frac{d}{dt}\mathcal{L}(\phi_t) = -\|\nabla_\phi \mathcal{L}(\phi_t)\|^2_{L^2(\nu_t)}.$$

Recalling that ν_t can be discretized well by $\nu_{t+\delta} \sim (id - \delta\nabla_\phi \mathcal{L}(\phi_t))_\sharp \nu_t$, we notice that Algorithm 18 is a discretization method of the gradient flow (8.5). In other words, building deep neural networks by stacking gradient layers is such a discretization procedure.

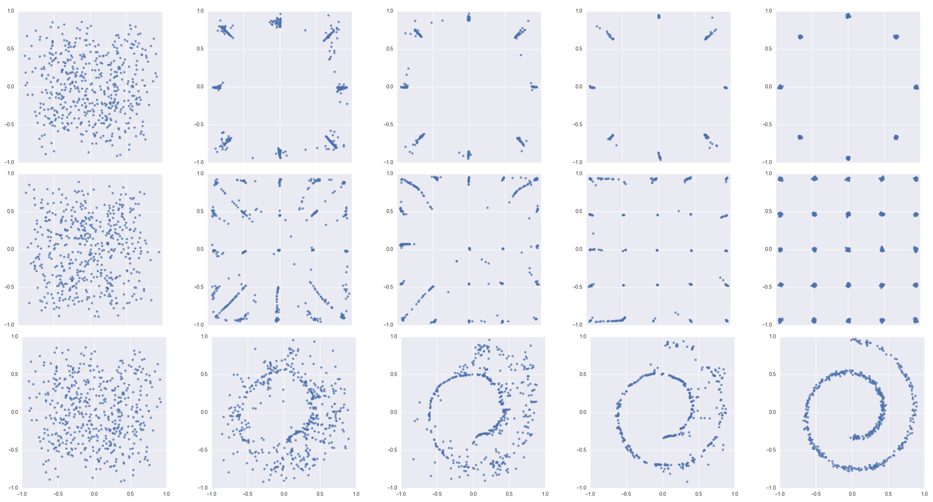

Figure 8.2: Generated samples by Algorithm 16 on 8-gaussian dataset for $0, 25, 50$, and 100 generator iterations (from left to right) and training data (rightmost).

8.7 Experiments

In this section, we show the powerful optimization ability of the gradient layer method empirically on training WGANs. Our implementation is done using Theano (Bergstra et al., 2010; Bastien et al., 2012). We first used three toy datasets: swiss roll, 8-gaussian, and 25-gaussian datasets (see Figure 8.2) to confirm the convergence behavior of the gradient layer. The sizes of toy datasets are 500, 500, and 1000, respectively. We next used the CIFAR-10 containing 50,000 images of size 32×32, and STL-10 containing 100,000 images. For STL-10 dataset, we downsample each dimension by 2, resulting image size is 48×48. We reported inception scores (Salimans et al., 2016) for image datasets, which is one of the conventional scores commonly used to measure the quality of generated samples.

Toy datasets We ran Algorithm 16 without pre-training of generators (i.e., $g = id$) on toy datasets from Gaussian noise distributions with the standard deviation 0.5. We used four-layer neural networks for the critics where the dimension of hidden layers were set to 128 for swiss roll and 8-gaussian datasets and 512 for 25-gaussian dataset. We adopted one-sided penalty with regularization parameter $\lambda = 10$. The output of generator was activated by $\tanh$. We used ADAM for training critics with parameters $\alpha = 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.9$, minibatch size $b = 50$. When we run Algorithm 16, gradient layers are stacked below $\tanh$. The learning rates were set to $\eta = 0.1$. The number of inner iterations T_0 for training the critics was set to $5 \times datasize/b$. Figure 8.2 shows the results for toy datasets for running $T = 100$ iterations of generators. Although we ran the algorithm without pre-training the generators, we obtained better results only for a few iterations. This is surprising, because these toy datasets are difficult to learn and fail to converge in the standard GANs and WGANs. Whereas improved variants of these models overcome this difficulty, they usually require more than 1,000 iterations to converge.

CIFAR-10 and STL-10 We first trained WGAN-GP with a two-sided penalty ($\lambda = 10$) on the CIFAR-10 and STL-10 datasets. We used DCGAN for both the critic and the generator. The batch normalization (Ioffe and Szegedy, 2015) was used only for the generator. The critic and the generator were trained by using ADAM with $\alpha = 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.9$, and minibatch size $b = 64$. The number of inner iterations for training the critics were 5 and we ran ADAM for 10^5-iterations. The left side of Figure 8.3 shows the inception scores obtained by WGAN-GP. It seems that the learning procedure is slowed down in a final training phase, especially for CIFAR-10. The final inception score on CIFAR-10 and STL-10 are 6.32 and 7.40, respectively. We next ran Algorithm 16 starting from the result of WGAN-GP. The critics were trained by ADAM with the same parameters, except for $\alpha = 5 \times 10^{-5}$ and $T_0 = datasize/b$. The learning rates were set to 0.5 for CIFAR-10 and 0.3 for STL-10. The right side of Figure 8.3 shows the inception

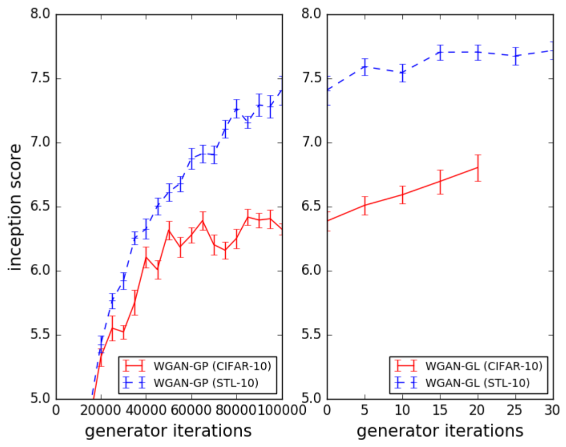

Figure 8.3: Left: Inception scores obtained by WGAN-GP, Right: Inception scores obtain by Algorithm 16 starting from the result of WGAN-GP.

scores obtained by Algorithm 16. Note that, since we focus on the optimization ability of generators, we plotted results with the horizontal axis as the number of outer-iterations. We observed a rapid increase in the inception scores, which were improved to 6.80 and 7.71 on CIFAR-10 and STL-10, respectively.

8.8 Appendix

8.8.1 The Other Usage

We introduce the usage that inserts a fixed number of gradient layers into the bottom of the generator to assist overall training procedure, which is described in Algorithm 19. Note that we always use latest parameters of f, g for gradient layers in Algorithm 19. When gradient layers are inserted in the middle of the generator: $g_1 \circ \phi \circ g_2$, we can apply Algorithm 19 by setting $\mu_n \leftarrow g_{2\sharp}\mu_n, g \leftarrow g_1$. After training, we can generate samples by using parameters of the critic and the generator, the learning rate, and the number of gradient layers, which is described in Algorithm20.

Algorithm 19 Assisting WGAN-GP

```
**Input:** The base distribution $\mu_n$, the minibatch size $b$, the number of iterations $T$, the
initial parameters $\tau_0$ and $\theta_0$ of the critic and the generator, the number of iterations
$T_0$ for the critic, the regularization parameter $c$, learning rate $\eta$ for gradient layers, the
number of gradient layers $l$.
for $k = 0$ to $T - 1$ do
    $\tau \leftarrow \tau_k$
    for $k_0 = 0$ to $T_0 - 1$ do
        $\{x_i\}_{i=1}^b \sim \mu_D^b$, $\{z_i\}_{i=1}^b \sim \mu_n^b$, $\{\epsilon_i\}_{i=1}^b \sim U[0,1]^b$
        # $G_\eta^{\tau_k,\theta_k}$ is applied $l$ times.
        $\{z_i\}_{i=1}^b \leftarrow \{g_{\theta_k} \circ G_\eta^{\tau_k,\theta_k} \circ \cdots \circ G_\eta^{\tau_k,\theta_k}(z_i)\}_{i=1}^b$
        $\{\tilde{x}_i\}_{i=1}^b \leftarrow \{\epsilon_i x_i + (1-\epsilon_i) z_i\}_{i=1}^b$
        $v = \nabla_\tau \frac{1}{b}\sum_{i=1}^b [f_\tau(z_i) - f_\tau(x_i) + \lambda R_{f_\tau}(\tilde{x}_i)])$
        $\tau \leftarrow \mathcal{A}(\tau, v)$
    end for
    $\tau_{k+1} \leftarrow \tau$
    $\{z_i\}_{i=1}^b \sim \mu_n^b$
    # $G_\eta^{\tau_{k+1},\theta_k}$ is applied $l$ times.
    $\{z_i\}_{i=1}^b \leftarrow \{G_\eta^{\tau_{k+1},\theta_k} \circ \cdots \circ G_\eta^{\tau_{k+1},\theta_k}\}_{i=1}^b$
    $v \leftarrow -\nabla_\theta \frac{1}{b}\sum_{i=1}^b f_{\tau_{k+1}}(g_{\theta_k}(z_i))$
    $\theta_{k+1} \leftarrow \mathcal{A}(\theta_k, v)$
end for
Return $\tau_T, \theta_T$.
```

We next briefly review Algorithm 19 in which a fixed number of gradient layers with latest parameters is inserted in the bottom of a generator of WGAN-GP. That is, gradient layers modify a noise distribution μ_n to improve the quality of a generator by the functional gradient method.

Algorithm 20 Data Generation for Algorithm 19

```
Input: the seed drawn from the base measure $z \sim \mu_n$, the parameter $\tau$ and $\theta$ of the critic and the generator, the learning rate $\eta$, the number of gradient layers $l$.
Apply gradient layers $l$ times $z' \leftarrow G_\eta^{\tau,\theta} \circ \cdots \circ G_\eta^{\tau,\theta}(z)$
Return the sample $g_\theta(z')$.
```

8.8.2 Brief Review of Wasserstein Distance

We introduce some facts concerning the Wasserstein distance, which is used for the proof of Proposition 15. We first describe a primal form of the Wasserstein distance. For $p \geq 1$ let $\mathcal{P}_p$ be the set of Borel probability measures with finite p-the moment on $\mathcal{X} \subset \mathbb{R}^v$. For $\mu, \nu \in \mathcal{P}_p$ a probability measure γ on $\mathcal{X} \times \mathcal{X}$ satisfying $\pi^1_\sharp \gamma = \mu$ and $\pi^2_\sharp \gamma = \nu$ is called a *plan* (*coupling*), where π^i denotes the projection from $\mathcal{X} \times \mathcal{X}$ to the i-th space $\mathcal{X}$. We denote by $\Gamma(\mu, \nu)$ the set of all plans between μ and ν. We now introduce Kantorovich's formulation of the p-Wasserstein distance W_p for $p \geq 1$.

$$W_p^p(\mu, \nu) = \min_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_2^p d\gamma(x, y) \tag{8.6}$$

When $p = 1$ and μ, ν have bounded supports, there is the Kantorovich-Rubinstein dual formulation of the 1-Wasserstein distance, which coincide with the definition introduced in the paper. The existence of optimal plans is guaranteed under more general integrand (c.f. Villani (2008); Ambrosio et al. (2008)) and we denote by Γ the set of optimal plans. Prior to this formulation, the optimal transport problem in Monge's formulation was proposed.

$$\inf_{\phi_\sharp \mu = \nu} \int_{\mathcal{X}} \|x - \phi(x)\|_2^p d\mu(x), \tag{8.7}$$

where the infimum is taken over all transport maps $\phi : \mathcal{X} \to \mathcal{X}$ from μ to ν, i.e., $\phi_\sharp \mu = \nu$. Because a transport map ϕ gives a plan $\gamma = (id \times \phi)_\sharp \mu$, we can easily find (8.6) $\leq$ (8.7). In general, an optimal transport map that solves the problem (8.7) does not always exist unlike Kantrovich problem (8.6). However, in the case where $p > 1$, $\mathcal{X} = \mathbb{R}^v$, and μ is absolutely continuous with respect to the Lebesgue measure, the existence of optimal transport maps is guaranteed (Brenier, 1987, 1991) and it is extended to more general integrand (see Ambrosio et al. (2008)). Moreover, this optimal transport map also solves Kantrovich problem (8.6), i.e., these two distances coincide. On the other hand, in the case $p = 1$, the existence of optimal transport maps is much more difficult, but it is shown in limited settings as follows.

Proposition 17 (Sudakov (Sudakov, 1979), see also (Ambrosio, 2003))**.** *Let $\mathcal{X}$ be a compact convex subset in $\mathbb{R}^v$ and assume that μ is absolutely continuous with respect to Lebesgue measure. Then, there exists an optimal transport map ϕ from μ to ν for the* *problem 8.7 with $p = 1$. Moreover, if ν is also absolutely continuous with respect to Lebesgue measure, we can choose ψ so that ψ^{-1} is well defined μ_0-a.e., and $\phi_\sharp^{-1}\nu = \mu$.*

Under the same assumption in Proposition 17, it is known that two distances (8.7) and (8.6) coincide (Ambrosio, 2003), that is, the Kantrovich problem (8.6) is solved by an optimal transport map.

8.8.3 Proofs

We here the give proof of Proposition 14.

Proof of Proposition 14. Note that $\mathcal{L}(\psi) = \hat{\mathcal{L}}(f^*_\psi, \psi)$. For $\psi \in B^\infty_r(\phi)$, we divide $\mathcal{L}(\psi)$ into two terms as follows.

$$\mathcal{L}(\psi) = (\hat{\mathcal{L}}(f^*_\psi, \psi) - \hat{\mathcal{L}}(f^*_\phi, \psi)) + \hat{\mathcal{L}}(f^*_\phi, \psi). \tag{8.8}$$

We first bound the first term in (8.8) by L-smoothness of $\hat{\mathcal{L}}(f^*_{\psi'}, \psi)$ with respect to ψ' at ψ in $B^\infty_r(\psi)$.

$$\left|\hat{\mathcal{L}}(f^*_\phi, \psi) - (\hat{\mathcal{L}}(f^*_\psi, \psi) + \left\langle \nabla_{\psi'}\hat{\mathcal{L}}(f^*_{\psi'}, \psi)\big|_{\psi'=\psi}, \phi - \psi \right\rangle_{L^2(\mu_g)})\right| \leq \frac{L}{2}\|\phi - \psi\|^2_{L^2(\mu_g)}.$$

Since $\hat{\mathcal{L}}(f^*_{\psi'}, \psi)$ attains the maximum, we have $\nabla_{\psi'}\hat{\mathcal{L}}(f^*_{\psi'}, \psi)\big|_{\psi'=\psi} = 0$ and have

$$\left|\hat{\mathcal{L}}(f^*_\phi, \psi) - \hat{\mathcal{L}}(f^*_\psi, \psi)\right| \leq \frac{L}{2}\|\phi - \psi\|^2_{L^2(\mu_g)}. \tag{8.9}$$

We next bound $\hat{\mathcal{L}}(f^*_\phi, \psi)$ in (8.8). We remember that

$$\hat{\mathcal{L}}(f^*_\phi, \psi) = \mathbb{E}_{x\sim\mu_D}[f^*_\phi(x)] - \mathbb{E}_{x\sim\mu_g}[f^*_\phi \circ \psi(x)] - \lambda R_{f^*_\phi}. \tag{8.10}$$

By L-smoothness of f^*_ϕ, it follows that

$$\left|f^*_\phi(\psi(x)) - (f^*_\phi(\phi(x)) + \left\langle \nabla_z f^*_\phi(z)\big|_{z=\phi(x)}, \psi(x) - \phi(x) \right\rangle_2)\right| \leq \frac{L}{2}\|\psi(x) - \phi(x)\|^2_2.$$

By taking the expectation with respect to $\mathbb{E}_{\mu_g}$, we get

$$\left|-\mathbb{E}_{x\sim\mu_g}[f^*_\phi \circ \psi(x)] + \mathbb{E}_{\mu_g}[f^*_\phi(\phi(x))] + \langle \nabla_z f^*_\phi \circ \phi, \psi - \phi \rangle_{L^2(\mu_g)}\right| \leq \frac{L}{2}\|\psi - \phi\|^2_{L^2(\mu_g)}.$$

We substitute this inequality into (8.10), we have

$$\hat{\mathcal{L}}(f^*_\phi, \psi) \leq \mathbb{E}_{x\sim\mu_D}[f^*_\phi(x)] + \frac{L}{2}\|\psi - \phi\|^2_{L^2(\mu_g)} - (\mathbb{E}_{\mu_g}[f^*_\phi(\phi(x))] + \langle \nabla_z f^*_\phi \circ \phi, \psi - \phi \rangle_{L^2(\mu_g)}) - \lambda R_{f^*_\phi}$$

$$
\begin{aligned}
&= \hat{\mathcal{L}}(f^*_\phi, \phi) - \left\langle \nabla_z f^*_\phi \circ \phi, \psi - \phi \right\rangle_{L^2(\mu_g)} + \frac{L}{2}\|\psi - \phi\|^2_{L^2(\mu_g)} \\
&= \mathcal{L}(\phi) + \langle \nabla_\phi \mathcal{L}(\phi), \psi - \phi \rangle_{L^2(\mu_g)} + \frac{L}{2}\|\psi - \phi\|^2_{L^2(\mu_g)}, \qquad (8.11)
\end{aligned}
$$

and the opposite inequality

$$
\hat{\mathcal{L}}(f^*_\phi, \psi) \geq \mathcal{L}(\phi) + \langle \nabla_\phi \mathcal{L}(\phi), \psi - \phi \rangle_{L^2(\mu_g)} - \frac{L}{2}\|\psi - \phi\|^2_{L^2(\mu_g)}, \qquad (8.12)
$$

where we used $\nabla_\phi \mathcal{L}(\phi) = -\nabla_z f^*_\phi(z)|_{z=\phi(\cdot)}$. By combining (8.8),(8.9), and (8.11), we have

$$
\mathcal{L}(\psi) \leq \mathcal{L}(\phi) + \langle \nabla_\phi \mathcal{L}(\phi), \psi - \phi \rangle_{L^2(\mu_g)} + L\|\phi - \psi\|^2_{L^2(\mu_g)}.
$$

Moreover, since $\hat{\mathcal{L}}(f^*_\psi, \psi) - \hat{\mathcal{L}}(f^*_\phi, \psi) \geq 0$ in (8.8), we have $\mathcal{L}(\psi) \geq \hat{\mathcal{L}}(f^*_\phi, \psi)$. Therefore, we get the opposite inequality by (8.12)

$$
\mathcal{L}(\psi) \geq \mathcal{L}(\phi) + \langle \nabla_\phi \mathcal{L}(\phi), \psi - \phi \rangle_{L^2(\mu_g)} - \frac{L}{2}\|\phi - \psi\|^2_{L^2(\mu_g)}.
$$

This finishes the proof. $\square$

We next provide the proof of Theorem 16.

Proof of Theorem 16. Noting that $\|\eta \nabla_{\phi_k} \mathcal{L}(\phi_k)\|_\infty \leq r$ and Lipschitz smoothness of $\mathcal{L}$, we have

$$
\begin{aligned}
\mathcal{L}(\phi_{k+1}) &\leq \mathcal{L}(\phi_k) - \eta\|\nabla_\phi \mathcal{L}(\phi_k)\|^2_{L^2(\mu_g)} + \frac{\eta^2 L}{2}\|\nabla_\phi \mathcal{L}(\phi_k)\|_{L^2(\mu_g)} \\
&= \mathcal{L}(\phi_k) - \eta(1 - \eta L/2)\|\nabla_\phi \mathcal{L}(\phi_k)\|^2_{L^2(\mu_g)}.
\end{aligned}
$$

Since $\eta \leq 1/L$, we have $\mathcal{L}(\phi_{k+1}) \leq \mathcal{L}(\phi_k) - \frac{\eta}{2}\|\nabla_\phi \mathcal{L}(\phi_k)\|^2_{L^2(\mu_g)}$. Summing up over $k \in \{0, \ldots, T-1\}$ and dividing by T we obtain

$$
\frac{1}{T}\sum_{k=0}^{T-1} \|\nabla_\phi \mathcal{L}(\phi_k)\|^2_{L^2(\mu_g)} \leq \frac{2}{\eta T}(\mathcal{L}(\phi_0) - \mathcal{L}(\phi_T)).
$$

This inequality finishes the proof of the theorem. $\square$

Proof of Proposition 15 . By Proposition 17, there exists an optimal transport ψ from μ_g to μ_D and an optimal plan is given by $\gamma = (id \times \psi)_\sharp \mu_g$. We set $\psi_t = (1-t)id + t\psi$ and $\mu_t = \psi_{t\sharp}\mu_g$. Because $(\psi_s, \psi_t)_\sharp \mu_g$ $(0 \leq s < t \leq 1)$ gives a plan between μ_g and μ_D, we have

$$
W_1(\mu_s, \mu_t) \leq \int_{\mathcal{X}\times\mathcal{X}} \|x - y\|_2 d(\psi_s, \psi_t)_\sharp \mu_g
$$

$$= \int_{\mathcal{X}} \|\psi_s(x) - \psi_t(x)\|_2 d\mu_g$$
$$= (t-s) \int_{\mathcal{X}} \|x - \psi(x)\|_2 d\mu_g = (t-s) W_1(\mu_g, \mu_D). \tag{8.13}$$

We next prove the opposite inequality. Noting that $(id, \psi_s)_\sharp \mu_g$ is a plan from μ_g to μ_s and $(\psi_t, \psi)_\sharp \mu_g$ is a plan from μ_t to μ_D, we have the following two inequalities

$$W_1(\mu_g, \mu_s) \leq \int_{\mathcal{X}\times\mathcal{X}} \|x-y\|_2 d(id, \psi_s)_\sharp \mu_g = \int_{\mathcal{X}} \|x - \psi_s(x)\|_2 d\mu_g = sW_1(\mu_g, \mu_D),$$
$$W_1(\mu_t, \mu_D) \leq \int_{\mathcal{X}\times\mathcal{X}} \|x-y\|_2 d(\psi_t, \psi)_\sharp \mu_g = \int_{\mathcal{X}} \|\psi_t(x) - \psi(x)\|_2 d\mu_g = (1-t)W_1(\mu_g, \mu_D).$$

Using these two inequalities and the triangle inequality, we get

$$W_1(\mu_g, \mu_D) \leq W_1(\mu_g, \mu_s) + W_1(\mu_s, \mu_t) + W_1(\mu_t, \mu_D) \leq (1+s-t)W_1(\mu_g, \mu_D) + W_1(\mu_s, \mu_t).$$

That is $(t-s)W_1(\mu_g, \mu_D) \leq W_1(\mu_s, \mu_t)$. By combining this inequality and (8.13), we have $(t-s)W_1(\mu_g, \mu_D) = W_1(\mu_s, \mu_t)$ and this finishes the proof. □

8.8.4 Labeled Faces in the Wild

We provide the result on the Labeled Faces in the Wild dataset. The result is depicted in Figure 8.4. After training WGAN-GP, we ran Algorithm 18 for a few iterations.

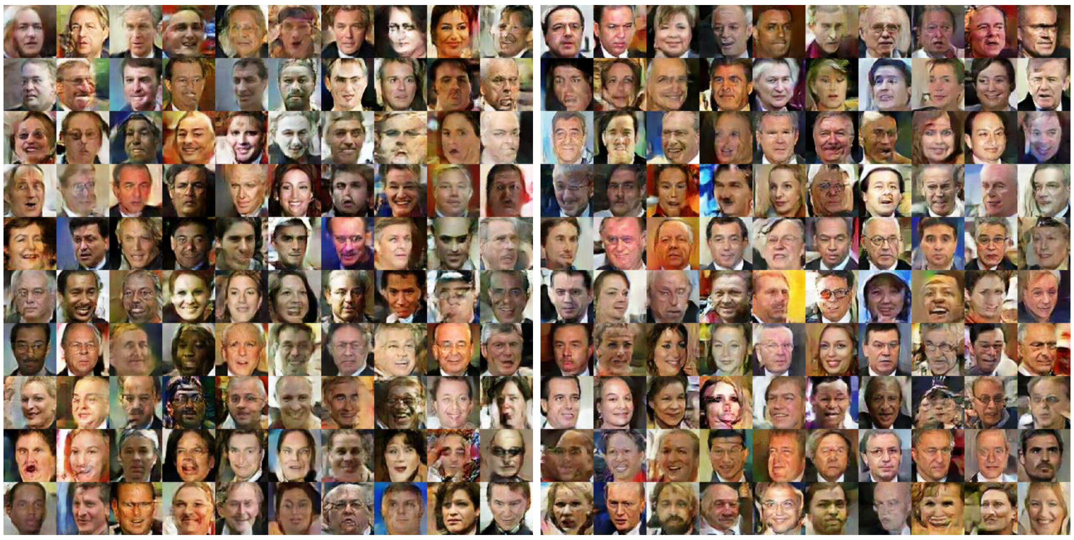

Figure 8.4: Random samples drawn from the generator trained by WGAN-GP (left) and the gradient layer (right).

Chapter 9

Functional gradient boosting based on residual network perception

Residual Networks (ResNets) have become state-of-the-art models in deep learning and several theoretical studies have been devoted to understanding why ResNet works so well. One attractive viewpoint on ResNet is that it is optimizing the risk in a functional space by combining an ensemble of effective features. In this chapter, we adopt this viewpoint to construct a new *gradient boosting method*, which is known to be very powerful in data analysis. To do so, we formalize the gradient boosting perspective of ResNet mathematically using the notion of functional gradients and propose a new method called *ResFGB* for classification tasks by leveraging ResNet perception. Two types of generalization guarantees are provided from the optimization perspective: one is the margin bound and the other is the expected risk bound by the sample-splitting technique. Experimental results show superior performance of the proposed method over state-of-the-art methods such as LightGBM.

This chapter is based on the work *Functional Gradient Boosting based on Residual Network Perception*, A. Nitanda and T. Suzuki, International Conference on Machine Learning, 2018 (Nitanda and Suzuki, 2018a).

9.1 Overview

Deep neural networks have achieved great success in classification tasks; in particular, residual network (ResNet) (He et al., 2016) and its variants such as wide-ResNet (Zagoruyko and Komodakis, 2016), ResNeXt (Xie et al., 2017), and DenseNet (Huang et al., 2017b) have become the most prominent architectures in computer vision. Thus, to reveal a factor in their success, several studies have explored the behavior of ResNets and some promising perceptions have been advocated. Concerning the behavior of ResNets, there are mainly two types of thoughts. One is the ensemble views, which were pointed out in Veit et al. (2016); Littwin and Wolf (2016). They presented that ResNets are ensemble of shallower models using an unraveled view of ResNets. Moreover, Veit et al. (2016) enhanced their claim by showing that dropping or shuffling residual blocks does not affect the performance of ResNets experimentally. The other is the optimization or ordinary differential equation views. In Jastrzebski et al. (2017), it was observed experimentally that ResNet layers iteratively move data representations along the negative gradient of the loss function with respect to hidden representations. Moreover, several studies (Weinan, 2017; Haber et al., 2017; Chang et al., 2017a,b; Lu et al., 2017) have pointed out that ResNet layers can be regarded as discretization steps of ordinary differential equations. Since optimization methods are constructed based on the discretization of gradient flows, these studies are closely related to each other.

On the other hand, gradient boosting (Mason et al., 1999; Friedman, 2001) is known to be a state-of-the-art method in data analysis; in particular, XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017) are notable because of their superior performance. Although ResNets and gradient boosting are prominent methods in different domains, we notice an interesting similarity by recalling that gradient boosting is an ensemble method based on iterative refinement by functional gradients for optimizing predictors. However, there is a key difference between ResNets and gradient boosting methods. While gradient boosting directly updates the predictor, ResNets iteratively optimize the feature extraction by stacking ResNet layers rather than the predictor, according to the existing work.

In this chapter, leveraging this observation, we propose a new gradient boosting method called *ResFGB* for classification tasks based on ResNet perception, that is, the feature extraction gradually grows by functional gradient methods in the space of feature extractions and the resulting predictor naturally forms a ResNet-type architecture. The expected benefit of the proposed method over usual gradient boosting methods is that functional gradients with respect to feature extraction can learn a deep model rather than a shallow model like usual gradient boosting. As a result, more efficient optimization is expected.

In the theoretical analysis of the proposed method, we first formalize the gradient boosting perspective of ResNet mathematically using the notion of functional gradients in the space of feature extractions. That is, we explain that optimization in that space is achieved by stacking ResNet layers. We next show a good consistency property of the functional gradient, which motivates us to find feature extraction with small functional gradient norms for estimating the correct label of data. This fact is very helpful from the optimization perspective because minimizing the gradient norm is much easier than minimizing the objective function without strong convexity. Moreover, we show the margin maximization property of the proposed method and derive the margin bound by utilizing this formalization and the standard complexity analysis techniques developed in Koltchinskii and Panchenko (2002); Bartlett and Mendelson (2002), which guarantee the generalization ability of the method. This bound gives theoretical justification for minimizing functional gradient norms in terms of both optimization and better generalization. Namely, we show that faster convergence of functional gradient norms leads to smaller classification errors. As for another generalization guarantee, we also provide convergence analysis of the sample-splitting variant of the method for the expected risk minimization. We finally show superior performance, empirically, of the proposed method over state-of-the-art methods including LightGBM.

Related work Several studies have attempted to grow neural networks sequentially based on the boosting theory. Bengio et al. (2006) introduced convex neural networks consisting of a single hidden layer, and proposed a gradient boosting-based method in which linear classifiers are incrementally added with their weights. However, the expressive power of the convex neural network is somewhat limited because the method cannot learn deep architectures. Moghimi et al. (2016) proposed boosted convolutional neural networks and showed superior empirical performance on fine-grained classification tasks, where convolutional neural networks are iteratively added, while our method constructs a deeper network by iteratively adding layers. Cortes et al. (2017) proposed AdaNet to adaptively learn both the structure of the network and its weight, and provided data-dependent generalization guarantees for an adaptively learned network; however, the learning strategy quite differs from our method and the convergence rate is unclear. The most related work is BoostResNet (Huang et al., 2017a), which constructs ResNet iteratively like our method; however, this method is based on an different theory rather than functional gradient boosting with a constant learning rate. This distinction makes the different optimization-generalization tradeoff. Indeed, our method exhibits a tradeoff with respect to the learning rate, which recalls perception of usual functional gradient boosting methods, namely a smaller learning rate leads to a good generalization performance.

9.2 Preliminary

In this section, we provide several notations and describe a problem setting of the classification. An important notion in this chapter is the functional gradient, which is also introduced in this section.

9.2.1 Problem setting

Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y}$ be a feature space and a finite label set of cardinal c, respectively. We denote by ν a true Borel probability measure on $\mathcal{X} \times \mathcal{Y}$ and by ν_n an empirical probability measure of samples $(x_i, y_i)_{i=1}^n$ independently drawn from ν, i.e., $d\nu_n(X,Y) = \sum_{i=1}^n \delta_{(x_i,y_i)}(X,Y)dXdY/n$, where δ denotes the Dirac delta function. We denote by ν_X the marginal distribution on X and by $\nu(\cdot|X)$ the conditional distribution on Y. We also denote empirical variants of these distributions by $\nu_{n,X}$ and $\nu_n(\cdot|X)$. In general, for a probability measure μ, we denote by $\mathbb{E}_\mu$ the expectation with respect to a random variable according to μ, by $L_2(\mu)$ the space of square-integrable functions with respect to μ, and by $L_2^q(\mu)$ the product space of $L_2(\mu)$ equipped with $\langle\cdot,\cdot\rangle_{L_2^q(\mu)}$-inner product: for $\forall\xi, \forall\zeta \in L_2^q(\mu)$,

$$\langle \xi, \zeta \rangle_{L_2^q(\mu)} \overset{\mathrm{def}}{=} \mathbb{E}_\mu[\xi(X)^\top \zeta(X)] = \mathbb{E}_\mu\left[\sum_{j=1}^{q} \xi_j(X)\zeta_j(X)\right].$$

We also use the following norm: for $\forall p \in (0,2]$ and $\forall \xi \in L_2^q(\mu)$, $\|\xi\|_{L_p^q(\mu)}^p \overset{\mathrm{def}}{=} \mathbb{E}_\mu[\|\xi(X)\|_2^p] = \mathbb{E}_\mu\left[(\sum_{j=1}^q \xi_j^2(X))^{p/2}\right]$.

The ultimate goal in classification problems is to find a predictor $f \in L_2^c(\nu_X)$ such that $\arg\max_{y\in\mathcal{Y}} f_y(x)$ correctly classifies its label. The quality of the predictor is measured by a loss function $l(\zeta, y) \geq 0$. A typical choice of l in multiclass classification problems is $l(\zeta, y) = -\log(\exp(\zeta_y)/\sum_{\overline{y}\in\mathcal{Y}} \exp(\zeta_{\overline{y}}))$, which is used for the multiclass logistic regression. The goal of classification is achieved by solving the expected risk minimization problem:

$$\min_{f\in L_2^c(\nu_X)} \left\{ \mathcal{L}(f) \overset{\mathrm{def}}{=} \mathbb{E}_\nu[l(f(X), Y)] \right\}. \tag{9.1}$$

However, the true probability measure ν is unknown, so we approximate $\mathcal{L}$ using the observed data probability measure ν_n and solve the empirical risk minimization problems:

$$\min_{f\in L_2^c(\nu_X)} \left\{ \mathcal{L}_n(f) \overset{\mathrm{def}}{=} \mathbb{E}_{\nu_n}[l(f(X), Y)] \right\}. \tag{9.2}$$

In general, some regularization is needed for the problem (9.2) to guarantee generalization. In this chapter, we rely on early stopping (Zhang and Yu, 2005) and some restriction on optimization methods for solving the problem.

Similar to neural networks, we split the predictor f into the feature extraction and linear predictor, that is, $f(x) = w^\top \phi(x)$, where $w \in \mathbb{R}^{d\times c}$ is a weight for the last layer and $\phi \in L_2^d(\nu_X)$ is a feature extraction from $\mathcal{X}$ to $\mathcal{X}$. For simplicity, we also denote $l(z, y, w) = l(w^\top z, y)$. Usually, ϕ is parameterized by a neural network and optimized using the stochastic gradient method. In this chapter, we propose a way to optimize ϕ in $L_2^d(\nu_X)$ via the following problem:

$$\min_{\substack{w\in\mathbb{R}^{d\times c}\\ \phi\in L_2^d(\nu_X)}} \left\{ \mathcal{R}(\phi, w) \overset{\mathrm{def}}{=} \mathbb{E}_\nu[l(\phi(X), Y, w)] + \frac{\lambda}{2}\|w\|_2^2 \right\} \tag{9.3}$$

where $\lambda > 0$ is a regularization parameter to stabilize the optimization procedure and $\|\cdot\|_2$ for w is a Euclidean norm. When we focus on the problem with respect to ϕ, we use the notation $\mathcal{R}(\phi) \overset{\mathrm{def}}{=} \min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}(\phi, w)$. We also denote by $\mathcal{R}_n(\phi, w)$ and $\mathcal{R}_n(\phi)$ empirical variants of $\mathcal{R}(\phi, w)$ and $\mathcal{R}(\phi)$, respectively, which are defined by replacing $\mathbb{E}_\nu$

by $\mathbb{E}_{\nu_n}$. In this chapter, we denote by ∂ the partial derivative and its subscript indicates the direction.

9.2.2 Functional gradient

The key notion used for solving the problem is the functional gradient in function spaces. Since they are taken in some function spaces, we first introduce Fréchet differential in general Hilbert spaces.

Definition 10. *Let $\mathcal{H}$ be a Hilbert space and h be a function on $\mathcal{H}$. For $\xi \in \mathcal{H}$, we call that h is Fréchet differentiable at ξ in $\mathcal{H}$ when there exists an element $\nabla_\xi h(\xi) \in \mathcal{H}$ such that*

$$h(\zeta) = h(\xi) + \langle \nabla_\xi h(\xi), \zeta - \xi \rangle_{\mathcal{H}} + o(\|\xi - \zeta\|_{\mathcal{H}}).$$

Moreover, for simplicity, we call $\nabla_\xi h(\xi)$ Fréchet differential or functional gradient.

We here make an assumption to guarantee Fréchet differentiability of $\mathcal{R}, \mathcal{R}_n$, which is valid for multiclass logistic loss: $l(z, y, w) = -\log(\exp(w_y^\top z)/\sum_{\overline{y}\in\mathcal{Y}} \exp(w_{\overline{y}}^\top z))$.

Assumption 8. *The loss function $l(\zeta, y) : \mathbb{R}^c \times \mathcal{Y} \to \mathbb{R}$ is a non-negative $\mathcal{C}^2$-convex function with respect to ζ and satisfies the following smoothness: There exists a positive real number A such that $\|\partial_\zeta^2 l(\zeta, y)\| \le A$ ($\forall (\zeta, y) \in \mathbb{R}^c \times \mathcal{Y}$), where $\|\cdot\|$ is the spectral norm.*

Note that under this assumption, the following bound holds:

$$\|\partial_z^2 l(z, y, w)\| \le Ar^2 \ for\ z \in \mathcal{X}, y \in \mathcal{Y}, w \in B_r(0),$$

where $B_r(0) \subset \mathbb{R}^{d\times c}$ is a closed ball of center 0 and radius r. After this, we set $A_r \overset{\text{def}}{=} Ar^2$ for simplicity.

For $\phi \in L_2^d(\nu_X)$, we set $w_\phi \overset{\text{def}}{=} \arg\min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}(\phi, w)$ and $w_{n,\phi} \overset{\text{def}}{=} \arg\min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}_n(\phi, w)$. Moreover, we define the following notations:

$$\nabla_\phi \mathcal{R}(\phi)(x) \overset{\text{def}}{=} \mathbb{E}_{\nu(Y|x)}[\partial_z l(\phi(x), Y, w_\phi)],$$
$$\nabla_\phi \mathcal{R}_n(\phi)(x) \overset{\text{def}}{=} \begin{cases} \partial_z l(\phi(x_i), y_i, w_{n,\phi}) & (x = x_i), \\ 0 & (\text{otherwise}). \end{cases}$$

We also similarly define functional gradients $\partial_\phi \mathcal{R}(\phi, w)$ and $\partial_\phi \mathcal{R}_n(\phi, w)$ for fixed w by replacing $w_\phi, w_{n,\phi}$ by w. It follows that

$$\nabla_\phi \mathcal{R}(\phi) = \partial_\phi \mathcal{R}(\phi, w_\phi), \nabla_\phi \mathcal{R}_n(\phi) = \partial_\phi \mathcal{R}_n(\phi, w_{n,\phi}).$$

The next proposition means that the above maps are functional gradients in $L_2^d(\nu_X)$ and $L_2^d(\nu_{n,X})$. We set $l_0 = \max_{y\in\mathcal{Y}} l(0, y)$.

Proposition 18. *Let Assumption 8 hold. Then, for* $\forall \phi, \psi \in L_2^d(\nu_X)$*, it follows that*

$$\mathcal{R}(\psi) = \mathcal{R}(\phi) + \langle \nabla_\phi \mathcal{R}(\phi), \psi - \phi \rangle_{L_2^d(\nu_X)} + H_\phi(\psi), \tag{9.4}$$

where $H_\phi(\psi) \leq \frac{A_{c_\lambda}}{2} \|\phi - \psi\|_{L_2^d(\nu_X)}^2$ $(c_\lambda = \sqrt{2l_0/\lambda})$*. Furthermore, the corresponding statements hold for* $\mathcal{R}(\cdot, w)$ $(\forall w \in \mathbb{R}^d)$ *by replacing* $\mathcal{R}(\cdot)$ *by* $\mathcal{R}(\cdot, w)$ *and for empirical variants by replacing* ν_X *by* $\nu_{n,X}$*.*

We can also show differentiability of $\mathcal{L}(f)$ and $\mathcal{L}_n(f)$. Their functional gradients have the form $\nabla_f \mathcal{L}(f)(x) = \mathbb{E}_{\nu(Y|x)}[\partial_\zeta l(f(x), Y)]$ and $\nabla_f \mathcal{L}_n(f)(x_i) = \partial_\zeta l(f(x_i), y_i)$. In this chapter, we derive functional gradient methods using $\nabla_\phi \mathcal{R}_n(\phi)$ rather than $\nabla_f \mathcal{L}_n(f)$ like usual gradient boosting (Mason et al., 1999; Friedman, 2001), and provide convergence analyses for problems (9.1) and (9.2). However, we cannot apply $\nabla_\phi \mathcal{R}_n(\phi)$ or $\partial_\phi \mathcal{R}_n(\phi, w)$ directly to the expected risk minimization problem because these functional gradients are zero outside the training data. Thus, we need a smoothing technique to propagate these to unseen data. The expected benefit of functional gradient methods using $\nabla_\phi \mathcal{R}_n(\phi)$ over usual gradient boosting is that the former can learn a deep model that is known to have high representational power. Before providing a concrete algorithm description, we first explain the basic property of functional gradients and functional gradient methods.

9.3 Basic Property of Functional Gradient

In this section, we explain the motivation for using functional gradients for solving classification problems. We first show the consistency of functional gradient norms, namely predicted probabilities by predictors with small norms converge to empirical/expected conditional probabilities. We next explain the superior performance of functional gradient methods intuitively, which motivate us to use it for finding predictors with small norms. Moreover, we explain that the optimization procedure of functional gradient methods can be realized by stacking ResNet layers iteratively on the top of feature extractions.

9.3.1 Consistency of functional gradient norm

We here provide upper bounds on the gaps between true empirical/expected conditional probabilities and predicted probabilities.

Proposition 19. *Let* $l(\zeta, y)$ *be the loss function for the multiclass logistic regression. Then,*

$$\|\nabla_f \mathcal{L}(f)\|_{L_1^c(\nu_X)} \geq \frac{1}{\sqrt{c}} \sum_{y \in \mathcal{Y}} \|\nu(y|\cdot) - p_f(y|\cdot)\|_{L_1(\nu_X)},$$

$$\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})} \geq \frac{1}{\sqrt{c}} \sum_{y \in \mathcal{Y}} \|\nu_n(y|\cdot) - p_f(y|\cdot)\|_{L_1(\nu_{n,X})},$$

where we denote by $p_f(y|x)$ the softmax function defined by the predictor f, i.e., $\exp(f_y(\cdot))/\sum_{\overline{y}\in\mathcal{Y}}\exp(f_{\overline{y}}(\cdot))$.

Many studies (Zhang, 2004; Steinwart, 2005; Bartlett et al., 2006) have exploited the consistency of convex loss functions for classification problems in terms of the classification error or conditional probability. Basically, these studies used the excess empirical/expected risk to estimate the excess classification error or the gap between the true conditional probability and the predicted probability. On the other hand, Proposition 19 argues that functional gradient norms give sufficient bounds on such gaps. This fact is very helpful from the optimization perspective for non-strongly convex smooth problems since the excess risk always bounds the functional gradient norm by the reasonable order, but the inverse relationship does not always hold. This means that finding a predictor with a small functional gradient is much easier than finding a small excess risk.

Note that the latter inequality in Proposition 19 provides the lower bound on empirical classification accuracy, which is confirmed by Markov inequality as follows.

$$\begin{aligned}\mathbb{P}_{\nu_n}[1 - p_f(Y|X) \geq 1/2] &\leq 2\mathbb{E}_{\nu_n}[1 - p_f(Y|X)] \\ &\leq 2\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.\end{aligned}$$

Generally, we can derive a bound on the empirical margin distribution (Koltchinskii and Panchenko, 2002) by using the functional gradient norm in a similar way, and can obtain a generalization bound using it, as shown later.

9.4 Algorithm Description

In this section, we provide concrete description of the proposed method. Let $\phi_t \in L_2^d(\nu_X)$ and w_t denote t-th iterates of ϕ and w. As mentioned above, since functional gradients $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ for the empirical risk vanish outside the training data, we need a smoothing technique to propagate these to unseen data. Hence, we use the convolution $T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ of the functional gradient by using an adaptively chosen kernel function k_t on $\mathcal{X}$. The convolution is applied element-wise as follows.

$$\begin{aligned}T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1}) &\overset{\mathrm{def}}{=} \mathbb{E}_{\nu_{n,X}}[\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(X)k_t(X,\cdot)] \\ &= \frac{1}{n}\sum_{i=1}^{n} \partial_z l(\phi_t(x_i), y_i, w_{t+1})k_t(x_i,\cdot).\end{aligned}$$

Namely, this quantity is a weighted sum of $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(x_i)$ by $k_t(x_i,\cdot)$, which we also call a functional gradient. In particular, we restrict the form of a kernel k_t to the inner-product of a non-linear feature embedding to a finite-dimensional space by $\iota_t : \mathbb{R}^d \to \mathbb{R}^D$, that is, $k_t(x, x') = \iota_t(\phi_t(x))^\top \iota_t(\phi_t(x))$. The requirements on the choice of ι_t to guarantee the convergence are the uniform boundedness and sufficiently preserving the magnitude of the functional gradient $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$. Let $\mathcal{F}$ be a given restricted class of bounded embeddings. We pick up ι_t from this class $\mathcal{F}$ by approximately solving the following problem to acquire magnitude preservation:

$$\max_{\iota_t \in \mathcal{F}} \|T_{k_t,n} \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2. \tag{9.5}$$

where we define $\|T_{k_t,n}\xi\|_{k_t}^2 = \langle \xi, T_{k_t,n}\xi \rangle_{L_2^d(\nu_{n,X})}$ for a vector function ξ. Detailed conditions on ι_t and an alternative problem to guarantee the convergence will be discussed later. Note that due to the restriction on the form of k_t, the computation of the functional gradient is compressed to the matrix-vector product. Namely,

$$\begin{aligned} A_t &\overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(x_i)\iota_t(\phi_t(x_i))^\top, \\ T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1}) &= A_t \iota_t(\phi_t(\cdot)). \end{aligned}$$

Therefore, the functional gradient method $\phi_{t+1} \leftarrow \phi_t - \eta_t T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ can be recognized as the procedure of successively stacking layers $id - \eta_t A_t \iota_t(\phi_t(\cdot))$ ($t \in \{0, \dots, T-1\}$) and obtaining a residual network. The entire algorithm is described in Algorithm 21. Note that because a loss function l is chosen typically to be convex with respect to w, a procedure in Algorithm 21 to obtain w_{n,ϕ_t} is easily achieved by running an efficient method for convex minimization problems. The notation T_0 is the stopping time of iterates with respect to w. That is, functional gradients $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ are computed at $w_{t+1} = w_{n,\phi_t}$ and correspond to $\nabla_\phi \mathcal{R}_n(\phi_t)$ when $t < T_0$ and computed at an older point of w when $t \geq T_0$, rather than $\nabla_\phi \mathcal{R}_n(\phi_t)$.

9.4.1 Choice of embedding

We here provide policies for the choice of ι_t. A sufficient condition for ι_t to achieve good convergence is to maintain the functional gradient norm, which is summarized below.

Assumption 9. *For positive values γ, ϵ, $p \leq 2$, q, and K, a function $k_t(x, x') = \iota_t(\phi_t(x))^\top \iota_t(\phi_t(x))$ satisfies $\|\iota_t(x)\|_2 \leq \sqrt{K}$ on $\mathcal{X}$, and $\gamma \|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_p^d(\nu_{n,X})}^q - \gamma\epsilon \leq \|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2$.*

This assumption is a counterpart of that imposed in Mason et al. (1999). The existence of ι_t, not necessarily included in $\mathcal{F}$, satisfying this assumption is confirmed as follows. We here assume that ϕ_t is a bijection that is a realistic assumption when learning rates are sufficiently small because of the inverse mapping theorem. Then, since $\nu_n(\cdot|X) =$

Algorithm 21 ResFGB

```
Input: $S = (x_i, y_i)_{i=1}^n$, initial points $\phi_0,\ w_0$, the number of iterations $T$ of $\phi$, the number of iterations $T_0$ of $w$, embedding class $\mathcal{F}$, and learning rates $\eta_t$
for $t = 0$ to $T-1$ do
  if $t < T_0$ then
    $w_{t+1} \leftarrow w_{n,\phi_t} = \arg\min_{w \in \mathbb{R}^{d\times c}} \mathcal{R}_n(\phi_t, w)$
  else
    $w_{t+1} \leftarrow w_t$
  end if
  Get $\iota_t$ by approximately solving (9.5) on $S$
  $A_t \leftarrow \frac{1}{n}\sum_{i=1}^n \partial_z l(\phi_t(x_i), y_i, w_{t+1})\iota_t(\phi_t(x_i))^\top$
  $\phi_{t+1} \leftarrow \phi_t - \eta_t A_t \iota_t(\phi_t(\cdot))$
end for
Return $\phi_{T-1}$ and $w_T$
```

$\nu_n(\cdot|\phi_t(X))$, functional gradients $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(x)$ become the map of $\phi_t(x)$, so we can choose ι_t such that

$$\iota_t(\phi_t(\cdot)) = \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)/\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)\|_2.$$

By simple computation, we find that $k_t(x, x') \leq 1$ and $\|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2$ are lower-bounded by $\frac{1}{d}\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})}^2$. A detailed derivation is provided in Appendix. Thus, Assumption 9 may be satisfied if an embedding class $\mathcal{F}$ is sufficiently large, but we note that too large $\mathcal{F}$ leads to overfitting. Therefore, one way of choosing ι_t is to approximate $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)/\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)\|_2$ rather than maximizing (9.5) directly, and indeed, this procedure has been adopted in experiments.

9.5 Convergence Analysis

In this section, we provide a convergence analysis for the proposed method. All proofs are included in Appendix. For the empirical risk minimization problem, we first show the global convergence rate, which also provides the generalization bound by combining the standard complexity analyses. Next, for the expected risk minimization problem, we describe how the size of $\mathcal{F}$ and the learning rate control the tradeoff between optimization speed and generalization by using the sample-splitting variant of Algorithm 21, whose detailed description will be provided later.

9.5.1 Empirical risk minimization

Using Proposition 18, Assumption 9, and an additional assumption on w_t, we can show the global convergence of Algorithm 21. The following inequality shows how functional gradients decrease the objective function, which is a direct consequence of Proposition 18. When $\eta \leq \frac{1}{A_{c_\lambda} K}$, we have

$$\begin{aligned}\mathcal{R}_n(\phi_{t+1}, w_{t+2}) \leq \mathcal{R}_n(\phi_t, w_{t+1})& \\ - \frac{\eta}{2}\|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2&.\end{aligned}$$

Therefore, Algorithm 21 provides a certain decrease in the objective function; moreover, we can conclude a stronger result.

Theorem 17. *Let Assumptions 8 and 9 hold. Consider running Algorithm 21 with a constant learning rate $\eta_t = \eta \leq \frac{1}{A_{c_\lambda} K}$. If $p \geq 1$ and the minimum eigenvalues of $(w_t{}^\top w_t)_{t=0}^{T_0}$ have a uniform lower bound $\sigma^2 > 0$, then*

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla_f \mathcal{L}_n(f_t)\|_{L_1^c(\nu_{n,X})}^q \leq \frac{2\mathcal{R}_n(\phi_0, w_1)}{\eta\gamma\sigma^q T} + \frac{\epsilon}{\sigma^q} \tag{9.6}$$

where we denote $f_t = w_{t+1}^\top \phi_t$.

Remark. (i) This theorem states the convergence of the average of functional gradient norms obtained by running Algorithm 21, but we note that it also leads to the convergence of the minimum functional gradient norms. (ii) Although a larger value of T_0 may affect the bound in Theorem 17 because of dependency on the minimum eigenvalue of $(w_t^\top w_t)_{t=0}^{T_0}$, optimizing w at each iteration facilitates the convergence speed empirically.

Theorem 17 means that the convergence becomes faster when an input distribution has the high degree of linear separability. However, even when it is somewhat large, a much faster convergence rate in the second half of the algorithm is achieved by making an additional assumption where loss function values attained by the algorithm are uniformly bounded.

Theorem 18. *Let Assumptions 8 and 9 with $(\epsilon, p, q) = (0, 1, 2)$ hold. We assume $T/2 \in \mathbb{N}$ for simplicity. Consider running Algorithm 21 with learning rates η_0 and η_1 in the first half and the second half of Algorithm, respectively. We assume $\eta_0, \eta_1 \leq \frac{\gamma}{A c_\lambda^2 K^2}$. We set $f_t = w_{t+1}^\top \phi_t$. Moreover, assume that there exists $\exists M > 0$ such that $l(f_t(X), Y) \leq M$ for $(X, Y) \sim \nu_{n,\mathcal{X}}$ and the minimum eigenvalues of $(w_t{}^\top w_t)_{t=0}^{T_0}$ have a uniform lower bound $\sigma^2 > 0$. Then we get*

$$\frac{1}{T}\sum_{t=0}^{\frac{T}{2}-1} \|\nabla_f \mathcal{L}_n(f_t)\|_{L_1^c(\nu_{n,X})}^2 \leq \frac{2\mathcal{L}_n(f_0)}{\eta_0\gamma\sigma^2 T},$$

$$\frac{1}{T}\sum_{t=\frac{T}{2}}^{T-1}\|\nabla_f \mathcal{L}_n(f_t)\|^2_{L_1^c(\nu_{n,X})} \leq \frac{4\mathcal{L}_n(f_0)}{\eta_1\gamma\sigma^2(2+\eta_0\alpha\mathcal{L}_n(f_0)T)T}.$$

9.5.2 Generalization bound

Here, we derive a generalization bound using the margin bound developed by Koltchinskii and Panchenko (2002), which is composed of the sum of the empirical margin distribution and Rademacher complexity of predictors. The margin and the empirical margin distribution for multiclass classification are defined as $m_f(x,y) \overset{\text{def}}{=} f_y(x) - \max_{y'\neq y} f_{y'}(x)$ and $\mathbb{P}_{\nu_n}[m_f(x,y) \leq \delta]$ ($\delta > 0$), respectively. When l is the multiclass logistic loss, using Markov inequality and Proposition 19, we can obtain an upper bound on the margin distribution:

$$\mathbb{P}_{\nu_n}[m_f(x,y) \leq \delta] \leq \left(1+\frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.$$

Since the convergence of functional gradient norms has been shown in Theorem 17 and 18, the resulting problem to derive a generalization bound is to estimate Rademacher complexity, which can be achieved using standard techniques developed by Bartlett and Mendelson (2002); Koltchinskii and Panchenko (2002). Thus, we specify here the architecture of predictors. In the theoretical analysis, we suppose $\mathcal{F}$ is the set of shallow neural networks $B\sigma(Cx)$ for simplicity, where B, C are weight matrices and σ is an element-wise activation function. Then, the t-th layer is represented as

$$\phi_{t+1}(x) = \phi_t(x) - D_t\sigma(C_t\phi_t(x)),$$

where $D_t = \eta_t A_t B_t$, and a predictor is $f_{T-1}(x) = w_T^\top \phi_{T-1}(x)$. Bounding norms of these weights by controlling the size of $\mathcal{F}$ and λ, we can restrict the Rademacher complexity of a set of predictors and obtain a generalization bound. We denote by $\mathcal{G}_{T-1}$ the set of predictors under constraints on weight matrices where L_1-norms of each row of $w_T^\top, C_t$, and D_t are bounded by Λ_w, Λ, and Λ'_t.

$$\begin{aligned}\mathcal{G}_{T-1} \overset{\text{def}}{=} \{&\|(w_T)_{*,y}\|_1 \leq \Lambda_w,\ \|(C_t)_{i,*}\|_1 \leq \Lambda, \\ &\|(D_t)_{j,*}\|_1 \leq \Lambda'_t,\ t\in\{0,\ldots,T-1\},\ \forall y, \forall i, \forall j\}.\end{aligned}$$

Theorem 19. *Let l be the multiclass logistic regression loss. Fix $\delta > 0$. Suppose σ is L_σ-Lipschitz continuous and $\|x\|_2 \leq \Lambda_\infty$ on $\mathcal{X}$. Then, for $\forall\rho > 0$, with probability at least $1-\rho$ over the random choice of S from ν^n, we have $\forall f \in \mathcal{G}_{T-1}$,*

$$\begin{aligned}\mathbb{P}_\nu[m_f(X,Y) \leq 0] \leq{}& \frac{2c^3\Lambda_\infty\Lambda_w}{\delta\sqrt{n}}\prod_{t=0}^{T-2}(1+\Lambda\Lambda'_t L_\sigma) \\ &+ \sqrt{\frac{\log(1/\rho)}{2n}} + \left(1+\frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f\mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.\end{aligned}$$

Combining Theorems 17, 18 and 19, we observe that the learning rates η_t, the number of iterations T, and the size of $\mathcal{F}$ have an impact on the optimization-generalization tradeoff, that is, larger values of these quantities facilitate the convergence on training data while the generalization bound becomes gradually loose. Especially, this bound has an exponential dependence on depth T, which is known to be unavoidable (Neyshabur et al., 2015) in the worst case for some networks with L_1 or the group norm constraints, but this bound is useful when an initial objective is small and required T is also small sufficiently.

We next derive an interesting bound for explaining the effectiveness of the proposed method. This bound can be obtained by instantiating bounds in Theorem 19 for various T, Λ'_t and making an union bound. Since norms of rows of A_t are uniformly bounded by their construction, norm constraints on $D_t = \eta_t A_t B_t$ is reduced to bounding a norm of B_t. Thus, we further assume $\sum_l \|(B_t)_{*,l}\|_2 \leq \Lambda''$.

Corollary 7. *Let l be the multiclass logistic regression loss. Fix $\delta > 0$. Suppose σ is L_σ-Lipschitz continuous and $\|x\|_2 \leq \Lambda_\infty$ on $\mathcal{X}$. Then, for $\forall \rho > 0$, with probability at least $1-\rho$ over the random choice of S from ν^n, the following bound is valid for any function f_{T-1} obtained by Algorithm 21 under constraints $\|(w_T)_{*,y}\|_1 \leq \Lambda_w$, $\sum_l \|(B_t)_{*,l}\|_2 \leq \Lambda''$, and $\|\iota_t(x)\|_2 \leq \sqrt{K}$.*

$$\begin{aligned}\mathbb{P}_\nu[m_{f_{T-1}}(X,Y) \leq 0] \leq{}& \frac{2c^3\Lambda L_\sigma \Lambda_\infty \Lambda_w}{\delta\sqrt{n}} \\ &+ \frac{c^3\Lambda_\infty\Lambda_w}{\delta\sqrt{n}}\left(1 + \frac{C}{T-1}\sum_{t=0}^{T-2}\eta_t\|\nabla_f \mathcal{L}_n(f_t)\|_{L_1^c(\nu_{n,X})}\right)^{T-1} \\ &+ \sqrt{\frac{1}{2n}\left(\log\left(\frac{1}{\rho}\right) + O(T\log T)\right)} \\ &+ \left(1 + \frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f \mathcal{L}_n(f_{T-1})\|_{L_1^c(\nu_{n,X})},\end{aligned}$$

where $f_t = w_{t+1}^\top \phi_t$ and $C = 2\Lambda L_\sigma \sqrt{Kd}\Lambda'' c_\lambda$.

This corollary shows an interesting and useful property of our method in terms of generalization, that is, fast convergence of functional gradient norms leads to small complexity of an obtained network, surprisingly. As a result, our method is expected to get a network with good generalization because it directly minimizes functional gradient norms.

By plugging in convergence rates of functional gradient norms in Theorem 17 and 18 for the generalization bound in Corollary 7, we can obtain explicit convergence rates of classification errors. For instance, under the assumption in Theorem 17 with $q = 2$, $\epsilon = 0$, and a learning rate $\eta = O(1/T^\alpha)$ $0 \leq \alpha < 1$, then the generalization bound becomes

$$O\left(\frac{1}{\sqrt{n}}\left(\exp(T^{\frac{1-\alpha}{2}}) + \sqrt{\log\frac{1}{\rho}}\right) + \frac{\mathcal{R}_n(\phi_0)}{T^{\frac{1-\alpha}{2}}}\right).$$

Moreover, under the assumption in Theorem 18 with learning rates $\eta_0 = O(1/T^{\alpha})$ and $\eta_1 = O(1/T^{2\alpha-1})$ $\frac{1}{2} \leq \alpha < 1$, a faster convergence rate is achieved.

$$O\left(\frac{1}{\sqrt{n}}\left(\exp(T^{\frac{1-\alpha}{2}}) + \sqrt{\log\frac{1}{\rho}}\right) + \frac{1}{T^{\frac{3(1-\alpha)}{2}}}\right).$$

Note that by utilizing the corollary, the optimization and generalization tradeoff depending on the number of iterations and learning rates is confirmed more clearly.

We note another type of bound can be derived by utilizing VC-dimension or pseudo-dimension (Vapnik and Chervonenkis, 1971). When the activation function is piece-wise linear, such as Relu function $\sigma(x) = \max\{0, x\}$, reasonable bounds on these quantities are given by Bartlett et al. (1998, 2017). Thus, for that case, we can obtain better bounds with respect to T by combining our analysis and the VC bound, but we omit the precise description for simplicity. We next show the other generalization guarantee from the optimization perspective by using the modified algorithm, which may slow down the optimization speed but alleviates the exponential dependence on T in the generalization bound.

9.5.3 Sample-splitting technique

To remedy the exponential dependence on T of the generalization bound, we introduce the sample-splitting technique which has been used recently to provide statistical guarantee of expectation-maximization algorithms (Balakrishnan et al., 2017; Wang et al., 2015). That is, instead of Algorithm 21, we analyze its sample-splitting variant. Although Algorithm 21 exhibits good empirical performance, the sample-splitting variant is useful for analyzing the behavior of the expected risk. In this variant, the entire dataset is split into T pieces, where T is the number of iterations, and each iteration uses a fresh batch of samples. The key benefit of the sample-splitting method is that it allows us to use concentration inequalities independently at each iterate ϕ_t rather than using the complexity measure of the entire model. As a result, sample-splitting alleviates the exponential dependence on T presented in Theorem 19. We now present the details in Algorithm 22. For simplicity, we assume $T_0 = 0$, namely the weight vector w_t is fixed to the initial weight w_0.

Our proof mainly relies on bounding a statistical error of the functional gradient at each iteration in Algorithm 22. Because the population version of Algorithm 21 strictly decreases the value of $\mathcal{R}$ due to its smoothness, we can show that Algorithm 22 also decreases it with high probability when the norm of a functional gradient is larger than a statistical error bound. Thus, we make here an additional assumption on the loss function to bound the statistical error, which is satisfied for a multiclass logistic loss function.

Algorithm 22 Sample-splitting ResFGB

```
Input: $S = (x_i, y_i)_{i=1}^n$, initial points $\phi_0,\ w_0$, the number of iterations $T$, embedding class $\mathcal{F}$, and learning rates $\eta$
Split $S$ into $T$ disjoint subsets $S_1, \ldots, S_T$ of size $\lfloor n/T \rfloor$
for $t = 0$ to $T-1$ do
    Define $\mathcal{R}_{\lfloor n/T \rfloor}(\phi_t, w)$ using $S_t$
    Get $\iota_t$ by approximately solving (9.5) on $S_t$
    $A_t \leftarrow \lfloor \frac{T}{n} \rfloor \sum_{i=1}^{\lfloor n/T \rfloor} \partial_z l(\phi_t(x_i), y_i, w_0)\iota_t(\phi_t(x_i))^\top$
    $\phi_{t+1} \leftarrow \phi_t - \eta A_t \iota_t(\phi_t(\cdot))$
end for
Return $\phi_{T-1}$ and $w_0$
```

Assumption 10. *For the differentiable loss function $l(z, y, w)$ with respect to z, w, there exists a positive real number β_r depending on $r > 0$ such that $\|\partial_z l(z, y, w)\|_2 \leq \beta_r$ for $z \in \mathcal{X}, y \in \mathcal{Y}, w \in B_r(0)$.*

We here introduce the notation required to describe the statement. We let $\mathcal{F}^j$ be a collection of j-th elements of functions in $\mathcal{F}$. For a positive value M, we set

$$\epsilon(m, \rho) \stackrel{\text{def}}{=} \beta_{\|w_0\|_2} \sqrt{\frac{KdD}{m}} \left(2M + \sqrt{2K \log \frac{2dD}{\rho}} \right).$$

The following proposition is a key result to bound a statistical error as mentioned above.

Proposition 20. *Let Assumption 10 hold and each $\mathcal{F}^j$ be the VC-class (for the definition see van der Vaart and Wellner (1996)). We assume $\|\iota_t(x)\|_2 \leq \sqrt{K}$ on $\mathcal{X}$. We set μ to be ν_X or $\nu_{m,X}$ and $k(x, x')$ to be $\iota(\phi(x))^\top \iota(\phi(x'))$. Then, there exists a positive value M depending on $\mathcal{F}$ and it follows that with probability at least $1 - \rho$ over the choice of the sample of size m, $\epsilon(m, \rho)$ upper-bounds the following.*

$$\sup_{\iota \in \mathcal{F}} \|T_k \partial_\phi \mathcal{R}(\phi, w_0) - T_{k,m} \partial_\phi \mathcal{R}_m(\phi, w_0)\|_{L_2^d(\mu)}.$$

Since each iterate in Algorithm 22 is computed on a fresh batch not depending on previous batches, Proposition 20 can be applied to all iterates with $m \leftarrow \lfloor n/T \rfloor$ and $\rho \leftarrow \delta/T$ for $\delta \in (0, 1)$. Thus, when $\lfloor n/T \rfloor$ is large and η is small sufficiently, functional gradients used in Algorithm 22 become good approximation to the population variant, and we find that the expected risk function is likely to decrease from Proposition 18. Moreover, we note that statistical errors are accumulated additively rather than the exponential growth. Concretely, we obtain the following generalization guarantee.

Theorem 20. *Let Assumptions 8, 9, and 10 and the same assumption in Proposition 20 hold. Consider running Algorithm 22. If* $p \geq 1$, $\|\partial_\zeta l(\zeta, y)\|_2 \leq B$, *and the minimum eigenvalue of* ${w_0}^\top w_0$ *is lower-bounded by* $\sigma^2 > 0$, *then we get with probability at least* $1 - \rho$,

$$\|\nabla_f \mathcal{L}(w_0^\top \phi_{t_*})\|_{L_1^c(\nu_X)} \leq B\left(\frac{2T}{n}\log\frac{T}{\rho}\right)^{\frac{1}{4}} + \sqrt{\frac{B}{\gamma^{\frac{1}{q}}\sigma}}$$

$$\cdot \left\{\frac{\mathcal{R}_0}{\eta T} + \beta_{\|w_0\|_2}\epsilon\left(\frac{n}{T}, \frac{\rho}{T}\right) + \frac{\eta}{2}A_{\|w_0\|_2}K^2\beta_{\|w_0\|_2}^2 + \gamma\epsilon\right\}^{\frac{1}{2q}}$$

where $\mathcal{R}_0 = \mathcal{R}(w_0, \phi_0)$ *and* t_* *is the index giving the minimum value of* $\|\nabla_f \mathcal{L}_{\lfloor n/T \rfloor}(w_0^\top \phi_t)\|_{L_p^c(\nu_{\lfloor n_T \rfloor, X})}$.

According to this theorem, η, T, and $\mathcal{F}$ control the optimization-generalization trade-off like Theorem 19.

Table 9.1: Test classification accuracy on binary and multiclass classification.

METHOD	LETTER	USPS	IJCNN1	MNIST	COVTYPE	SUSY
RESFGB (LOGISTIC)	**0.976** **(0.0019)**	**0.953** **(0.0007)**	**0.989** **(0.0004)**	0.986 (0.0007)	0.966 (0.0004)	**0.804** **(0.0000)**
RESFGB (SMOOTH HINGE)	0.975 (0.0014)	0.952 (0.0023)	**0.989** **(0.0005)**	**0.987** **(0.0010)**	0.965 (0.0058)	**0.804** **(0.0004)**
MULTILAYER PERCEPTRON	0.971 (0.0059)	0.948 (0.0045)	0.988 (0.0010)	0.986 (0.0010)	0.965 (0.0015)	**0.804** (**0.0004**)
SUPPORT VECTOR MACHINE	0.959 (0.0062)	0.948 (0.0023)	0.977 (0.0015)	0.969 (0.0041)	0.824 (0.0059)	0.754 (0.0534)
RANDOM FOREST	0.964 (0.0012)	0.939 (0.0018)	0.980 (0.0005)	0.972 (0.0005)	0.948 (0.0005)	0.802 (0.0004)
GRADIENT BOOSTING	0.964 (0.0011)	0.938 (0.0039)	0.982 (0.0010)	0.981 (0.0004)	**0.972** **(0.0005)**	**0.804** **(0.0005)**

9.6 Experiments

In this section, we present experimental results of the binary and multiclass classification tasks. We run Algorithm 21 and compare it with support vector machine, random forest, and gradient boosting methods. We here introduce settings used for Algorithm 21. As for the loss function, we test both multiclass logistic loss and smooth hinge loss, and as for the embedding class $\mathcal{F}$, we use three or four hidden-layer neural networks. The number of hidden units in each layer is set to $100, 200$, or 1000. Linear classifiers and embeddings are trained by Nesterov's momentum method. The learning rate is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$. These parameters and the number of iterations T are tuned based on the performance on the validation set.

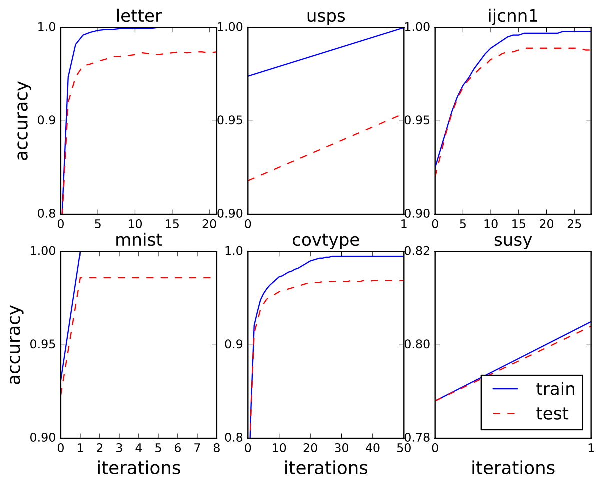

Figure 9.1: Learning curves for Algorithm 21 with multiclass logistic loss on libsvm datasets showing classification accuracy on training and test sets versus the number of iterations.

We use the following benchmark datasets: letter, usps, ijcnn1, mnist, covtype, and susy. We now explain the experimental procedure. For datasets not providing a fixed test set, we first divide each dataset randomly into two parts: 80% for training and the rest for test. We next divide each training set randomly and use 80% for training and the rest for validation. We perform each method on the training dataset with several hyperparameter settings and choose the best setting on the validation dataset. Finally, we train each model on the entire training dataset using this setting and evaluate it on the test dataset. This procedure is run 5 times.

The mean classification accuracy and the standard deviation are listed in Table 9.1. The support vector machine is performed using a random Fourier feature (Rahimi and Recht, 2007) with an embedding dimension of 10^3 or 10^4. For multilayer perceptron, we use three, four, or five hidden layers and rectified linear unit as the activation function. The number of hidden units in each layer is set to 100 or 1000. As for random forest, the number of trees is set to 100, 500, or 1000 and the maximum depth is set to 10, 20, or 30. Gradient boosting in Table 9.1 indicates LightGBM (Ke et al., 2017) with the hyperparameter settings: the maximum number of estimators is 1000, the learning rate is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$, and number of leaves in one tree is chosen from $\{16, 32, \ldots, 1024\}$.

As seen in Table 9.1, our method shows superior performance over the competitors except for covtype. However, the method that achieves higher accuracy than our method is only LightGBM on covtype.

We plot learning curves for one run of Algorithm 21 with logistic loss, which depicts classification accuracies on training and test sets. Note that the number of iterations are determined by classification results on validation sets. This figure shows the efficiency of the proposed method.

In this section, we introduce auxiliary lemmas used in our analysis. The first one is Hoeffding's inequality.

Lemma 14 (Hoeffding's inequality)**.** *Let $Z_1, \ldots, Z_s$ be i.i.d. random variables to $[-a, a]$ for $a > 0$. Denote by A_s the sample average $\sum_{i=1}^{s} Z_i/s$. Then, for any $\epsilon > 0$, we get*

$$\mathbb{P}[A_s + \epsilon \le \mathbb{E}[A_s]] \le \exp\left(-\frac{\epsilon^2 s}{2a^2}\right).$$

Note that this statement can be reinterpreted as follows: it follows that for $\delta \in (0, 1)$ with probability at least $1 - \delta$

$$A_s + a\sqrt{\frac{2}{s}\log\frac{1}{\delta}} \ge \mathbb{E}[A_s].$$

We next introduce the uniform bound by Rademacher complexity. For a set $\mathcal{G}$ of functions from $\mathcal{Z}$ to $[-a, a]$ and a dataset $S = \{z_i\}_{i=1}^{s} \subset \mathcal{Z}$, we denote empirical Rademacher complexity by $\hat{\Re}_S(\mathcal{G})$ and denote Rademacher complexity by $\Re_s(\mathcal{G})$; let $\sigma = (\sigma_i)_{i=1}^{s}$ be i.i.d random variables taking -1 or 1 with equal probability and let S be distributed according to a distribution μ^s,

$$\hat{\Re}_S(\mathcal{G}) = \mathbb{E}_\sigma\left[\sup_{f \in \mathcal{G}} \frac{1}{s}\sum_{i=1}^{s} \sigma_i f(x_i)\right], \quad \Re_s(\mathcal{G}) = \mathbb{E}_{\mu^s}[\hat{\Re}_S(\mathcal{G})].$$

Lemma 15. *Let $Z_1, \ldots, Z_s$ be i.i.d random variables to $\mathcal{Z}$. Denote by $A_s(f)$ the sample average $\sum_{i=1}^{s} f(Z_i)/s$. Then, for any $\delta \in (0, 1)$, we get with probability at least $1 - \delta$ over the choice of S,*

$$\sup_{f \in \mathcal{G}} |A_s(f) - \mathbb{E}[A_s(f)]| \le 2\Re_s(\mathcal{G}) + a\sqrt{\frac{2}{s}\log\frac{2}{\delta}}.$$

When a function class is VC-class (for the definite see van der Vaart and Wellner (1996)), its Rademacher complexity is uniformly bounded as in the following lemma which can be easily shown by Dudley's integral bound (Dudley, 1999) and the bound on the covering number by VC-dimension (pseudo-dimension) (van der Vaart and Wellner, 1996).

Lemma 16. *Let $\mathcal{G}$ be VC-class. Then, there exists positive value M depending on $\mathcal{G}$ such that $\Re_s(\mathcal{G}) \le M/\sqrt{m}$.*

The following lemma is useful in estimating Rademacher complexity.

Lemma 17. *(i) Let $h_i : \mathbb{R} \to \mathbb{R}$ $(i \in \{1, \dots, s\})$ be L-Lipschitz functions. Then it follows that*

$$\mathbb{E}_\sigma \left[\sup_{f \in \mathcal{G}} \sum_{i=1}^{s} \sigma_i h_i \circ f(x_i) \right] \le L\mathbb{E}_\sigma \left[\sup_{f \in \mathcal{G}} \sum_{i=1}^{s} \sigma_i \circ f(x_i) \right].$$

(ii) We denote by $\mathrm{conv}(\mathcal{G})$ *the convex hull of* $\mathcal{G}$*. Then, we have* $\hat{\mathfrak{R}}_S(\mathrm{conv}(\mathcal{G})) = \hat{\mathfrak{R}}_S(\mathcal{G})$*.*

The following lemma gives the generalization bound by the margin distribution, which is originally derived by Koltchinskii and Panchenko (2002). Let $\mathcal{G}$ be the set of predictors; $\mathcal{G} \subset \{f : \mathcal{X} \to \mathbb{R}^c\}$ and denote $\Pi\mathcal{G} = \{f_y(\cdot) : \mathcal{X} \to | \ f \in \mathcal{G}, y \in \mathcal{Y}\}$, then the following holds.

Lemma 18. *Fix $\delta > 0$. Then, for $\forall \rho > 0$, with probability at least $1 - \rho$ over the random choice of S from ν^n, we have $\forall f \in \mathcal{G}$,*

$$\mathbb{P}_\nu[m_f(X,Y) \le 0] \le \mathbb{P}_{\nu_n}[m_f(X,Y) \le \delta] + \frac{2c^2}{\delta}\mathfrak{R}_n(\Pi\mathcal{G}) + \sqrt{\frac{1}{2n}\log\frac{1}{\rho}}.$$

9.7 Proofs

In this section, we provide missing proofs in the paper.

9.7.1 Proofs of Section 9.3 and 9.4

We first prove Proposition 18 that states Lipschitz smoothness of the risk function.

Proof of Proposition 18 . Because $l(z,y,w)$ is $\mathcal{C}^2$-function with respect to z, w, there exist semi-positive definite matrices $A^{\phi,\psi}_{x,y}, B^{\phi,\psi}_{x,y}$ such that

$$\begin{aligned} l(\psi(x), y, w_\phi) &= l(\phi(x), y, w_\phi) + \partial_z l(\phi(x), y, w_\phi)^\top (\psi(x) - \phi(x)) \\ &\quad + \frac{1}{2}(\psi(x) - \phi(x))^\top A^{\phi,\psi}_{x,y}(\psi(x) - \phi(x)), \qquad (9.7) \\ l(\psi(x), y, w_\phi) + \frac{\lambda}{2}\|w_\phi\|_2^2 &= l(\psi(x), y, w_\psi) + \frac{\lambda}{2}\|w_\psi\|_2^2 \\ &\quad + (\partial_w l(\psi(x), y, w_\psi) + \lambda w_\psi)^\top (w_\phi - w_\psi) \\ &\quad + \frac{1}{2}(w_\phi - w_\psi)^\top B^{\phi,\psi}_{x,y}(w_\phi - w_\psi). \qquad (9.8) \end{aligned}$$

Note that we regard w_ϕ and w_ψ are flattened into column vectors if necessary. By Assumption 8, we find spectral norms of $A^{\phi,\psi}_{x,y}$ is uniformly bounded with respect to

x, y, ϕ, ψ, hence eigen-values are also uniformly bounded. In particular, since $\frac{\lambda}{2}\|w_\phi\|_2^2 \leq \mathcal{R}(\phi, w_\phi) \leq \mathcal{R}(\phi, 0) \leq l_0$, we see $-A_{c_\lambda} I \preceq A_{x,y}^{\phi,\psi} \preceq A_{c_\lambda} I$.

By taking the expectation $\mathbb{E}_\nu$ of the equality (9.7), we get

$$\mathcal{R}(\psi, w_\phi) = \mathcal{R}(\phi, w_\phi) + \langle \nabla_\phi \mathcal{R}(\phi), \psi - \phi \rangle_{L_2^d(\nu_X)} + \frac{1}{2}\mathbb{E}_\nu[(\psi(x) - \phi(x))^\top A_{x,y}^{\phi,\psi}(\psi(x) - \phi(x))] \tag{9.9}$$

and by taking the expectation $\mathbb{E}_\nu$ of the equality (9.8), we get

$$\mathcal{R}(\psi, w_\phi) = \mathcal{R}(\psi, w_\psi) + \frac{1}{2}(w_\phi - w_\psi)^\top \mathbb{E}_\nu[B_{x,y}^{\phi,\psi}](w_\phi - w_\psi), \tag{9.10}$$

where we used $\partial_w \mathcal{R}(\psi, w_\psi) = 0$. By combining equalities (9.9) and (9.10), we have

$$\mathcal{R}(\psi) = \mathcal{R}(\phi) + \langle \nabla_\phi \mathcal{R}(\phi), \psi - \phi \rangle_{L_2^d(\nu_X)} + H_\phi(\psi),$$

where

$$H_\phi(\psi) = \frac{1}{2}\mathbb{E}_\nu[(\psi(x) - \phi(x))^\top A_{x,y}^{\phi,\psi}(\psi(x) - \phi(x))] - \frac{1}{2}(w_\phi - w_\psi)^\top \mathbb{E}_\nu[B_{x,y}^{\phi,\psi}](w_\phi - w_\psi).$$

By the uniformly boundedness of $A_{x,y}^{\phi,\psi}$ and the semi-positivity of $B_{x,y}^{\phi,\psi}$, we find $H_\phi(\psi) \leq \frac{A_{c_\lambda}}{2}\|\phi - \psi\|_{L_2^d(\nu_X)}^2$.

The other cases can be shown in the same manner, thus, we finish the proof. □

We next show the consistency of functional gradient norms.

Proof of Proposition 19. We now prove the first inequality. Note that the integrand of y'-th element of $\nabla_f \mathcal{L}(f)(x)$ for multiclass logistic loss can be written as

$$\partial_{\zeta_{y'}} l(f(x), y) = -\mathbf{1}[y = y'] + \frac{\exp(f_{y'}(x))}{\sum_{\overline{y} \in \mathcal{Y}} \exp(f_{\overline{y}}(x))}.$$

Therefore, we get

$$\begin{aligned}
\|\nabla_f \mathcal{L}(f)\|_{L_1^c(\nu_X)} &= \mathbb{E}_{\nu_X}\|\nabla_f \mathcal{L}(f)(X)\|_2 \\
&= \mathbb{E}_{\nu_X}\|\mathbb{E}_{\nu(Y|X)}[\partial_\zeta(f(X), Y)]\|_2 \\
&= \mathbb{E}_{\nu_X}\left[\sqrt{\sum_{y' \in \mathcal{Y}} (\mathbb{E}_{\nu(Y|X)}[\partial_{\zeta_{y'}}(f(X), Y)])^2}\right] \\
&\geq \frac{1}{\sqrt{c}} \sum_{y' \in \mathcal{Y}} \mathbb{E}_{\nu_X}\left[\left|\mathbb{E}_{\nu(Y|X)}[\partial_{\zeta_{y'}}(f(X), Y)]\right|\right]
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\sqrt{c}} \sum_{y' \in \mathcal{Y}} \mathbb{E}_{\nu_X} \left[\left| \nu(y'|X) \left(-1 + \frac{\exp(f_{y'}(X))}{\sum_{\overline{y} \in \mathcal{Y}} \exp(f_{\overline{y}}(X))} \right) \right.\right. \\
&\left.\left. + \sum_{y \neq y'} \nu(y|X) \frac{\exp(f_{y'}(X))}{\sum_{\overline{y} \in \mathcal{Y}} \exp(f_{\overline{y}}(X))} \right| \right] \\
&= \frac{1}{\sqrt{c}} \sum_{y' \in \mathcal{Y}} \mathbb{E}_{\nu_X} \left[\left| \nu(y'|X) \left(-1 + \frac{\exp(f_{y'}(X))}{\sum_{\overline{y} \in \mathcal{Y}} \exp(f_{\overline{y}}(X))} \right) \right.\right. \\
&\left.\left. + (1 - \nu(y'|X)) \frac{\exp(f_{y'}(X))}{\sum_{\overline{y} \in \mathcal{Y}} \exp(f_{\overline{y}}(X))} \right| \right] \\
&= \frac{1}{\sqrt{c}} \sum_{y' \in \mathcal{Y}} \mathbb{E}_{\nu_X} \left[\left| -\nu(y'|X) + \frac{\exp(f_{y'}(X))}{\sum_{\overline{y} \in \mathcal{Y}} \exp(f_{\overline{y}}(X))} \right| \right] \\
&= \frac{1}{\sqrt{c}} \sum_{y' \in \mathcal{Y}} \| - \nu(y'|\cdot) + p_f(y'|\cdot) \|_{L_1(\nu_X)},
\end{aligned}
$$

where for the first inequality we used $(\sum_{i=1}^{c} a_i)^2 \leq c \sum_{i=1}^{c} a_i^2$. Noting that the second inequality in Proposition 19 can be shown in the same way by replacing ν by ν_n, we finish the proof. $\square$

We here give the proof of the following inequality concerning choice of embedding introduced in section 9.4.

$$
\|T_{k_t,n} \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2 \geq \frac{1}{d} \|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})}^2 \tag{9.11}
$$

Proof of (9.11) . For notational simplicity, we denote by $G_t = \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)$ and by G_t^i the i-the element of G_t. Then, we get

$$
\begin{aligned}
\|T_{k_t,n}(G_t)\|_2)\|_{k_t}^2 &= \langle G_t, T_{k_t,n} G_t \rangle_{L_2^d(\nu_{n,X})} \\
&= \mathbb{E}_{(X,X') \sim \nu_{n,X}^2} [G_t(X)^\top G_t(X') G_t(X')^\top G_t(X) / (\|G_t(X)\|_2 \|G_t(X')\|_2)] \\
&= \sum_{i,j=1}^{d} (\mathbb{E}_{\nu_{n,X}} [G_t^i(X) G_t^j(X) / \|G_t(X)\|_2])^2 \\
&\geq \sum_{i=1}^{d} (\mathbb{E}_{\nu_{n,X}} [G_t^i(X)^2 / \|G_t(X)\|_2])^2 \\
&\geq \frac{1}{d} \mathbb{E}_{\nu_{n,X}} [\|G_t(X))\|_2]^2 = \frac{1}{d} \|G_t\|_{L_1^d(\nu_{n,X})}^2,
\end{aligned}
$$

where we used $(\sum_{i=1}^{c} a_i)^2 \leq c \sum_{i=1}^{c} a_i^2$. $\square$

9.7.2 Empirical risk minimization and generalization bound

In this section, we give the proof of convergence of Algorithm 21 for the empirical risk minimization. We here briefly introduce the kernel function that provides useful bound in our analysis. A kernel function k is a symmetric function $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for arbitrary $s \in \mathbb{N}$ and points $\forall (x_i)_{i=1}^s$, a matrix $(k(x_i, x_j))_{i,j=1}^s$ is positive semi-definite. This kernel defines a reproducing kernel Hilbert space $\mathcal{H}_k$ of functions on $\mathcal{X}$, which has two characteristic properties: (i) for $\forall x \in \mathcal{X}$, a function $k(x, \cdot) : \mathcal{X} \to \mathbb{R}$ is an element of $\mathcal{H}_k$, (ii) for $\forall f \in \mathcal{H}_k$ and $\forall x \in \mathcal{X}$, $f(x) = \langle f, k(x, \cdot)\rangle_{\mathcal{H}_k}$, where $\langle , \rangle_{\mathcal{H}_k}$ is the inner-product in $\mathcal{H}_k$. These properties are very important and the latter one is called reproducing property. We extend the inner-product into the product space $\mathcal{H}_k^d$ in a straightforward way, i.e., $\langle f, g\rangle_{\mathcal{H}_k^d} = \sum_{i=1}^d \langle f^i, g^i\rangle_{\mathcal{H}_k}$.

The following proposition is useful in our analysis. The first property mean that the notation $\|T_{k_t,n}\nabla\mathcal{R}_n(\phi_t)\|_{k_t}$ provided in the paper is nothing but the norm of $T_{k_t,n}\nabla\mathcal{R}_n(\phi_t)$ by the inner-product $\langle , \rangle_{\mathcal{H}_{k_t}^d}$.

Proposition 21. *For a kernel function k, the following hold.*

- $\langle f, g\rangle_{L_2(\nu_X)} = \langle T_k f, g\rangle_{\mathcal{H}_k^d}$ *for* $f \in L_2^d(\nu_X)$, $g \in \mathcal{H}_k^d$ *where* $T_k f = \mathbb{E}_{\nu_X}[f(X)k(X,\cdot)]$,
  $\langle f, g\rangle_{L_2(\nu_{n,X})} = \langle T_{k,n} f, g\rangle_{\mathcal{H}_k^d}$ *for* $f \in L_2^d(\nu_{n,X})$, $g \in \mathcal{H}_k^d$ *where* $T_{k,n} f = \mathbb{E}_{\nu_{n,X}}[f(X)k(X,\cdot)]$,
- $\|f\|_{L_2(\nu_X)}^2 \le \mathbb{E}_{\nu_X}[k(X,X)]\|f\|_{\mathcal{H}_k^d}^2$ *for* $f \in \mathcal{H}_k^d$,
  $\|f\|_{L_2(\nu_{n,X})}^2 \le \mathbb{E}_{\nu_{n,X}}[k(X,X)]\|f\|_{\mathcal{H}_k^d}^2$ *for* $f \in \mathcal{H}_k^d$.

Proof. We show only the case of ν_X because we can prove the other case in the same manner. For $f \in L_2(\nu_X), g \in \mathcal{H}_k^d$, we get the first property by using reproducing property,

$$\langle f, g\rangle_{L_2(\nu_X)} = \mathbb{E}_{\nu_X}[f(X)^\top \langle g, k(X,\cdot)\rangle_{\mathcal{H}_k^d}] = \langle g, T_k f\rangle_{\mathcal{H}_k^d}.$$

We next show the second property as follows. For $\forall f \in \mathcal{H}_k^d$, we get

$$\begin{aligned}
\|f\|_{L_2(\nu_X)}^2 &= \mathbb{E}_{\nu_X}\|f(X)\|_2^2 \\
&= \mathbb{E}_{\nu_X}\| \langle f(\cdot), k(X,\cdot)\rangle_{\mathcal{H}_k^d} \|_2^2 \\
&\le \mathbb{E}_{\nu_X}\|k(X,\cdot)\|_{\mathcal{H}_k}^2 \|f\|_{\mathcal{H}_k^d}^2 \\
&= \mathbb{E}_{\nu_X}[k(X,X)]\|f\|_{\mathcal{H}_k^d}^2.
\end{aligned}$$

□

We give the proof of Theorem 17 concerning the convergence of functional gradient norms.

Proof of Theorem 17 . When $\eta \leq \frac{1}{A_{c_\lambda} K}$, we have from Proposition 18 and Proposition 21,

$$\mathcal{R}_n(\phi_{t+1}, w_{t+2}) \leq \mathcal{R}_n(\phi_t, w_{t+1}) - \frac{\eta}{2} \|T_{k_t,n} \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2.$$

By Summing this inequality over $t \in \{0, \ldots, T-1\}$ and dividing by T, we get

$$\frac{1}{T} \sum_{t=0}^{T-1} \|T_{k_t,n} \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2 \leq \frac{2}{\eta T} \mathcal{R}_n(\phi_0, w_1), \tag{9.12}$$

where we used $\mathcal{R}_n \geq 0$.

On the other hand, since $\partial_z l(z, y, w) = \partial_z l(w^\top z, y) = w \partial_\zeta l(w^\top z, y)$, it follows that

$$\begin{aligned} \partial_\phi \mathcal{R}_n(\phi, w)(x) &= \mathbb{E}_{\nu_n(Y|x)}[\partial_z l(\phi(x), y, w)] \\ &= \mathbb{E}_{\nu_n(Y|x)}[w \partial_\zeta l(w^\top \phi(x), y)] \\ &= w \nabla_f \mathcal{L}_n(w^\top \phi)(x). \end{aligned}$$

Thus, by the assumption on $({w_t}^\top w_t)_{t=0}^{T_0}$, we get for $t \in \{0, \ldots, T-1\}$

$$\begin{aligned} \|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_p^d(\nu_{n,X})} &= \mathbb{E}_{\nu_{n,X}}[\|w_{t+1} \nabla_f \mathcal{L}_n(w_{t+1}^\top \phi_t)(X)\|_2^p]^{1/p} \\ &\geq \sigma \mathbb{E}_{\nu_{n,X}}[\|\nabla_f \mathcal{L}_n(w_{t+1}^\top \phi_t)(X)\|_2^p]^{1/p} \\ &= \sigma \|\nabla_f \mathcal{L}_n(w_{t+1}^\top \phi_t)\|_{L_p^c(\nu_{n,X})}. \end{aligned} \tag{9.13}$$

Combining inequalities (9.12) (9.13) and Assumption 9, we get

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_f \mathcal{L}_n(w_{t+1}^\top \phi_t)\|_{L_p^c(\nu_{n,X})}^q \leq \frac{2}{\eta \gamma \sigma^q T} \mathcal{R}_n(\phi_0, w_1) + \frac{\epsilon}{\sigma^q}.$$

Since $p \geq 1$, we observe $\|\nabla_f \mathcal{L}_n(w_{t+1}^\top \phi_t)\|_{L_1^c(\nu_{n,X})} \leq \|\nabla_f \mathcal{L}_n(w_{t+1}^\top \phi_t)\|_{L_p^c(\nu_{n,X})}$ and we finish the proof. $\square$

To provide the proof of Theorem 18, we here give an useful proposition to show fast convergence rate for the multiclass logistic regression.

Proposition 22. *Let $l(\zeta, y)$ be the loss function for the multiclass logistic regression. Let $M > 0$ be arbitrary constant. For a predictor f, we assume $l(f(X), Y) \leq M$ for $(X, Y) \sim \nu_n$. Then, we have*

$$\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})} \geq \frac{1 - \exp(-M)}{\sqrt{c} M} \mathcal{L}_n(f).$$

Proof. Since $\exp(-t) \leq 1 - \frac{1-\exp(-M)}{M}t$ for $\forall t \in [-M, 0)$, we get

$$\mathbb{E}_{\nu_n}[\exp(-l(f(X), Y))] \leq 1 - \frac{1-\exp(-M)}{M}\mathcal{L}_n(f).$$

Using $l(f(X), Y) = -\log p_f(Y|X)$ and the above inequality with Proposition 19, we obtain

$$\begin{aligned}
\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})} &\geq \frac{1}{\sqrt{c}}\sum_{y\in\mathcal{Y}} \|\nu_n(y|\cdot) - p_f(y|\cdot)\|_{L_1(\nu_{n,X})} \\
&= \frac{1}{\sqrt{c}}\sum_{y\in\mathcal{Y}} \mathbb{E}_{\nu_{n,X}}|\nu_n(y|X) - p_f(y|X)| \\
&\geq \frac{1}{\sqrt{c}}\mathbb{E}_{\nu_n}[1 - p_f(Y|X)] \\
&= \frac{1}{\sqrt{c}}\mathbb{E}_{\nu_n}[1 - \exp(-l(f(X), Y))] \\
&\geq \frac{1-\exp(-M)}{\sqrt{c}M}\mathcal{L}_n(f).
\end{aligned}$$

□

The following is the proof for Theorem 18.

Proof of Theorem 18 . Noting that $f_{t+1} \leftarrow f_t - \eta w_{t+1}^\top T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$, we get the following bound by a similar way in the proof for Theorem 17.

$$\begin{aligned}
\mathcal{L}_n(f_{t+1}) \leq \mathcal{L}(f_t) &- \eta \left\langle \nabla\mathcal{L}_n(f_t), w_{t+1}^\top T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\right\rangle_{L_2^c(\nu_{n,X})} \\
&+ \frac{A\eta^2}{2}\|w_{t+1}^\top T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_2^c(\nu_{n,X})}^2.
\end{aligned}$$

We here bound the right hand side of this inequality as follows.

$$\begin{aligned}
\left\langle \nabla\mathcal{L}_n(f_t), w_{t+1}^\top T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\right\rangle_{L_2^c(\nu_{n,X})} &= \langle w_{t+1}\nabla\mathcal{L}_n(f_t), T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\rangle_{L_2^d(\nu_{n,X})} \\
&= \langle \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1}), T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\rangle_{L_2^d(\nu_{n,X})} \\
&\geq \gamma\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})}^2,
\end{aligned}$$

where we used Proposition 21 for the second equality and Assumption 9 for the last inequality. Recalling $\|w_{t+1}\|_2 \leq c_\lambda$, we have

$$\begin{aligned}
\|w_{t+1}^\top T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_2^c(\nu_{n,X})}^2 &= \mathbb{E}_{X\sim\nu_{n,X}}\|w_{t+1}^\top T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_2^2 \\
&\leq c_\lambda^2 \mathbb{E}_{\nu_{n,X}}\|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(X)\|_2^2
\end{aligned}$$

$$\leq c_\lambda^2 K^2 \|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(X)\|^2_{L_1^d(\nu_{n,X})}.$$

where for the second inequality, we used $\|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(X)\|_2 \leq K\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})}$ which is a consequence of the triangle inequality. Combining the above three inequalities, we have

$$\begin{aligned}
\mathcal{L}_n(f_{t+1}) &\leq \mathcal{L}_n(f_t) - \eta\left(\gamma - \frac{1}{2}A\eta c_\lambda^2 K^2\right)\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|^2_{L_1^d(\nu_{n,X})} \\
&\leq \mathcal{L}_n(f_t) - \frac{\eta\gamma}{2}\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|^2_{L_1^d(\nu_{n,X})} \\
&\leq \mathcal{L}_n(f_t) - \frac{\eta\gamma\sigma^2}{2}\|\nabla_f \mathcal{L}_n(f_t)\|^2_{L_1^c(\nu_{n,X})}, \qquad (9.14)
\end{aligned}$$

where we used $A\eta c_\lambda^2 K^2 \leq \gamma$ for the second inequality and we used (9.13) for the last inequality. Thus, we obtain from (9.14) and Proposition 22 that

$$\mathcal{L}_n(f_{t+1}) \leq \mathcal{L}_n(f_t) - \eta\alpha\mathcal{L}_n^2(f_t),$$

where $\alpha = \frac{\gamma\sigma^2(1-\exp(-M))^2}{2cM^2}$.

From this inequality, we get

$$\frac{1}{\mathcal{L}_n(f_t)} \geq \frac{1}{\mathcal{L}_n(f_{t+1})} - \eta\alpha\frac{\mathcal{L}_n(f_t)}{\mathcal{L}_n(f_{t+1})} \geq \frac{1}{\mathcal{L}_n(f_{t+1})} - \eta\alpha,$$

where for the last inequality we used the fact that $\mathcal{L}_n(f_t)$ is monotone decreasing which is confirmed from the inequality (9.14). Therefore, by applying this bound recursively for $t \in \{0, \ldots, T/2 - 1\}$ with $\eta = \eta_0$, we conclude

$$\mathcal{L}_n(f_{T/2}) \leq \frac{2\mathcal{L}_n(f_0)}{2 + \eta_0\alpha\mathcal{L}_n(f_0)T}. \qquad (9.15)$$

On the other hand, by summing up the inequality (9.14) over $t \in \{T/2, \ldots, T-1\}$ with η_1 and dividing by $T/2$, we get

$$\frac{\eta_1\gamma\sigma^2}{T}\sum_{t=T/2}^{T-1}\|\nabla_f \mathcal{L}_n(f_t)\|^2_{L_1^c(\nu_{n,X})} \leq \frac{2}{T}\mathcal{L}_n(f_{T/2}). \qquad (9.16)$$

From inequalities (9.15) and (9.16), we conclude

$$\frac{\eta_1\gamma\sigma^2}{T}\sum_{t=T/2}^{T-1}\|\nabla_f \mathcal{L}_n(f_t)\|^2_{L_1^c(\nu_{n,X})} \leq \frac{4\mathcal{L}_n(f_0)}{(2 + \eta_0\alpha\mathcal{L}_n(f_0)T)T}.$$

□

We next show Theorem 19 that gives the generalization bound by the margin distribution. To do that, we give an upper-bound on the margin distribution by the functional gradient norm.

Proposition 23. *For $\forall \delta > 0$, the following bound holds.*

$$\mathbb{P}_{\nu_n}[m_f(X,Y) \le \delta] \le \left(1 + \frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.$$

Proof. If $m_f(x,y) \le \delta$, then, we see

$$\sum_{y'\neq y}\exp(f_{y'}(x) - f_y(x)) \ge \exp\left(\max_{y'\neq y} f_{y'}(x) - f_y(x)\right) = \exp(-m_f(x,y)) \ge \exp(-\delta).$$

This implies,

$$p_f(y|x) = \frac{1}{1 + \sum_{y'\neq y}\exp(f_{y'}(x) - f_y(x))} \le \frac{1}{1+\exp(-\delta)}.$$

Thus, we get by Markov inequality and Proposition 19,

$$\begin{aligned}
\mathbb{P}_{\nu_n}[m_f(X,Y) \le \delta] &\le \mathbb{P}_{\nu_n}\left[p_f(Y|X) \le \frac{1}{1+\exp(-\delta)}\right] \\
&= \mathbb{P}_{\nu_n}\left[1 - p_f(Y|X) \ge \frac{\exp(-\delta)}{1+\exp(-\delta)}\right] \\
&\le \left(1 + \frac{1}{\exp(-\delta)}\right)\mathbb{E}_{\nu_n}[1 - p_f(Y|X)] \\
&= \left(1 + \frac{1}{\exp(-\delta)}\right)\mathbb{E}_{\nu_n}[\nu_n(Y|X) - p_f(Y|X)] \\
&\le \left(1 + \frac{1}{\exp(-\delta)}\right)\sum_{y\in\mathcal{Y}}\|\nu_n(y|\cdot) - p_f(y|\cdot)]\|_{L_1(\nu_{n,X})} \\
&\le \left(1 + \frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.
\end{aligned}$$

□

We prove here Theorem 19.

Proof of Theorem 19 . To proof the theorem, we give the network structure. Note that the connection at the t-th layer is as follows.

$$\phi_{t+1}(x) = \phi_t(x) - D_t\sigma(C_t\phi_t(x)).$$

We define recursively the family of functions $\mathcal{H}_t$ and $\hat{\mathcal{H}}_t$ where each neuron belong: We denote by $P_j \in \mathbb{R}^d$ the projection vector to j-th coordinate.

$$\begin{aligned}
\mathcal{H}_0 &\overset{\text{def}}{=} \{P_j : \mathcal{X} \to \mathbb{R} \mid j \in \{1,\dots,d\}\},\\
\hat{\mathcal{H}}_t &\overset{\text{def}}{=} \{\sigma(c_t^\top \phi_t) : \mathcal{X} \to \mathbb{R} \mid \phi_t \in \mathcal{H}_t^d, c_{t-1} \in \mathbb{R}^d, \|c_{t-1}\|_1 \le \Lambda\},\\
\mathcal{H}_{t+1} &\overset{\text{def}}{=} \{\phi_t^j - d_t^\top \psi_t : \mathcal{X} \to \mathbb{R} \mid \phi_t^j \in \mathcal{H}_t, \psi_t \in \hat{\mathcal{H}}_t^d, d_t \in \mathbb{R}^d, \|d_t\|_1 \le \Lambda'_t\}.
\end{aligned}$$

Then, the family of predictors of $y \in \mathcal{Y}$ can be written as

$$\mathcal{G}_{T-1,y} \overset{\text{def}}{=} \{w_y^\top \phi_{T-1} : \mathcal{X} \to \mathbb{R} \mid \phi \in \mathcal{H}_{T-1}^d, w_y \in \mathbb{R}^d, \|w_y\|_1 \le \Lambda_w\}.$$

Note that $\mathcal{G}_{T-1} = \{(f_y)_{y\in\mathcal{Y}} \mid f_y \in \mathcal{G}_{T-1,y}, y \in \mathcal{Y}\}$.

From these relationships and Lemma 17, we get

$$\begin{aligned}
\hat{\Re}_S(\mathcal{H}_t) &\le \hat{\Re}_S(\mathcal{H}_{t-1}) + \Lambda'_{t-1}\hat{\Re}_S(\hat{\mathcal{H}}_{t-1})\\
&\le (1 + \Lambda'_{t-1}\Lambda L_\sigma)\hat{\Re}_S(\mathcal{H}_{t-1}),\\
\hat{\Re}_S(\mathcal{G}_{T-1,y}) &\le \Lambda_w \hat{\Re}_S(\mathcal{H}_{T-1}).
\end{aligned}$$

The Rademacher complexity of $\mathcal{H}_0$ is obtained as follows. Since $\|P_j\|_2 = 1$, we have

$$\begin{aligned}
\hat{\Re}_S(\mathcal{H}_0) &= \frac{1}{n}\mathbb{E}_{(\sigma_i)_{i=1}^n}\left[\sup_{j\in\{1,\dots,d\}} \sum_{i=1}^n \sigma_i P_j x_i\right]\\
&\le \frac{1}{n}\mathbb{E}_{(\sigma_i)_{i=1}^n}\left[\sup_{j\in\{1,\dots,d\}} \|P_j\|_2 \left\|\sum_{i=1}^n \sigma_i x_i\right\|_2\right]\\
&= \frac{1}{n}\mathbb{E}_{(\sigma_i)_{i=1}^n}\left[\left\|\sum_{i=1}^n \sigma_i x_i\right\|_2\right]\\
&\le \frac{1}{n}\left(\mathbb{E}_{(\sigma_i)_{i=1}^n}\left[\left\|\sum_{i=1}^n \sigma_i x_i\right\|_2^2\right]\right)^{\frac{1}{2}}\\
&= \frac{1}{n}\left(\sum_{i=1}^n \|x_i\|_2^2\right)^{\frac{1}{2}} \le \frac{\Lambda_\infty}{\sqrt{n}},
\end{aligned}$$

where we used the independence of σ_i when taking the expectation.

We set $\Pi\mathcal{G}_{T-1} = \{f_y(\cdot) : \mathcal{X} \to \mid f \in \mathcal{G}_{T-1}, y \in \mathcal{Y}\}$. Noting that $\hat{\Re}_S(\Pi\mathcal{G}_{T-1}) \le \sum_{y\in\mathcal{Y}} \hat{\Re}_S(\mathcal{G}_{T-1,y})$, we get

$$\hat{\Re}_S(\Pi\mathcal{G}_{T-1}) \le c\Lambda_w\Lambda_\infty \prod_{t=0}^{T-2}(1 + \Lambda\Lambda'_t L_\sigma)/\sqrt{n}.$$

Thus, we can finish the proof by applying Proposition 23 and Lemma 18. □

Corollary 7 can be derived by instantiating Theorem 19 for various choices of $T,\ \Lambda'_t$.

Proof of Corollary 7 . For simplicity, we set $v_{t,j}$ the L_1-norm of j-th row of D_t, namely, $v_{t,j} \overset{\mathrm{def}}{=} \|(D_t)_{j,*}\|_1$. For arbitrary positive integers $(T, \overline{k}_{T-2}) = (T, k_0, \ldots, k_{T-2})$, we set $B(T, \overline{k}_{T-2})$ to networks defined by parameters included in

$$\left\{ \max_j v_{t,j} \le \frac{k_t}{T},\ \max_c \|(w_T)_{*,c}\|_1 \le \Lambda_w,\ \max_i \|(C_t)_{i,*}\|_1 \le \Lambda,\ \forall t \in \{0, \ldots, T-2\} \right\}$$

and set

$$\rho(T, \overline{k}_{T-2}) \overset{\mathrm{def}}{=} \frac{\rho}{T(T+1)k_0(k_0+1)\cdots k_{T-2}(k_{T-2}+1)}.$$

Moreover, we set $B \overset{\mathrm{def}}{=} \cup_{T,\overline{k}_{T-2}} B(T, \overline{k}_{T-2})$. Clearly, we see $\sum_{T,\overline{k}_{T-2}} \rho(T, \overline{k}_{T-2}) = \rho$. Therefore, by instantiating Theorem 19 for all $(T, \overline{k}_{T-2})$ with probability at least $1 - \rho(T, \overline{k}_{T-2})$ and taking an union bound, we have that with probability at least $1-\rho$ for $\forall f \in B$,

$$\begin{aligned}
\mathbb{P}_\nu[m_f(X,Y) \le 0] \le{}& \frac{2c^3 \Lambda_\infty \Lambda_w}{\delta\sqrt{n}} \prod_{t=0}^{T-2} \left(1 + \Lambda L_\sigma \frac{k_t}{T}\right) \\
&+ \sqrt{\frac{1}{2n} \log\left(\frac{1}{\rho}\right) + 2\log(T+1) + \sum_{t=0}^{T-2} 2\log(k_t+1)} \\
&+ \left(1 + \frac{1}{\exp(-\delta)}\right) \sqrt{c} \|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.
\end{aligned}$$

Let $f \in B$ be a function obtained by Algorithm 21 and $(f_t) = (w_{t+1}^\top \phi_t)$ be a sequence to obtain f in the algorithm. We choose the minimum integers $(T, \overline{k}_{T-2})$ such that $f \in B(T, \overline{k}_{T-2})$, then

$$\max_j v_{t,j} \le \frac{k_t}{T} \le \max_j v_{t,j} + \frac{1}{T}.$$

Thus, it follows that

$$\begin{aligned}
\mathbb{P}_\nu[m_f(X,Y) \le 0] \le{}& \left(1 + \frac{1}{\exp(-\delta)}\right) \sqrt{c} \|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})} \\
&+ \frac{2c^3 \Lambda_\infty \Lambda_w}{\delta\sqrt{n}} \prod_{t=0}^{T-2} \left(1 + \Lambda L_\sigma \left(\max_j v_{t,j} + \frac{1}{T}\right)\right) \\
&+ \sqrt{\frac{1}{2n} \left(\log\left(\frac{1}{\rho}\right) + 2\log(T+1) + \sum_{t=0}^{T-2} 2\log(T \max_j v_{t,j} + 2)\right)}.
\end{aligned}$$

We next estimate an upper bound on $\max_j v_{t,j}$. Note that $\|(A_tB_t)_{i,*}\|_1 \leq \|(A_t)_{i,*}\|_2 \sum_l \|(B_t)_{*,l}\|_2 \leq \|(A_t)_{i,*}\|_2\Lambda''$ and $\sum_i \|(A_t)_{i,*}\|_2 \leq \sqrt{Kd}\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})}$ by its construction. Thus, we get

$$\max_j v_{t,j} \leq \sum_j v_{t,j} \leq \eta_t\sqrt{Kd}\Lambda''\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})} \leq \eta_t\sqrt{Kd}\Lambda'' c_\lambda\|\nabla_f \mathcal{L}_n(f_t)\|_{L_1^c(\nu_{n,X})},$$

where we used $\|w_t\|_2 \leq c_\lambda$ for the last inequality.

Using the inequality $\prod_{t=0}^{T-2}(1+p_t) \leq \left(1+\frac{1}{T-1}\sum_{t=0}^{T-2} p_t\right)^{T-1}$ for positive integers (p_t) and Jensen's inequality, we have

$$\begin{aligned}\mathbb{P}_\nu[m_f(X,Y)\leq 0] \leq{}& \left(1+\frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f\mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})} + \frac{c^3\Lambda_\infty\Lambda_w}{\delta\sqrt{n}}\left(1+\frac{2\Lambda L_\sigma}{T}\right)^{T-1} \\ &+ \frac{c^3\Lambda_\infty\Lambda_w}{\delta\sqrt{n}}\left(1+\frac{2\Lambda L_\sigma\sqrt{Kd}\Lambda'' c_\lambda}{T-1}\sum_{t=0}^{T-2}\eta_t\|\nabla_f\mathcal{L}_n(f_t)\|_{L_1^c(\nu_{n,X})}\right)^{T-1} \\ &+ \sqrt{\frac{1}{2n}\left(\log\left(\frac{1}{\rho}\right)+O(T\log T)\right)}.\end{aligned}$$

Since $\left(1+\frac{2\Lambda L_\sigma}{T}\right)^{T-1}$ is an increasing with respect to T and converges to $2\Lambda L_\sigma$, we finish the proof. $\square$

9.7.3 Sample-splitting technique

In this subsection, we provide proofs for the convergence analysis of the sample-splitting variant of the method for the expected risk minimization. We first give the statistical error bound on the gap between the empirical and expected functional gradients.

Proof of Proposition 20 . For the probability measure ν, we denote by $\phi_\sharp\nu$ the push-forward measure $(\phi, id)_\sharp\nu$, namely, $(\phi, id)_\sharp\nu$ is the measure that the random variable $(\phi(X), Y)$ follows. We also define $\phi_\sharp\nu_m$ in the same manner. Then, we get

$$\begin{aligned}&\|T_k\partial_\phi\mathcal{R}(\phi,w_0) - T_{k,m}\partial_\phi\mathcal{R}_m(\phi,w_0)\|_{L_2^d(\mu)} \\ &= \sqrt{\mathbb{E}_{X'\sim\mu}\|\mathbb{E}_\nu[\partial_z l(\phi(X),Y,w_0)k(X,X')] - \mathbb{E}_{\nu_m}[\partial_z l(\phi(X),Y,w_0)k(X,X')]\|_2^2} \\ &= \sqrt{\sum_{j=1}^d \mathbb{E}_{X'\sim\mu}|(\mathbb{E}_\nu[\partial_{z_j} l(\phi(X),Y,w_0)\iota(\phi(X)))] - \mathbb{E}_{\nu_m}[\partial_{z_j} l(\phi(X),Y,w_0)\iota(\phi(X))])^\top\iota(\phi(X'))|^2} \\ &\leq \sqrt{K\sum_{j=1}^d \|\mathbb{E}_\nu[\partial_{z_j} l(\phi(X),Y,w_0)\iota(\phi(X)))] - \mathbb{E}_{\nu_m}[\partial_{z_j} l(\phi(X),Y,w_0)\iota(\phi(X))]\|_2^2}\end{aligned}$$

$$\leq \sqrt{K \sum_{j=1}^{d} \sum_{i=1}^{D} \left| \mathbb{E}_{\phi_\sharp \nu}[\partial_{z_j} l(X,Y,w_0)\iota^i(X))] - \mathbb{E}_{\phi_\sharp \nu_m}[\partial_{z_j} l(X,Y,w_0)\iota^i(X)] \right|^2}. \tag{9.17}$$

To derive an uniform bound on (9.17), we estimate Rademacher complexity of

$$\mathcal{G}_{ij} \overset{\text{def}}{=} \{\partial_{z_j} l(x,y,w_0)\iota^i(x) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \mid \iota^i \in \mathcal{F}^i\}.$$

For $(x_l, y_l)_{l=1}^m \subset \mathcal{X} \times \mathcal{Y}$, we set $h_l(r) = r\partial_{z_j} l(x_l, y_l, w_0)$. Since, $|\partial_{z_j} l(x_l, y_l, w_0)| \leq \beta_{\|w_0\|_2}$ by Assumption 10, h_l is $\beta_{\|w_0\|_2}$-Lipschitz continuous. Thus, from Lemma 16 and Lemma 17, there exists M such that for all $i \in \{1, \ldots, D\},\ j \in \{1, \ldots, d\}$,

$$\begin{aligned}
\hat{\Re}_m(\mathcal{G}_{ij}) &= \mathbb{E}_\sigma \left[\sup_{\iota^i \in \mathcal{F}^i} \sum_{l=1}^{m} \sigma_l h_l(\iota^i(x_l)) \right] \\
&\leq \beta_{\|w_0\|_2} \mathbb{E}_\sigma \left[\sup_{\iota^i \in \mathcal{F}^i} \sum_{l=1}^{m} \sigma_l \iota^i(x_l) \right] \\
&\leq \beta_{\|w_0\|_2} \frac{M}{\sqrt{m}}.
\end{aligned}$$

Therefore, by applying Lemma 15 with $\delta = \frac{\rho}{dD}$ for $\forall i, j$ simultaneously, it follows that with probability at least $1 - \rho$ for $\forall i, j$

$$\begin{aligned}
&\sup_{\iota^i \in \mathcal{F}^i} \left| \mathbb{E}_{\phi_\sharp \nu}[\partial_{z_j} l(X,Y,w_0)\iota^i(X))] - \mathbb{E}_{\phi_\sharp \nu_m}[\partial_{z_j} l(X,Y,w_0)\iota^i(X)] \right| \\
&\qquad \leq \frac{\beta_{\|w_0\|_2}}{\sqrt{m}} \left(2M + \sqrt{2K \log \frac{2dD}{\rho}} \right).
\end{aligned} \tag{9.18}$$

Putting (9.18) int (9.17), we get with probability at least $1 - \rho$

$$\sup_{\iota \in \mathcal{F}} \|T_k \partial_\phi \mathcal{R}(\phi, w_0) - T_{k,m} \partial_\phi \mathcal{R}_m(\phi, w_0)\|_{L_2^d(\mu)} \leq \beta_{\|w_0\|_2} \sqrt{\frac{KdD}{m}} \left(2M + \sqrt{2K \log \frac{2dD}{\rho}} \right).$$

□

We here prove Theorem 20 by using statistical guarantees of empirical functional gradients.

Proof of Theorem 20 . For notational simplicity, we set $m \leftarrow \lfloor n/T \rfloor$ and $\delta \leftarrow \rho/T$. We first note that

$$\langle \partial_\phi \mathcal{R}(\phi_t, w_0), T_{k_t,m} \partial_\phi \mathcal{R}_m(\phi_t, w_0) \rangle_{L_2^d(\nu_X)}$$

$$= \frac{1}{m}\sum_{j=1}^{m} \mathbb{E}_{\nu_X}[\partial_\phi \mathcal{R}(\phi_t, w_0)(X)^\top \partial_\phi \mathcal{R}_m(\phi_t, w_0)(x_j) k_t(X, x_j)]$$
$$= \frac{1}{m}\sum_{j=1}^{m} T_{k_t}\partial_\phi \mathcal{R}(\phi_t, w_0)(x_j)^\top \partial_\phi \mathcal{R}_m(\phi_t, w_0)(x_j)$$
$$= \langle T_{k_t}\partial_\phi \mathcal{R}(\phi_t, w_0), \partial_\phi \mathcal{R}_m(\phi_t, w_0)\rangle_{L_2^d(\nu_{m,X})}.$$

Noting that $\|\partial_z l(\phi_t(x_j), y_j, w_0)\|_2 \le \beta_{\|w_0\|_2}$ by Assumption 8, and applying Proposition 20 for all $t \in \{0, \dots, T-1\}$ independently, it follows that with probability at least $1 - T\delta$ (i.e., $1-\rho$) for $\forall t \in \{0, \dots, T-1\}$

$$\begin{aligned}
&\left|\langle \partial_\phi \mathcal{R}(\phi_t, w_0), T_{k_t,m}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\rangle_{L_2^d(\nu_X)} - \langle T_{k_t,m}\partial_\phi \mathcal{R}_m(\phi_t, w_0), \partial_\phi \mathcal{R}_m(\phi_t, w_0)\rangle_{L_2^d(\nu_{m,X})}\right| \\
&\quad \le \|T_{k_t}\partial_\phi \mathcal{R}(\phi_t, w_0) - T_{k_t,m}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\|_{L_2^d(\nu_{m,X})} \|\partial_\phi \mathcal{R}_m(\phi_t, w_0)\|_{L_2^d(\nu_{m,X})} \\
&\quad \le \beta_{\|w_0\|_2}\epsilon(m, \delta). \qquad (9.19)
\end{aligned}$$

We next give the following bound.

$$\|T_{k_t}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\|^2_{L_2^d(\nu_X)} = \mathbb{E}_{\nu_X}\left\|\frac{1}{m}\sum_{j=1}^{m} \partial_z l(\phi_t(x_i), y_i, w_0) k_t(x_i, X)\right\|_2^2 \le \beta^2_{\|w_0\|_2} K^2. \qquad (9.20)$$

On the other hand, we get by Proposition 18

$$\begin{aligned}
\mathcal{R}(\phi_{t+1}, w_0) \le \mathcal{R}(\phi_{t+1}, w_0) &- \eta \langle \partial_\phi \mathcal{R}(\phi_t, w_0), T_{k_t,m}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\rangle_{L_2^d(\nu_X)} \\
&+ \frac{\eta^2 A_{\|w_0\|_2}}{2}\|T_{k_t}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\|^2_{L_2^d(\nu_X)}. \qquad (9.21)
\end{aligned}$$

Combining inequalities (9.19), (9.20), and (9.21), we have with probability at least $1 - T\delta$ for $t \in \{0, \dots, T-1\}$,

$$\mathcal{R}(\phi_{t+1}, w_0) \le \mathcal{R}(\phi_{t+1}, w_0) - \eta\|T_{k_t,m}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\|^2_{k_t} + \eta\beta_{\|w_0\|_2}\epsilon(m, \delta) + \frac{\eta^2\beta^2_{\|w_0\|_2}K^2 A_{\|w_0\|_2}}{2}.$$

By Summing this inequality over $t \in \{0, \dots, T-1\}$ and dividing by T, we get with probability $1 - T\delta$

$$\frac{1}{T}\sum_{t=0}^{T-1}\|T_{k_t,m}\partial_\phi \mathcal{R}_m(\phi_t, w_0)\|^2_{k_t} \le \frac{\mathcal{R}(\phi_0, w_0)}{\eta T} + \beta_{\|w_0\|_2}\epsilon(m, \delta) + \frac{\eta\beta^2_{\|w_0\|_2}K^2 A_{\|w_0\|_2}}{2}.$$

Thus by Assumption 9 and the assumption on $w_0^\top w_0$, we get

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla_f \mathcal{L}_m(w_0^\top \phi_t)\|^q_{L_p^d(\nu_{m,X})}$$

$$\leq \frac{1}{\gamma\sigma^q}\left\{\frac{\mathcal{R}(\phi_0, w_0)}{\eta T} + \beta_{\|w_0\|_2}\epsilon(m,\delta) + \frac{\eta\beta_{\|w_0\|_2}^2 K^2 A_{\|w_0\|_2}}{2} + \gamma\epsilon\right\}. \tag{9.22}$$

To clarify the relationship between $\|\nabla_f \mathcal{L}_m(f)\|_{L_1^c(\nu_{m,X})}$ and $\|\nabla_f \mathcal{L}(f)\|_{L_1^c(\nu_X)}$, we take an expectation of the former term with respect to samples $(X_j, Y_j)_{j=1}^m \sim \nu^m$. Since $\|\partial_\zeta l(\zeta, y)\|_2 \leq B$, we obtain

$$\begin{aligned}
\mathbb{E}_{(X_j,Y_j)_{j=1}^m \sim \nu^m}\|\nabla_f \mathcal{L}_m(f)\|_{L_1^c(\nu_{m,X})} &= \mathbb{E}_{(X,Y)\sim\nu_m}\|\partial_\zeta l(f(X),Y)\|_2 \\
&\geq \frac{1}{B}\mathbb{E}_{(X,Y)\sim\nu_m}\|\partial_\zeta l(f(X),Y)\|_2^2 \\
&\geq \frac{1}{B}\mathbb{E}_{\nu_{m,X}}\|\mathbb{E}_{\nu(Y|X)}[\partial_\zeta l(f(X),Y)]\|_2^2 \\
&= \frac{1}{B}\mathbb{E}_{\nu_{m,X}}\|\nabla_f \mathcal{L}(f)(X)\|_2^2 \\
&= \frac{1}{B}\|\nabla_f \mathcal{L}(f)\|_{L_2^c(\nu_X)}^2.
\end{aligned}$$

Hence, applying Hoeffding's inequality with $\delta \leftarrow \rho/T$ to $\mathbb{E}_{(X_j,Y_j)_{j=1}^m \sim \nu^m}\|\nabla_f \mathcal{L}_m(w_0^\top \phi_t)\|_{L_1^c(\nu_{m,X})}$ for all $t \in \{0, \ldots, T-1\}$ independently, we find that with probability $1 - T\delta$ for $\forall t \in \{0, \ldots, T-1\}$,

$$\begin{aligned}
\|\nabla_f \mathcal{L}_m(w_0^\top \phi_t)\|_{L_1^c(\nu_{m,X})} + B\sqrt{\frac{2}{m}\log\frac{1}{\delta}} &\geq \mathbb{E}_{\sim\nu^m}\|\nabla_f \mathcal{L}_m(w_0^\top \phi_t)\|_{L_1^c(\nu_{m,X})} \\
&\geq \frac{1}{B}\|\nabla_f \mathcal{L}(w_0^\top \phi_t)\|_{L_1^c(\nu_X)}^2,
\end{aligned} \tag{9.23}$$

where we used for the last inequality $\|\cdot\|_{L_2^c(\nu_X)}^2 \geq \|\cdot\|_{L_1^c(\nu_X)}^2$.

We set $t_* = \arg\min_{t\in\{0,\ldots,T-1\}}\|\nabla_f \mathcal{L}_m(w_0^\top \phi_t)\|_{L_p^d(\nu_{m,X})}$. Combining inequalities (9.22) and (9.23) and noting $p \geq 1$, we get with probability at least $1 - 2T\delta$,

$$\begin{aligned}
\frac{1}{B}\|\nabla_f \mathcal{L}(w_0^\top \phi_{t_*})\|_{L_1^c(\nu_X)}^2 \leq B\sqrt{\frac{2}{m}\log\frac{1}{\delta}} \\
+ \frac{1}{\gamma^{1/q}\sigma}\left\{\frac{\mathcal{R}(\phi_0, w_0)}{\eta T} + \beta_{\|w_0\|_2}\epsilon(m,\delta) + \frac{\eta\beta_{\|w_0\|_2}^2 K^2 A_{\|w_0\|_2}}{2} + \gamma\epsilon\right\}^{\frac{1}{q}}.
\end{aligned}$$

Noting that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b > 0$, we finally obtain

$$\|\nabla_f \mathcal{L}(w_0^\top \phi_{t_*})\|_{L_1^c(\nu_X)} \leq B\left(\frac{2}{m}\log\frac{1}{\delta}\right)^{\frac{1}{4}}$$

$$+ \sqrt{\frac{B}{\gamma^{1/q}\sigma}} \left\{ \frac{\mathcal{R}(\phi_0, w_0)}{\eta T} + \beta_{\|w_0\|_2}\epsilon(m, \delta) + \frac{\eta\beta^2_{\|w_0\|_2} K^2 A_{\|w_0\|_2}}{2} + \gamma\epsilon \right\}^{\frac{1}{2q}} .$$

Recalling that $m \leftarrow \lfloor n/T \rfloor$ and $\delta \leftarrow \rho/T$, the proof is finished. $\square$

Future Work

In this thesis, we have introduced our work for large-scale or complicated machine learning problems. We here mention possible extensions of our work.

In Chapter 3 and 4, we have introduced optimal methods in terms of the total and iteration complexities, which are expected to better work by parallelization of minibatch computing. Thus, we will try to further investigate in this line of research and compare with asynchronous variants of stochastic gradient methods and variance reduced methods (Leblond et al., 2017) for huge datasets empirically and theoretically. In Chapter 5, we have introduced a stochastic variant of difference of convex algorithm and provided better theoretical analysis. Moreover, superior empirical performance has been verified especially on training restricted Boltzmann machines. Therefore, it is expected to explore applications using this model efficiently. One candidate for such an application is unsupervised ensemble (Dawid and Skene, 1979; Shaham et al., 2016) which has gained attention recently because of increasing demand for crowd labeling for supervised learning tasks (Brabham, 2008; Kittur et al., 2008). Furthermore, exploring other applications from Boltzmann machines is also highly expected because the difference of convex structure can appear in several fields.

In Chapter 6 and subsequence chapters, the functional gradient method for learning probability measures and its applications with several theoretical results have been provided. One common notion appeared in these chapters is notable deep learning network structure called residual networks (He et al., 2016). It is widely known empirically that residual networks have remarkable performance especially for computer vision tasks, but there is a lack of better theoretical explanations of their ability. Thus, such an analysis is desired in machine learning community. In our work, we haves shown the convergence of the functional gradient method in several situations. As shown in this thesis, the method has the connections with residual network. This means that we also provide sufficient size of residual networks to achieve a required accuracy. This is quite meaningful because network size is important for analyzing its generalization ability. Moreover, functional gradient methods under settings considered in Chapter 7 and 8 are also closely related with sampling methods. Therefore, we think that our analyses also provide a new way for theoretical guarantees of some specific sampling methods. Especially, Stein variational gradient descent (Liu and Wang, 2016) recently proposed sampling method in Bayesian inference literature to better match to a posterior distribution may be further investigated theoretically using our technique by combing the analysis for kernel function like that provided in Chapter 9, although we only used simple analysis for the kernel because we adopted variable kernel functions thanks to its better empirical performance rather than the more analyzable fixed kernel function. We next comment on each application in Chapter 8 and 9. In Chapter 8, we have introduce the method for enhancing the convergence of adversarial generative models (Goodfellow et al., 2014; Arjovsky et al., 2017; Gulrajani et al., 2017) by applying the functional gradient descent on the generator. Thus, we focused only on the generator, but performance of the discriminator (critic) is also clearly important. In particular, we think that the smoothing technique for the discriminator is useful to generate better samples as done in Miyato et al. (2018) because the smoothness may exclude a kind of adversarial example (Goodfellow et al., 2015; Kurakin et al., 2016) which cheats discriminator by unrealistic samples. Thus, we believe that the performance of the functional gradient descent can be further improved by combining such a smoothing technique. In Chapter 9, the new functional gradient boosting method based on the property of the residual network has been proposed. Indeed, an interesting and better theoretical analysis of the method was shown, but the important problem is left, that is, a theoretical benefit of deep structure. If we can show the benefit of our method to shallow models such as usual functional gradient boosting methods (Mason et al., 1999; Friedman, 2001), the concrete superiority of our method is also shown. The first step for this line of research is to give some examples where a deep structure approximate functional gradients well more efficiently than shallow models. As for empirical verification, we have shown that the proposed method outperforms existing methods including the state-of-the-art method such as LightGBM (Ke et al., 2017) on general benchmark datasets. However, such a result on image datasets on which regular residual networks perform well has not yet observed and additional efforts will be required to achieve this goal. This is one of important topics left for future work.

Bibliography

Agarwal, A. and Bottou, L. (2014). A lower bound for the optimization of finite sums. *arXiv preprint arXiv:1410.0723*.

Agarwal, A. and Duchi, J. C. (2011). Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems 24*, pages 873–881.

Agarwal, A., Wainwright, M. J., Bartlett, P. L., and Ravikumar, P. K. (2009). Information-theoretic lower bounds on the oracle complexity of convex optimization. In *Advances in Neural Information Processing Systems 22*, pages 1–9.

Ahmadi, A. A. and Hall, G. (2015). DC decomposition of nonconvex polynomials with algebraic techniques. *Mathematical Programming*, pages 1–26.

Allen-Zhu, Z. (2017). Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of Annual ACM SIGACT Symposium on Theory of Computing 49*, pages 1200–1205. ACM.

Allen-Zhu, Z. and Hazan, E. (2016). Optimal black-bbox reductions between optimization objectives. In *Advances in Neural Information Processing Systems 29*, pages 1614–1622.

Allen-Zhu, Z. and Orecchia, L. (2014). Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Ambrosio, L. (2003). Lecture notes on optimal transport problems. In *Mathematical aspects of evolving interfaces*, pages 1–52.

Ambrosio, L., Gigli, N., and Savaré, G. (2008). *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Lectures in Mathematics. ETH Zürich. Birkhäuser Basel.

Argyriou, A., Hauser, R., Micchelli, C. A., and Pontil, M. (2006). A DC-programming algorithm for kernel selection. In *Proceedings of international conference on Machine learning 23*, pages 41–48.

Arjevani, Y. and Shamir, O. (2016). Dimension-free iteration complexity of finite sum optimization problems. In *Advances in Neural Information Processing Systems 29*, pages 3540–3548.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of International Conference on Machine Learning 34*, pages 214–223.

Bach, F. (2014). Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53.

Bach, F. and Moulines, E. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems 24*, pages 451–459.

Bach, F. and Moulines, E. (2013). Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In *Advances in Neural Information Processing Systems 26*, pages 773–781.

Balakrishnan, S., Wainwright, M. J., and Yu, B. (2017). Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77–120.

Bartlett, P. L., Evans, S. N., and Long, P. M. (2018). Representing smooth functions as compositions of near-identity functions with implications for deep network optimization. *arXiv preprint arXiv:1804.05012*.

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2017). Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *arXiv preprint arXiv:1703.02930*.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Bartlett, P. L., Maiorov, V., and Meir, R. (1998). Almost linear VC dimension bounds for piecewise polynomial networks. In *Advances in Neural Information Processing Systems 11*, pages 190–196.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Bartlett, P. L. and Traskin, M. (2007). Adaboost is consistent. *Journal of Machine Learning Research*, 8(Oct):2347–2368.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. In *NIPS 2012 Workshop: Deep Learning and Unsupervised Feature Learning*.

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36:105–139.

Becker, S. and Le Cun, Y. (1988). Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37.

Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O., and Marcotte, P. (2006). Convex neural networks. In *Advances in Neural Information Processing Systems 19*, pages 123–130.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Blanchard, G., Lugosi, G., and Vayatis, N. (2003). On the rate of convergence of regularized boosting classifiers. *Journal of Machine Learning Research*, 4(Oct):861–894.

Bottou, L., Curtis, F. E., and Nocedal, J. (2016). Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*.

Bottou, L. and Le Cun, Y. (2005). On-line learning for very large data sets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151.

Bousquet, O. and Bottou, L. (2008). The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 21*, pages 161–168.

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, 2(Mar):499–526.

Brabham, D. C. (2008). Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517.

Brenier, Y. (1987). Décomposition polaire et réarrangement monotone des champs de vecteurs. *CR Acad. Sci. Paris Sér. I Math*, 305(19):805–808.

Brenier, Y. (1991). Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

Carlson, D., Cevher, V., and Carin, L. (2015). Stochastic spectral descent for restricted Boltzmann machines. In *Proceedings of International Conference on Artificial Intelligence and Statistics 18*, pages 111–119.

Carlson, D., Hsieh, Y.-P., Collins, E., Carin, L., and Cevher, V. (2016). Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):296–311.

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., and Holtham, E. (2017a). Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*.

Chang, B., Meng, L., Haber, E., Tung, F., and Begert, D. (2017b). Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*.

Chen, C. and Zhang, R. (2017). Particle optimization in stochastic gradient mcmc. *arXiv preprint arXiv:1711.10927*.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 22*, pages 785–794.

Chen, X., Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29*, pages 2172–2180.

Chen, X., Lin, Q., and Pena, J. (2012). Optimal regularized dual averaging methods for stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 395–403.

Chizat, L. and Bach, F. (2018). On the global convergence of gradient descent for over-parameterized models using optimal transport. *arXiv preprint arXiv:1805.09545*.

Cho, K., Raiko, T., Ilin, A., and Karhunen, J. (2013). A two-stage pretraining algorithm for deep Boltzmann machines. In *International Conference on Artificial Neural Networks 23*, pages 106–113. Springer.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. (2017). Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of International Conference on Machine Learning 34*, pages 874–883.

Cortes, C., Mohri, M., and Syed, U. (2014). Deep boosting. In *Proceedings of International Conference on Machine Learning 31*, pages 1179–1187.

Cotter, A., Shamir, O., Srebro, N., and Sridharan, K. (2011). Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems 24*, pages 1647–1655.

Dai, B., He, N., Dai, H., and Song, L. (2016). Provable Bayesian inference via particle mirror descent. In *Proceedings of International Conference on Artificial Intelligence and Statistics 19*, pages 985–994.

Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.-F. F., and Song, L. (2014). Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems 27*, pages 3041–3049.

Daneri, S. and Savaré, G. (2010). Lecture notes on gradient flows and optimal transport. *arXiv preprint arXiv:1009.3737*.

Dawid, A. P. and Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, pages 20–28.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems 27*, pages 1646–1654.

Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L. (2012). Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40:139–157.

Dieuleveut, A. and Bach, F. (2016). Nonparametric stochastic approximation with large step-sizes. *The Annals of Statistics*, 44(4):1363–1399.

Dieuleveut, A., Flammarion, N., and Bach, F. (2017). Harder, better, faster, stronger convergence rates for least-squares regression. *Journal of Machine Learning Research*, 18(1):3520–3570.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Dudley, R. M. (1968). Distances of probability measures and random variables. *The Annals of Mathematical Statistics*, 39(5):1563–1572.

Dudley, R. M. (1999). *Uniform Central Limit Theorems*. Cambridge University Press.

Ferrer, A. (2001). Representation of a polynomial function as a difference of convex polynomials, with an application. *Lectures Notes in Economics and Mathematical Systems*, 502:189–207.

Freund, J. (1971). *Mathematical statistics*. Prentice-Hall.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, pages 1189–1232.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407.

Frostig, R., Ge, R., Kakade, S., and Sidford, A. (2015). Un-regularizing: Approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *Proceedings of International Conference on Machine Learning 32*, pages 2540–2548.

Ghadimi, S. and Lan, G. (2013a). Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, II: Shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089.

Ghadimi, S. and Lan, G. (2013b). Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368.

Giselsson, P. and Boyd, S. (2014). Monotonicity and restart in fast gradient methods. In *IEEE Conference on Decision and Control 53*, pages 5058–5063.

Gong, P. and Ye, J. (2014). Linear convergence of variance-reduced stochastic gradient without strong convexity. *arXiv preprint arXiv:1406.1102*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples.

Guillemin, V. and Pollack, A. (1974). *Differential Topology*. Prentice-Hall.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30*, pages 5769–5779.

Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. (2015). A globally convergent incremental Newton method. *Mathematical Programming*, 151(1):283–313.

Haber, E., Ruthotto, L., and Holtham, E. (2017). Learning across scales-a multiscale method for convolution neural networks. *arXiv preprint arXiv:1703.02009*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Hörmander, L. (1963). *Linear Partial Differential Operators*. Springer.

Huang, F., Ash, J., Langford, J., and Schapire, R. (2017a). Learning deep resnet blocks sequentially using boosting theory. *arXiv preprint arXiv:1706.04964*.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2017b). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700—4708.

Ikeda, S. (2000). Acceleration of the EM algorithm. *Systems and Computers in Japan*, 31(2):10–18.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning 32*, pages 448–456.

Jastrzebski, S., Arpit, D., Ballas, N., Verma, V., Che, T., and Bengio, Y. (2017). Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323.

Johnson, R. and Zhang, T. (2018). Composite functional gradient learning of generative adversarial models. *arXiv preprint arXiv:1801.06309*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pages 3149–3157.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations 3*.

Kittur, A., Chi, E. H., and Suh, B. (2008). Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 453–456.

Kivinen, J., Smola, A. J., and Williamson, R. C. (2004). Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). How to train your DRAGAN. *arXiv preprint arXiv:1705.07215*.

Koltchinskii, V. and Panchenko, D. (2002). Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50.

Koltchinskii, V., Panchenko, D., and Lozano, F. (2003). Bounding the generalization error of convex combinations of classifiers: Balancing the dimensionality and the margins. *Annals of Applied Probability*, 13(1):213–252.

Konečnỳ, J., Liu, J., Richtárik, P., and Takáč, M. (2016). Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255.

Konečnỳ, J. and Richtárik, P. (2013). Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Kushner, H. and Yin, G. G. (2003). *Stochastic approximation and recursive algorithms and applications*, volume 35 of *Applications of Mathematics: Stochastic Modelling and Applied Probability*. Springer.

Lacoste-Julien, S., Schmidt, M., and Bach, F. (2012). A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of international conference on Machine learning 33*, pages 1558–1566.

Le Roux, N. and Bengio, Y. (2007). Continuous neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics 11*, pages 404–411.

Le Thi, H. A. and Dinh, T. P. (2011). On solving linear complementarity problems by DC programming and DCA. *Computational optimization and applications*, 50(3):507–524.

Le Thi, H. A., Huynh, V. N., and Dinh, T. P. (2009). Convergence analysis of DC algorithm for DC programming with subanalytic data. Technical report, LMI, INSA-Rouen.

Le Thi, H. A., Le, H. M., and Dinh, T. P. (2008). A DC programming approach for feature selection in support vector machines learning. *Advances in Data Analysis and Classification*, 2(3):259–278.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. (2017). ASAGA: Asynchronous parallel SAGA. In *Proceedings of International Conference on Artificial Intelligence and Statistics 20*, pages 46–54.

Lin, H., Mairal, J., and Harchaoui, Z. (2015). A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems 28*, pages 3384–3392.

Lin, H., Mairal, J., and Harchaoui, Z. (2017). Catalyst acceleration for first-order convex optimization: from theory to practice. *arXiv preprint arXiv:1712.05654*.

Lin, Q., Lu, Z., and Xiao, L. (2014). An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems 27*, pages 3059–3067.

Littwin, E. and Wolf, L. (2016). The loss surface of residual networks: Ensembles and the role of batch normalization. *arXiv preprint arXiv:1611.02525*.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2017). Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2018). Hierarchical representations for efficient architecture search. In *Proceedings of International Conference on Learning Representations 6*.

Liu, Q. (2017). Stein variational gradient descent as gradient flow. In *Advances in Neural Information Processing Systems 30*, pages 3117–3125.

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems 29*, pages 2378–2386.

Livni, R., Carmon, D., and Globerson, A. (2017). Learning infinite layer networks without the kernel trick. In *Proceedings of International Conference on Machine Learning 34*, pages 2198–2207.

Lu, Y., Zhong, A., Li, Q., and Dong, B. (2017). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods*. John Wiley & Sons.

Lugosi, G. and Vayatis, N. (2004). On the Bayes-risk consistency of regularized boosting methods. *The Annals of Statistics*, 32(1):30–55.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

MacKay, D. J. (1995). Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505.

Mairal, J. (2013). Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems 26*, pages 2283–2291.

Mairal, J. (2015). Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855.

Mannor, S., Meir, R., and Zhang, T. (2003). Greedy algorithms for classification–consistency, convergence rates, and adaptivity. *Journal of Machine Learning Research*, 4(Oct):713–742.

Marlin, B., Swersky, K., Chen, B., and Freitas, N. (2010). Inductive principles for restricted Boltzmann machine learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics 13*, pages 509–516.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. (2000). Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (1999). Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, pages 512–518.

Milgrom, P. and Segal, I. (2002). Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks.

Moghimi, M., Belongie, S. J., Saberian, M. J., Yang, J., Vasconcelos, N., and Li, L.-J. (2016). Boosted convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, pages 24.1–24.13.

Mukherjee, S., Rifkin, R., and Poggio, T. (2003). Regression and classification with regularization. In *Nonlinear estimation and classification*, pages 111–128.

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29:429–443.

Murata, T. and Suzuki, T. (2017). Doubly accelerated stochastic variance reduced dual averaging method for regularized empirical risk minimization. In *Advances in Neural Information Processing Systems 30*, pages 608–617.

Neal, R. M. (2012). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer.

Nemirovski, A. S., Juditsky, A., Lan, G., and Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609.

Nemirovskii, A. and Yudin, D. B. (1983). *Problem Complexity and Method Efficiency in Optimization*. John Wiley.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $O(1/k^2)$. 27(2):372–376.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152.

Nesterov, Y. (2013). Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161.

Neyshabur, B., Tomioka, R., and Srebro, N. (2015). Norm-based capacity control in neural networks. In *Proceedings of Conference on Learning Theory 28*, pages 1376–1401.

Nitanda, A. (2014). Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems 27*, pages 1574–1582.

Nitanda, A. (2016). Accelerated stochastic gradient descent for minimizing finite sums. In *Proceedings of International Conference on Artificial Intelligence and Statistics 19*, pages 195–203.

Nitanda, A. and Suzuki, T. (2017a). Stochastic difference of convex algorithm and its application to training deep Boltzmann machines. In *Proceedings of International Conference on Artificial Intelligence and Statistics 20*, pages 470–478.

Nitanda, A. and Suzuki, T. (2017b). Stochastic particle gradient descent for infinite ensembles. *arXiv preprint arXiv:1712.05438*.

Nitanda, A. and Suzuki, T. (2018a). Functional gradient boosting based on residual network perception. *arXiv preprint arXiv:1802.09031*.

Nitanda, A. and Suzuki, T. (2018b). Gradient layer: Enhancing the convergence of adversarial training for generative models. In *Proceedings of International Conference on Artificial Intelligence and Statistics 21*.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*, pages 271–279.

Otto, F. (2001). The geometry of dissipative evolution equations: The porous medium equation. *Communications in Partial Differential Equations*, 26(1-2):101–174.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples*.

O ’ Donoghue, B. and Candes, E. (2015). Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. (2018). Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of International Conference on Learning Representations 4*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184.

Rahimi, A. and Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems 22*, pages 1313–1320.

Rakhlin, A., Shamir, O., and Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of International Conference on Machine Learning 29*, pages 1571–1578.

Rätsch, G. and Warmuth, M. K. (2005). Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(Dec):2131–2152.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of International Conference on Machine Learning 32*, pages 1530–1538.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Rosset, S., Swirszcz, G., Srebro, N., and Zhu, J. (2007). $_1$-regularization in infinite dimensional feature spaces. In *Proceedings of Conference on Learning Theory 20*, pages 544–558.

Rosset, S., Zhu, J., and Hastie, T. (2004). Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973.

Roux, N. L., Schmidt, M., and Bach, F. R. (2012). A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671.

Ruiz, D. (2012). A note on the uniformity of the constant in the Poincaré inequality. *Advanced Nonlinear Studies*, 12:889–903.

Ruppert, D. (1988). Efficient estimations from a slowly convergent Robbins-Monro process. Technical report, Cornell University Operations Research and Industrial Engineering.

Salakhutdinov, R. and Hinton, G. (2009). Deep Boltzmann machines. In *Proceedings of International Conference on Artificial Intelligence and Statistics 12*, pages 448–455.

Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of deep belief networks. In *Proceedings of International Conference on Machine Learning 25*, pages 872–879.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in Neural Information Processing Systems 29*, pages 2234–2242.

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686.

Schmidt, M., Le Roux, N., and Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112.

Schmidt, M. and Roux, N. L. (2013). Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*.

Shaham, U., Cheng, X., Dror, O., Jaffe, A., Nadler, B., Chang, J., and Kluger, Y. (2016). A deep learning approach to unsupervised ensemble learning. In *Proceedings of international conference on Machine learning 33*, pages 30–39.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shalev-Shwartz, S. and Zhang, T. (2012). Proximal stochastic dual coordinate ascent. *arXiv preprint arXiv:1211.2717*.

Shalev-Shwartz, S. and Zhang, T. (2013a). Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems 26*, pages 378–385.

Shalev-Shwartz, S. and Zhang, T. (2013b). Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599.

Shalev-Shwartz, S. and Zhang, T. (2014). Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *Proceedings of International Conference on Machine Learning 31*, pages 64–72.

Smola, A. J. and Schölkopf, B. (1998). *Learning with Kernels*. GMD-Forschungszentrum Informationstechnik.

Steinwart, I. (2005). Consistency of support vector machines and other regularized kernel classifiers. *IEEE Transactions on Information Theory*, 51(1):128–142.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines*. Springer.

Su, W., Boyd, S., and Candes, E. (2014). A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems 27*, pages 2510–2518.

Sudakov, V. N. (1979). *Geometric problems in the theory of infinite-dimensional probability distributions*. Number 141. American Mathematical Soc.

Tao, P. D. (1986). Algorithms for solving a class of nonconvex optimization problems. methods of subgradients. *North-Holland Mathematics Studies*, 129:249–271.

Tieleman, T. and Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. In *Proceedings of International Conference on Machine Learning 26*, pages 1033–1040.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. Technical report, COURSERA: Neural networks for machine learning.

Tsybakov, A. B. (2003). Optimal rates of aggregation. In *Proceedings of Conference on Learning Theory 16*, pages 303–313.

van der Vaart, A. and Wellner, J. (1996). *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer.

Vapnik, V. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

Veit, A., Wilber, M. J., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems 29*, pages 550–558.

Villani, C. (2008). *Optimal transport: old and new*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 511–518.

Wang, D. and Liu, Q. (2016). Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*.

Wang, S., Schwing, A., and Urtasun, R. (2014). Efficient inference of continuous markov random fields with polynomial potentials. In *Advances in Neural Information Processing Systems 27*, pages 936–944.

Wang, Z., Gu, Q., Ning, Y., and Liu, H. (2015). High dimensional em algorithm: Statistical optimization and asymptotic normality. In *Advances in Neural Information Processing Systems 28*, pages 2521–2529.

Weinan, E. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of International Conference on Machine Learning 28*, pages 681–688.

Woodworth, B. E. and Srebro, N. (2016). Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems 29*, pages 3639–3647.

Xiao, L. and Zhang, T. (2014). A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 5987–5995.

Yasuda, M. and Tanaka, K. (2008). Approximate learning algorithm for restricted Boltzmann machines. In *Proceedings of International Conference on Computational Intelligence for Modelling, Control & Automation Intelligent Agents, Web Technologies & Internet Commerce Innovation in Software Engineering*, pages 692–697.

Yasuda, M., Tannai, J., and Tanaka, K. (2012). Learning algorithm for Boltzmann machines using max-product algorithm and pseudo-likelihood. *Interdisciplinary information sciences*, 18(1):55–63.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pages 87.1—87.12.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5907–5915.

Zhang, T. (2003). Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691.

Zhang, T. (2004). Statistical behavior and consistency of classification methods based on convex ris minimization. *The Annals of Statistics*, 32(1):56–134.

Zhang, T. and Yu, B. (2005). Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579.

Zhang, Y. and Xiao, L. (2017). Stochastic primal-dual coordinate method for regularized empirical risk minimization. *The Journal of Machine Learning Research*, 18(1):2939–2980.

Zhu, C. and Xu, H. (2015). Online gradient descent in function space. *arXiv preprint arXiv:1512.02394*.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *Proceedings of International Conference on Learning Representations 5*.