Topics in Non-Convex Optimization and Learning

by

Hongyi Zhang

B.S., Peking University (2013)

Submitted to the Department of Brain and Cognitive Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2019

Signature redacted

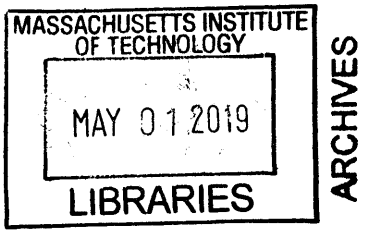
Author .....
Department of Brain and Cognitive Sciences
January 18, 2019

Signature redacted

Certified by .....
Suvrit Sra
Assistant Professor of EECS
Thesis Supervisor

Signature redacted

Accepted by .....
Matthew A. Wilson
Sherman Fairchild Professor of Neuroscience and Picower Scholar
Director of Graduate Education for Brain and Cognitive Sciences



Topics in Non-Convex Optimization and Learning

by

Hongyi Zhang

Submitted to the Department of Brain and Cognitive Sciences
on January 18, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Non-convex optimization and learning play an important role in data science and machine learning, yet so far they still elude our understanding in many aspects. In this thesis, I study two important aspects of non-convex optimization and learning: Riemannian optimization and deep neural networks.

In the first part, I develop iteration complexity analysis for Riemannian optimization, i.e., optimization problems defined on Riemannian manifolds. Through bounding the distortion introduced by the metric curvature, iteration complexity of Riemannian (stochastic) gradient descent methods is derived. I also show that some fast first-order methods in Euclidean space, such as Nesterov's accelerated gradient descent (AGD) and stochastic variance reduced gradient (SVRG), have Riemannian counterparts that are also fast under certain conditions.

In the second part, I challenge two common practices in deep learning, namely *empirical risk minimization* (ERM) and *normalization*. Specifically, I show (1) training on convex combinations of samples improves model robustness and generalization, and (2) a good initialization is sufficient for training deep residual networks without normalization. The method in (1), called mixup, is motivated by a data-dependent Lipschitzness regularization of the network. The method in (2), called ZeroInit, makes the network update scale invariant to its depth at initialization.

Thesis Supervisor: Suvrit Sra
Title: Assistant Professor of EECS

Acknowledgments

While the intellectual pursuit of a doctoral degree is a long journey, I am extremely lucky to have many people in companion. For those people who have generously offered their guidance, support, or simply shared their stories, thoughts and time with me, I would like to express my sincerest gratitude.

First and foremost, I would like to thank my doctoral advisor Professor Suvrit Sra, for supporting me in many different ways, such as astonishing me with his broad knowledge of mathematics literature, patiently helping me to polish slides and presentations, and reaching out for my career development. For the wisdom and kindness you pass on to me, it will take me many years to come to pass them on to others.

I would also like to thank my thesis committee members — Professor Tomaso Poggio, Professor Alexander Rakhlin and Professor Constantinos Daskalakis — for their helpful feedback on the content of this thesis. And special thanks go to my brilliant collaborators — Professor Suvrit Sra, Sashank J. Reddi, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz and Tengyu Ma. The work in this thesis would not be anything close to its current shape without their contribution of ideas and efforts.

During my graduate school life at MIT, a lot of people have shared their kindness with me. While the list of names would be too long to recall exhaustively, I would like to especially mention some. Professor Joshua B. Tenenbaum and Professor Joseph J. Lim provided their selfless help and suggestions in my most confused time. Much of joy and wisdom are shared by my roommates Peiguang Hu, Haizheng Zhang and Tianli Zhou, my office mates Xia Miao, Quan Li, Jingzhao Zhang, as well as Youzhi Liang, Chengtao Li, Hongzhou Lin and Zhen Yang.

Finally, the research work in this thesis would not be possible without fundings from the Singleton Fellowship and Leventhal Fellowship of the Department of Brain and Cognitive Sciences at MIT, as well as the NSF grant: IIS-1409802 and the DARPA-Lagrange grant.

Contents

List of Figures

List of Tables

1

Introduction

Mathematical models are central to human endeavor to better observe, understand and change the world, with applications across physical sciences, engineering, and social sciences. Importantly, as we try to tackle increasingly more complex problems, we have to rely less on hand-crafted solutions, and instead note down desired model characteristics as design specifications. This leads to a typical problem-solving pipeline: in the *modeling* phase, a model family with its search space, as well as the design specifications is described; in the *optimization* phase, an "optimal" model is found within the search space which best satisfies the design specifications. For example, if we want to optimize the shape of an airplane for maximal lift and minimal resistance, we build a model that precisely simulates the aerodynamics, then vary the shape parameters to improve the metric for lift and resistance. Similarly, if we want to build a mobile app that can recognize cats

and dogs in photos, we choose a model space (e.g., a neural network) and specify the desired model via a set of input-output pairs (i.e., image-label pairs); we then search for a model that best fits our specifications. At a coarse scale, the problems I study in this thesis all fall into this broad realm. In particular, they share an important feature with the latter example – the model design specifications are given as input-output pairs, known as the *training set* in machine learning. Now I will motivate and describe the scope of this work in more detail.

1.1 Scope of This Work

Many problems in machine learning and statistics share the common goal of finding a predictor (a.k.a. hypothesis) f^* in some hypothesis space $\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y}\}$ that achieves the smallest cost possible for a specific loss function $V : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ on the data distribution μ_D, i.e.,

$$f^* \in \arg\min_{f \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mu_D}[V(f(x), y)]. \tag{1.1}$$

Since μ_D is generally unknown, we resort to estimate the best predictor using training data. A unifying formulation is the *(penalized) risk minimization* framework:

$$\bar{f} \in \arg\min_{f \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mu_{\mathcal{A}}}[V(f(x), y)] + \lambda\Omega(f), \tag{1.2}$$

where $\bar{f}$ is our estimate and $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is the input-output pair used for training, which is drawn from some distribution $\mu_{\mathcal{A}}$ supposedly relevant to μ_D. We denote $S = \{(x_i, y_i)\}_{i=1}^n$ as the raw training set, consisting of independently and identically distributed (i.i.d.) data sampled from μ_D.[1] We also denote μ_S as the empirical data distribution, and $\mathcal{A} : \mu_S \mapsto \mu_{\mathcal{A}}$ as the (stochastic) transformations applied to μ_S to generate $\mu_{\mathcal{A}}$. The processed training data is sampled from $\mu_{\mathcal{A}}$ and then fed into the training algorithm. $\Omega : \mathcal{H} \to \mathbb{R}_+$ is called the *regularizer* which assigns some non-negative cost to each f. The constant λ trades off the effect of data-fitting and regularization.

[1] This is the standard assumption in the *supervised learning* setting. Studying data augmentation in a weaker setting, for example in *transfer learning*, is beyond the scope of this thesis.

In most cases, f is parametrized by $w \in \mathcal{W}$ where $\mathcal{W}$ is called the *parameter space*, and Problem (1.2) is transformed into an optimization problem over its parameters

$$\bar{w} \in \arg\min_{w \in \mathcal{W}} \mathbb{E}_{(x,y)\sim\mu_{\mathcal{A}}}[\mathcal{L}(w; x, y)] + \lambda g(w). \tag{1.3}$$

Problem (1.3) is the starting point for our study in this thesis. We are interested in solving the *optimization* problem (i.e., efficiently finding the $\arg\min$ solution), and understanding the *learning* problem (i.e., studying the statistical properties of our solution on the data distribution μ_D).

In some settings, Equation (1.3) is well studied. For example, when $\mathcal{W}$ is a convex set in a vector space, and both $\mathcal{L}$ and g are convex w.r.t. w, the optimization problem reduces to stochastic/finite-sum convex optimization and is well-studied; if $\mu_{\mathcal{A}} = \mu_S$, then the learning problem is called *(penalized) empirical risk minimization* and is the focus of statistical learning theory. However, in general, these assumptions may not be met. For example, the parameters may need to satisfy nonlinear constraints that correspond to a manifold, making it hard to develop and analyze the optimization algorithms. Moreover, in practice $\mu_{\mathcal{A}}$ is often different from μ_S, invalidating a key assumption of statistical learning theory.

In this thesis, I study three settings where the above assumptions break down:

(i) the optimization problem (1.3) where $\mathcal{W}$ is a Riemannian manifold, henceforth called *Riemannian optimization*;

(ii) the optimization problem (1.3) where $\mathcal{L}$ is the composition of a loss function with a neural network with additive skip connections, henceforth called *deep residual networks*;

(iii) the learning problem (1.3) where a nontrivial training data distribution $\mathcal{A}$ is used, henceforth called *data augmentation*.

1.2 Background

The goal of this section is to expose an interested reader to the relevant background that leads to my thesis work. In particular, I will introduce the concepts of Rie-

mannian manifold and Riemannian optimization, touching on a few important data science problems that can be naturally cast as optimization on Riemannian manifolds. Then I will move to some recent developments in optimizing and regularizing neural networks. General related work will be summarized while more specific references will be discussed in later chapters. Readers familiar with the general background should feel confident to skip this section if they wish to.

1.2.1 Riemannian Geometry

An Intuitive Example. Human are small creatures compared with their habitat, the Planet Earth. We experience the physical space as a three dimensional Euclidean space and the earth ground as a two dimensional surface within. The earth ground poses interesting challenges to human investigation — at the scale of our body ($10^0 - 10^1$ meters), it often looks flat; at the scale of a hill ($10^3 - 10^4$ meters), it often presents interesting curvatures; at the scale of a sea ($10^6 - 10^7$ meters), as human come to realize, it starts to reveal its beautiful spherical shape. Comparing the earth ground at different scales with a flat surface, a curious explorer can find both important similarities and astonishing differences, when it comes to basic geometric concepts such as lines, angles, distances and triangles. The earth surface, with an intuitive definition of distance and angle, is an example of *Riemannian manifold*.

With an intuitive mathematical object in mind, we now state the formal definitions, adapted from [Jost, 2011, Kühnel, 2015].

Definition 1.1 (manifold). A *manifold* M of dimension d is a connected paracompact Hausdorff space for which every point has a neighborhood U that is homeomorphic to an open subset S of $\mathbb{R}^d$. Such a homeomorphism $x : U \to S$ is called a *(coordinate) chart*. An *atlas* is a family $\{U_\alpha, x_\alpha\}$ of charts for which the U_α constitute an open covering of M.

Definition 1.2 (tangent space). Let $p \in U \subset \mathcal{M}$. Define the equivalent class $\{(x, v) : x : U \to \mathcal{M}' \text{ is a chart}, v \in T_{x(p)}\mathcal{M}'\}$, then one can show $(x, v) \sim (y, w) \iff w = d(y \circ x^{-1})v$. The space of equivalence classes is called the *tangent space* to $\mathcal{M}$ at the

point p, and is denoted by $T_p\mathcal{M}$. Let $\pi : T\mathcal{M} \to \mathcal{M}$ with $\pi(w) = p$ for $w \in T_p\mathcal{M}$ be the projection onto the "base point", the triple $(T\mathcal{M}, \pi, \mathcal{M})$ is called the *tangent bundle* of $\mathcal{M}$ and $T\mathcal{M}$ is called the *total space* of the tangent bundle.

With the tangent space structure defined on differentiable manifolds, we now move on to define lengths and angles using the tangent vectors. For this purpose, we introduce the notion of Riemannian manifolds.

Definition 1.3 (Riemannian metric)**.** A *Riemannian metric* on a differentiable manifold $\mathcal{M}$ is given by a scalar product on each tangent space $T_p\mathcal{M}$ which depends smoothly (i.e., C^∞) on the base point p. A *Riemannian manifold* is a differentiable manifold, equipped with a Riemannian metric.

In local coordinates $x = (x^1, \ldots, x^d)$, a Riemannian metric is represented by a positive definite, symmetric matrix $(g_{ij}(x))_{i,j=1,\ldots,d}$, where the coefficients depend smoothly on x. The inner product of two tangent vectors $v, w \in T_p\mathcal{M}$ with coordinate representations $(v^1, \ldots, v^d)$ and $(w^1, \ldots, w^d)$ is $\langle v, w\rangle \triangleq g_{ij}(x(p))v^i w^j$. In particular, $\langle \frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\rangle = g_{ij}$, and length of v is given by $\|v\| \triangleq \langle v, v\rangle^{\frac{1}{2}}$. When p is parametrized by $t \in \mathbb{R}$, we also write $\dot{x}^i(t) \triangleq \frac{d}{dt}(x^i(p(t)))$.

By the Picard-Lindelöf Theorem on the local existenece and uniqueness of an ODE solution, we have the following theorem:

Theorem 1.1. *Let $\mathcal{M}$ be a Riemannian manifold, $p \in \mathcal{M}, v \in T_p\mathcal{M}$. Then there exist $\epsilon > 0$ and precisely one geodesic $c : [0, \epsilon] \to \mathcal{M}$ with $c(0) = p, \dot{c}(0) = v$. In addition, c depends smoothly on p and v.*

We remark that by reparametrizing time, i.e., using $c_v(t) \equiv c_{\lambda v}(\frac{t}{\lambda})$ for $\lambda > 0, t \in [0, \epsilon]$, one can show there exists $\epsilon_0 > 0$ such that for any $w \in T_p\mathcal{M}$ with $\|w\| \leq \epsilon_0$, c_w is defined at least on $[0, 1]$.

The *angle* between two geodesics emanating from $p \in \mathcal{M}$ is defined as the angle between their respective velocity vectors in the tangent space $T_p\mathcal{M}$. A *geodesic triangle* is defined by its three vertices (distinct points in a manifold) and the three geodesics connecting them.

The next concept is important for both theoretical analysis and numerical implementation:

Definition 1.4 (exponential map). The *exponential map* of a Riemannian manifold $\mathcal{M}$ at $p \in \mathcal{M}$ is defined by $\text{Exp}_p : V_p \to \mathcal{M}, v \mapsto c_v(1)$ where $V_p \triangleq \{v \in T_p\mathcal{M} : c_v \text{ is defined on } [0,1]\}$.

It can be shown that the exponential map is a diffeomorphism. In particular, $d\text{Exp}_p(0) = id_{|T_p\mathcal{M}}$ is the identity map in the tangent space.

Definition 1.5 (parallel transport). Let $\gamma : [0,1] \to \mathcal{M}$ be a continuous differentiable curve on a Riemannian manifold, and $Y_0 \in T_{\gamma(0)}\mathcal{M}$ a tangent vector. There exists a unique vector field Y along γ such that Y is parallel (along γ). Y is called the *parallel transport* (or parallel displacement) of Y_0 along γ.

In this work, we always consider parallel transport along geodesics. Let $p = \gamma(a), q = \gamma(b)$ and $X \in T_p\mathcal{M}$, we denote the parallel transport of X along γ to q as $\Gamma_p^q X$. It can be shown that parallel transport preserves the inner product, i.e., $\langle X, Y\rangle_p \equiv \langle \Gamma_p^q X, \Gamma_p^q Y\rangle_q$.

Let f be a smooth scalar function on a Riemannian manifold $\mathcal{M}$. We use $X(f)$ to denote the scalar field generated by computing the derivative of f in the direction of X. The covariant derivative of f is nothing but the differential $df = Df = \nabla f$ with $\nabla f(X) = \nabla_X f = X(f)$. We further define:

Definition 1.6 (Riemannian gradient). The *gradient* of f with respect to a metric g, written as $\text{grad}_g f$ or simply $\text{grad} f$, is the vector determined by the relation $g(\text{grad} f, X) \triangleq \nabla f(X)$ for all X.

Definition 1.7 (Riemannian Hessian). The second covariant derivative (also called *Hessian*) of f is given by $\nabla^2 f = \nabla\nabla f$. In particular, $(\nabla^2 f)(X,Y) \triangleq (\nabla_X \nabla f)(Y) \triangleq \nabla_X(\nabla f(Y)) - \nabla f(\nabla_X Y) = \nabla_X \nabla_Y f - (\nabla_X Y)(f)$.

Now we introduce what is perhaps the most intriguing concept in Riemannian geometry — curvature. In later chapters, we will see that the existence of metric

curvature often poses a significant difficulty for analyzing Riemannian optimization algorithms. We have

Definition 1.8 (Lie bracket). Let X, Y be differentiable vector fields on $\mathcal{M}$, and let $f : \mathcal{M} \to \mathbb{R}$ be a differentiable function. The *Lie bracket* of X, Y is defined as the vector field $[X, Y]$ such that $[X, Y](f) \triangleq X(Y(f)) - Y(X(f))$.

Intuitively, the Lie bracket measures the degree of non-commutativity of the derivatives, characterized by the change of Y in the direction of X. It can be shown that the Lie bracket is related to the covariant derivative defined previously: $[X, Y] \equiv \nabla_X Y - \nabla_Y X$.

Definition 1.9 (curvature tensor). The *curvature tensor* is a $(1, 3)$-tensor defined as

$$R(X, Y)Z \triangleq \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z.$$

The following concept, called *sectional curvature*, is critical for our analysis of Riemannian optimization algorithms.

Definition 1.10 (sectional curvature). Define the *standard curvature tensor*

$$R_1(X, Y)Z \triangleq \langle Y, Z\rangle X - \langle X, Z\rangle Y$$

where $\langle \cdot, \cdot \rangle$ denotes a Riemannian metric. Let

$$\kappa_1(X, Y) \triangleq \langle R_1(X, Y)Y, X\rangle = \langle X, X\rangle\langle Y, Y\rangle - \langle X, Y\rangle^2,$$
$$\kappa(X, Y) \triangleq \langle R(X, Y)Y, X\rangle,$$

and $\sigma \subset T_p\mathcal{M}$ be a two-dimensional subspace spanned by X, Y. Then the quantity

$$K_\sigma \triangleq \frac{\kappa(X, Y)}{\kappa_1(X, Y)}$$

is called the *sectional curvature* of the Riemannian manifold w.r.t. the plane σ.

Intuitively, the sectional curvature of a plane σ at a point $p \in \mathcal{M}$ is the Gauss curvature at the point $p \in \mathcal{N}$ where $\mathcal{N}$ is a two-dimensional submanifold of $\mathcal{M}$ induced by the restriction of $\mathrm{Exp}_p : T_p\mathcal{M} \to \mathcal{M}$ to the plane σ. For interested readers, [Ollivier, 2011] provides further geometric intuition of curvature and related concepts in Riemannian geometry.

1.2.2 Riemannian Optimization

An Intuitive Example. Suppose the owner of a telegraph company would like to build a station to connect to cities across the continent. For simplicity, consider only four cities – Boston, Seattle, San Francisco and Miami. The goal is to find the best location for the station such that the sum of the distances to the four cities (hence the cost of cables) is minimized. We can formulate this problem as (approximately) finding a point on a Riemannian manifold (a two-dimensional sphere) that minimizes a loss function (the sum of four geodesic distances), i.e.,

$$\min_{p \in \mathbb{S}^2} \sum_{i=1}^{4} d(p, q_i)$$

The problem appears simple, especially since it would become a convex optimization problem if the feasible domain were a convex set and the distance in use were Euclidean; however, one cannot readily apply the convex optimization toolkit, due to the nonlinear constraint and non-Euclidean distance function. Such problems indicate challenges in analyzing Riemannian optimization, which will be probed in details in later chapters. For now, we need to introduce a few more definitions.

As Riemannian gradients live in tangent spaces of a Riemannian manifold, we need a sensible way to map a gradient vector in the tangent space to a new iterate (point) on the manifold. The exponential map defined earlier serves exactly this purpose. In practice, however, a more general concept is often used to construct optimization algorithms.

Definition 1.11 (retraction). A *retraction* on a manifold $\mathcal{M}$ is a smooth mapping R from the tangent bundle $T\mathcal{M}$ onto $\mathcal{M}$. Let R_x denote the restriction of R to $T_x\mathcal{M}$. A retraction has the following properties:

(i) $R_x(0_x) = x$, where 0_x denotes the zero element of $T_x\mathcal{M}$.

(ii) With the canonical identification $T_{0_x}T_x\mathcal{M} \simeq T_x\mathcal{M}$, R_x satisfies $DR_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$, where $\mathrm{id}_{T_x\mathcal{M}}$ denotes the identity mapping on $T_x\mathcal{M}$.

Intuitively, a retraction is a first-order approximation to the exponential map. Next we define function classes on Riemannian manifolds that generalize their

vector space counterparts.

Definition 1.12 (geodesically convex). A subset $\mathcal{X} \subset \mathcal{M}$ is called a *geodesically convex set* if given any two points in $\mathcal{X}$, there is a minimizing geodesic contained within $\mathcal{X}$ that joins the two points. A function $f : \mathcal{X} \to \mathbb{R}$ is called a *geodesically convex function* if for any geodesic $\gamma : [0,1] \to \mathcal{X}$ and $\forall t \in [0,1]$, it holds that $f(\gamma(t)) \leq (1-t)f(\gamma(0)) + tf(\gamma(1))$.

Note that this definition does not assume f is differentiable. In fact, it can be shown that, within a set $\mathcal{X}$ where the exponential map is bijective, an equivalent definition is that there exists $g_x \in T_x\mathcal{M}$, such that $f(y) \geq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y)\rangle$ for all $y \in \mathcal{X}$. (See, e.g, the proof of Proposition 1.4 in [Greene and Shiohama, 1981].)

Following the convention in vector space, we call g_x a subgradient of f at x.

Definition 1.13 (geodesically Lipschitz). A function $f : \mathcal{M} \supset \mathcal{X} \to \mathbb{R}$ is called *geodesically $L_{(0)}$-Lipschitz* if for any two points $x, y \in \mathcal{X}$, $|f(x) - f(y)| \leq L_{(0)}d(x,y)$.

The definition of smoothness requires comparing Riemannian gradients from different tangent spaces using parallel transport.

Definition 1.14 (geodesically smooth). A differentiable function $f : \mathcal{M} \supset \mathcal{X} \to \mathbb{R}$ is called *geodesically $L_{(1)}$-smooth*, if its gradient is geodesically $L_{(1)}$-Lipschitz, that is, $\|\mathrm{grad}f(x) - \Gamma_y^x \mathrm{grad}f(y)\| \leq L_{(1)}d(x,y)$ for any $x, y \in \mathcal{X}$.

Example 1.1 (noncompact Stiefel manifold, $\mathbb{R}_*^{n\times p}$ [Absil et al., 2009b]). Let $\mathbb{R}_*^{n\times p}(p \leq n)$ denote the set of all $n \times p$ matrices whose columns are linearly independent. Note that its complement, $\{X \in \mathbb{R}^{n\times p} : \det(X^TX) = 0\}$ is a closed set. Therefore, $\mathbb{R}_*^{n\times p}$ with the chart $\varphi : \mathbb{R}_*^{n\times p} \to \mathbb{R}^{np} : X \mapsto \mathrm{vec}(X)$ is an open submanifold of $\mathbb{R}^{n\times p}$, called the *noncompact Stiefel manifold* of full-rank $n \times p$ matrices.

When $p = 1$, the noncompact Stiefel manifold reduces to the Euclidean space $\mathbb{R}^n$ with the origin removed. When $p = n$, the noncompact Stiefel manifold becomes the general linear group GL_n.

Example 1.2 (real projective space, $\mathbb{RP}^{n-1}$ [Absil et al., 2009b]). The real projective space $\mathbb{RP}^{n-1}$ is the set of all directions in $\mathbb{R}^n$, i.e., the set of all straight lines passing through the origin of $\mathbb{R}^n$. Let $\mathbb{R}^n_* \triangleq \mathbb{R}^n - \{0\}$ denote the Euclidean space $\mathbb{R}^n$ with the origin removed (that is also a noncompact Stiefel manifold $\mathbb{R}^{n\times p}_*$ with $p = 1$. The real projective space can be identified with the quotient $\mathbb{R}^n_*/\sim$, where the equivalence relation is defined by $x \sim y \iff \exists t \in \mathbb{R}_* : y = xt$, so we write $\mathbb{RP}^{n-1} \simeq \mathbb{R}^n_*/\sim$.

Historical Notes. Optimization on Riemannian manifolds was first studied in [Edelman et al., 1998, Ferreira and Oliveira, 1998, 2002, Gabay, 1982, Smith, 1994, Udriste, 1994]. Matrix manifolds are the most studied manifolds for geometric optimization due to their ubiquitous presence in engineering and statistics—see [Absil et al., 2009b] for a detailed introduction. Several standard nonlinear optimization methods have been generalized to manifold optimization, including Newton's [Absil et al., 2004, 2009a, 2014, Smith, 1994], trust-region [Absil and Gallivan, 2009, Absil et al., 2007, Baker et al., 2008, Huang et al., 2014], line search algorithms [Absil and Gallivan, 2009], gradient-descent [Samir et al., 2012], subgradient method [Borckmans et al., 2013] and preconditioning [Boumal and Absil, 2015]. Among applications of Riemannian optimization, low-rank manifolds have enjoyed great recent interest. Absil and Oseledets [2014] surveys different retraction methods on low-rank manifolds. Others have focused on PSD manifolds [Sra, 2012, Sra and Hosseini, 2013], in particular for solving geodesically convex problems [Hosseini and Sra, 2015b, Sra and Hosseini, 2015]. Bonnabel [2013] is the first to show asymptotic convergence of stochastic gradient methods on Riemannian manifolds.

Our work on first-order methods in geodesically convex optimization [Zhang and Sra, 2016] (Chapter 2) is the first general non-asymptotic convergence results for Riemannian optimization in the literature. In parallel with our work on Riemannian variance reduced incremental gradient method [Zhang et al., 2016] (Chapter 3), Kasai et al. [2016] studied the same idea, albeit restricted to Grassmann manifold. Liu et al. [2017] studied how to acceleate Riemannian gradient descent

given oracles to a nonlinear equation solver, however constructing such oracles is as difficult as solving the original optimization problem, even in the Euclidean case. Instead, the algorithm we construct (Chapter 4) is a computationally tractable generalization of the famous Nesterov's accelerated gradient method, which can provably accelerate Riemannian gradient descent around the minimizer within a radius depending on the manifold curvatures.

The other line of work [Boumal et al., 2016a] attempts to circumvent the difficulty of analyzing curved space by making assumptions about the pullback of the objective function in the tangent space. Sufficient descent and other established proof techniques in the vector space can thus be transferred for use in the analysis. The drawback of this approach is that little can be said about the properties (e.g. smoothness constant) of the pullback function, as they, also, are inevitably tied to the manifold curvatures. Therefore, work in this direction typically resorts to line search methods to ensure the sufficient descent condition. Finally, other recent work [Bento et al., 2017, Hu et al., 2018, Jiang et al., 2017, Zhang and Zhang, 2018] also follow one of the two proof approaches, but studies general retraction, vector transport, proximal gradient, or quasi-Newton methods.

1.2.3 Regularization in Deep Learning

Regularization has been a very powerful tool in statistical machine learning. Classical statistical models, such as least squares or Gaussian mixture models, typically introduce regularizers in the optimization objective to ensure the obtained solution has certain desired properties, especially when the system is underdetermined such that the optimal solution to the original problem is not unique [Friedman et al., 2001]. Among these models, two notable examples are *compressive sensing* [Candès and Wakin, 2008] and *support vector machines* (SVM) [Boser et al., 1992]. These algorithms are able to generate good solutions despite that their original problems are prone to overfitting. They have made significant theoretical and practical impacts.

However, it gradually becomes obvious that the set of regularizers developed for classical models is rather restricted. As increasingly large models try to absorb the information from increasingly large datasets and satiate the ever-increasing computing power, practitioners of deep learning have discovered many new techniques to regularize their colossal neural networks. These techniques fit into four categories, namely *regularizers*, *noise*, *loss functions* and *data augmentation*. We now examine them one by one.

1.2.3.1 Regularizer

Similar to other machine learning models, commonly used regularizers in deep learning include L_2 weight penalty (corresponding to Gaussian prior in maximum *a posteriori* (MAP) estimation), L_1 weight penalty (also known as the least absolute shrinkage and selection operator (Lasso), corresponding to Laplace prior in MAP estimation), and Grouped Lasso which combines inter-group sparsity (i.e., Laplace) prior and intra-group small norm (i.e., Gaussian) prior [Goodfellow et al., 2016]. Many other regularizers have been proposed, including penalty for deviating from weight orthogonality [Cisse et al., 2017], hinge loss for promoting large margin solutions [Elsayed et al., 2018], etc.

1.2.3.2 Loss function

Regularization may also come from properly defining a new objective function that encodes the desired property of the learned representation. For example, the standard multi-class classification problem features the cross entropy as the loss function, which is susceptible to overfitting. In label smoothing [Pereyra et al., 2017], the target labels are smoothed out so that weights in the network are not pushed towards infinity. In adversarial training [Madry et al., 2017], a network is trained not on the raw inputs, but on the most vulnerable points in their neighborhood to enhance model robustness. Loss functions that operate on latent representation rather than final outputs, such as the Siamese loss [Chopra et al., 2005]

and the triplet loss [Schroff et al., 2015], provide additional supervision to the representation learning process, and prove to work well even under great intra-class variability.

1.2.3.3 Noise

It is hypothesized and empirically supported that certain level of noise in the parameter gradients helps to improve model generalization, presumably because such noise prevents the model parameters from getting stuck in a sharp local minimum, and thereby must finally reaching a flat minimum at equilibrium. While the notion of flat minimum is still a half-baked concept, it does capture some intuitive explanation about generalization.

Some of the most successful regularization techniques for deep learning seem to benefit from certain noise structures in the parameter updates. For example, it is found that training with stochastic gradient descent (SGD) using large step size for an extended amount of time is often important for best test performance; some methods introduce noise to the hidden layer activations, including dropout, dropconnect, stochastic depth, dropblock, as well as various activation normalization methods, including batch normalization, layer normalization, group normalization, etc.; other methods perturb the backpropagation process to introduce structural noise in the gradients, such as Shake-Shake and ShakeDrop regularization. The mixup method I propose in Chapter 5 can also be seen as adding structural noise in the training process, albeit at the input and target levels instead. While noise can be injected in different places, it is important to note that they all take effects by perturbing the direction and scale of the parameter updates.

1.2.3.4 Data augmentation

In the machine learning context, data augmentation refers to the practice of supplementing the learning algorithm with synthesized training data, typically by modifying the raw training data via label-preserving transforms. There is no established

rule for designing data augmentation methods, however it is widely believed that the augmented data should appear to human as plausible samples from the data distribution μ_D. Data augmentation lies at the heart of all successful applications of deep learning, ranging from image classification [Krizhevsky et al., 2012] to speech recognition [Amodei et al., 2016, Graves et al., 2013]. In all cases, substantial domain knowledge is leveraged to design suitable data transformations leading to improved generalization. In image classification, for example, one routinely uses rotation, translation, cropping, resizing, flipping [Lecun et al., 2001, Simonyan and Zisserman, 2015], and random erasing [Zhong et al., 2017] to enforce visually plausible invariances in the model through the training data. Similarly, in speech recognition, noise injection is a prevalent practice to improve the robustness and accuracy of the trained models [Amodei et al., 2016].

1.2.4 Optimization in Deep Learning

Optimization in the classical literature mostly focuses on developing efficient optimization algorithms (i.e., optimizers) and their convergence analysis under various (yet typically very general) assumptions. However, as neural network models become the predominant objective functions in machine learning, the relevance of some classical results is challenged. In particular, some of the most successful neural networks are non-smooth, non-convex functions with combinatorially many stationary points and local minimums, which seems hopeless to solve according to classical theory. On the other hand, additional structures in the network function, such as layerwise composition and skip connection, call for a new theory of optimization that is aware of these architectural assumptions.

Despite having relatively little theoretical understanding, significant empirical progress has been made during the last decade on training neural networks. These results can be roughly categorized into *optimizers*, *initializations*, *normalizations* and *architectures*.

1.2.4.1 Optimizers

A deep neural network is almost always trained by a minibatch stochastic optimization algorithm. The most popular optimizer choices are first-order methods, including SGD with momentum, and adaptive gradient methods such as Adagrad [Duchi et al., 2011], RMSprop [Tieleman and Hinton, 2012] and Adam [Kingma and Ba, 2014]. Recently, it is reported that layerwise normalized gradient methods help training in settings of poor initialization or extremely large minibatches. Inspired by the theory of quasi-Newton methods for optimizing non-convex smooth functions, quasi-Newton style algorithms for training neural networks have also been explored, among which the Kronecker-factorized approximate curvature (K-FAC) algorithm [Martens and Grosse, 2015] is a particular promising variant that balances between adapting to function curvature and efficient computation.

1.2.4.2 Initializations

Only a decade ago, multi-layer neural networks were still believed to be very hard to train beyond a few layers. The blame commonly fell on a problem called exploding and vanishing gradients, which made training hard to proceed from the very beginning. The first practitioners of deep learning exploited unsupervised pre-training to initialize the network weights and found that it helped the supervised training (a.k.a. fine-tuning) phase [Hinton et al., 2006]. Later however, it was realized that the costly pre-training phase was unnecessary, and could be replaced by a proper random initialization. Perhaps the two most influential neural network weight initialization techniques are Xavier (a.k.a. Glorot) initialization [Glorot and Bengio, 2010] and Kaiming (a.k.a. He or MSR-A) initialization [He et al., 2015]. Both initialization methods are designed to approximately preserve the output (or gradient) variance of a feed-forward network across layers – Xavier initialization works for nonlinearities such as sigmoid and tanh, whereas Kaiming initialization is specifically tailored for ReLU activations. With either pre-training or proper initialization, networks up to 30 layers were able to set the new state-of-the-art in

computer vision [He et al., 2015, Krizhevsky et al., 2012, Simonyan and Zisserman, 2015].

1.2.4.3 Normalizations

Normalization is another set of techniques that greatly helps training. The earliest practice dates back to divisive normalization such as in [Lyu and Simoncelli, 2008, Pinto et al., 2008], however arguably the most influential and successful technique is batch normalization [Ioffe and Szegedy, 2015]. With batch normalization layers, Ioffe and Szegedy [2015] reported being able to train deep (10+ layers) networks with much larger learning rate, less careful initialization, and observed faster convergence than training vanilla networks. Other normalization techniques, including weight normalization [Salimans and Kingma, 2016], layer normalization [Ba et al., 2016], instance normalization [Ulyanov et al., 2016], group normalization [Wu and He, 2018] etc. have also been found useful in applications ranging from neural style transfer, object detection to machine translation. Further discussion about the history of normalization methods can be found in Section 6.7.4.

1.2.4.4 Architectures

Finally, perhaps the most important component for the successful training of a deep neural network is its model architecture. It has long been known that introducing gating mechanisms into the recurrent neural network structure helps to solve the exploding and vanishing gradient problems [Hochreiter and Schmidhuber, 1997]. Srivastava et al. [2015] applied similar ideas for training convolutional neural networks and observed that it greatly improves the training of deeper networks – their 900 layer *highway networks* reportedly can be trained at ease. He et al. [2016a] and their follow-up work [He et al., 2016b] got rid of the gating mechanisms by introducing additive identity connections across layers, and using (batch) normalization techniques to control the scale of the activations. Their proposed network architecture, *residual networks*, is by far the most influential model in com-

puter vision at this point. Another notable innovation is the *densely connected networks* [Huang et al., 2017], which replaces the additive identity connections with an averaging operation. Importantly, as we will argue, this change greatly alleviates the exploding gradient problem without normalization.

1.3 Main Results

This work contributes to the study of nonconvex optimization and learning from two perspectives — (1) theoretical analysis of first-order optimization with Riemannian manifold constraints, and (2) practical techniques for improving the generalization and optimization of nonconvex models, specifically deep neural networks. As a brief preview, I list some of the key results developed in later chapters:

1.3.1 Riemannian optimization

- I develop the first iteration complexity analysis for general first-order geodesically convex optimization problems. Our analyses cover Riemannian gradient, subgradient, stochastic gradient and proximal gradient methods under various settings. (Chapter 2)
- I develop the first variance reduced incremental gradient method and its iteration complexity analysis for Riemannian optimization, which enjoys faster convergence than both Riemannian gradient descent and stochastic gradient methods, and is well-suited for data science applications. (Chapter 3)
- I develop the first tractable Riemannian optimization algorithm that locally enjoys Nesterov-style acceleration, i.e., requires fewer number of iterations to converge than Riemannian gradient descent. (Chapter 4)

1.3.2 Deep neural networks

- I develop mixup, a simple yet effective replacement to the empirical risk minimization objective, for classification on high-dimensional inputs with the cross-

entropy loss function. mixup can be seen as a data augmentation method or data-dependent regularization. Training with the mixup objective improves the generalization and robustness of deep neural networks on various tasks, with virtually no overhead. (Chapter 5)

- I develop ZeroInit, a simple yet effective replacement to the normalization modules in deep residual networks, by properly rescaling the standard initialization. Training residual networks with ZeroInit is stable with the default learning rate schedule, and as fast as training with normalization. With proper regularization, our method matches state-of-the-art performance on benchmark datasets in image classification and machine translation, with no normalization and fewer parameters. (Chapter 6)

In the final chapter, I will zoom out to fit my work into the global picture of nonconvex optimization and learning for data science and artificial intelligence applications. I will also discuss unsolved challenges and exciting new possibilities for the field to move forward. (Chapter 7)

2

Iteration Complexity of Riemannian (Sub)gradient Methods

Geodesic convexity generalizes the notion of (vector space) convexity to nonlinear metric spaces. But unlike convex optimization, geodesically convex (g-convex) optimization is much less developed. In this chapter, we contribute to the understanding of g-convex optimization by developing iteration complexity analysis for several first-order algorithms on Hadamard manifolds. Specifically, we prove upper bounds for the global complexity of deterministic and stochastic (sub)gradient methods for optimizing smooth and nonsmooth g-convex functions, both with and without strong g-convexity. Our analysis also reveals how the manifold geometry, especially *sectional curvature*, impacts convergence rates.

2.1 Introduction

Convex optimization is fundamental to numerous areas including machine learning. Convexity often helps guarantee polynomial runtimes and enables robust, more stable numerical methods. But almost invariably, the use of convexity in machine learning is limited to vector spaces, even though convexity *per se* is not limited to vector spaces. Most notably, it generalizes to *geodesically convex* metric spaces [Bridson and Haefliger, 1999, Burago et al., 2001, Gromov, 1978], through which it offers a much richer setting for developing mathematical models amenable to global optimization.

Our specific focus in this chapter is on contributing to the understanding of geodesically convex (g-convex) optimization, besides the broader goal of increasing awareness about g-convexity (see Definition 2.1). In particular, we study first-order algorithms for smooth and nonsmooth g-convex optimization, for which we prove iteration complexity upper bounds. Except for a fundamental lemma that applies to general g-convex metric spaces, we limit our discussion to Hadamard manifolds (Riemannian manifolds with global nonpositive curvature), as they offer the most convenient grounds for generalization[1] while also being relevant to numerous applications (see e.g., Section 2.1.1).

Specifically, we study optimization problems of the form

$$\min \quad f(x) \qquad \text{such that} \quad x \in \mathcal{X} \subset \mathcal{M}, \tag{2.1}$$

where $f : \mathcal{M} \to \mathbb{R} \cup \{\infty\}$ is a proper g-convex function, $\mathcal{X}$ is a geodesically convex set and $\mathcal{M}$ is a Hadamard manifold [Bishop and O'Neill, 1969, Gromov, 1978]. We solve equation 2.1 via first-order methods under a variety of settings analogous to the Euclidean case: nonsmooth, Lipschitz-smooth, and strongly g-convex. We present results for both deterministic and stochastic (where $f(x) = \mathbb{E}[F(x, \xi)]$) g-

[1]Hadamard manifolds have unique geodesics between any two points. This key property ensures that the exponential map is a global diffeomorphism. Unique geodesics also make it possible to generalize notions such as convex sets and convex functions. (Compact manifolds such as spheres, do not admit globally geodesically convex functions other than the constant function; local g-convexity is possible, but that is a separate topic).

convex optimization.

Although Riemannian geometry provides tools that enable generalization of Euclidean algorithms [Absil et al., 2009b, Udriste, 1994], to obtain iteration complexity bounds we must overcome some fundamental geometric hurdles. We introduce key results that overcome some of these hurdles, and pave the way to analyzing first-order g-convex optimization algorithms.

2.1.1 Related work and motivating examples

We recollect below a few items of related work and some examples relevant to machine learning, where g-convexity and more generally Riemannian optimization play an important role.

Standard references on Riemannian optimization are [Absil et al., 2009b, Udriste, 1994], but these primarily consider problems on manifolds without necessarily assuming g-convexity. Consequently, their analysis is limited to asymptotic convergence (except for [Theorem 4.2, Udriste, 1994] that proves linear convergence for functions with positive-definite and bounded Riemannian Hessians). The recent monograph [Bacák, 2014] is devoted to g-convexity and g-convex optimization on geodesic metric spaces, though without any attention to global complexity analysis. Bacák [2014] also details a noteworthy application: averaging trees in the geodesic metric space of phylogenetic trees [Billera et al., 2001].

At a more familiar level, implicitly the topic of "geometric programming" [Boyd et al., 2007] may be viewed as a special case of g-convex optimization [Sra and Hosseini, 2015]. For instance, computing stationary states of Markov chains (e.g., while computing PageRank) may be viewed as g-convex optimization problems by placing suitable geometry on the positive orthant; this idea has a fascinating extension to nonlinear iterations on convex cones (in Banach spaces) endowed with the structure of a geodesic metric space [Lemmens and Nussbaum, 2012].

Perhaps the most important example of such metric spaces is the set of positive definite matrices viewed as a Riemannian or Finsler manifold; a careful study of

this setup was undertaken by Sra and Hosseini [2015]. They also highlighted applications to maximum likelihood estimation for certain non-Gaussian (heavy- or light-tailed) distributions, resulting in various g-convex and nonconvex likelihood problems; see also [Wiesel, 2012, Zhang et al., 2013]. However, none of these three works presents a global convergence rate analysis for their algorithms.

There exist several nonconvex problems where Riemannian optimization has proved quite useful, e.g., low-rank matrix and tensor factorization [Ishteva et al., 2011, Mishra et al., 2013, Vandereycken, 2013]; dictionary learning [Harandi et al., 2012, Sun et al., 2015]; optimization under orthogonality constraints [Edelman et al., 1998, Liu et al., 2015, Moakher, 2002, Shen et al., 2009]; and Gaussian mixture models [Hosseini and Sra, 2015a], for which g-convexity helps accelerate manifold optimization to substantially outperform the Expectation Maximization (EM) algorithm.

2.1.2 Contributions

We summarize the main contributions of this chapter below.

- We develop a new inequality (Lemma 2.1) useful for analyzing the behavior of optimization algorithms for functions in Alexandrov space with curvature bounded below, which can be applied to (not necessarily g-convex) optimization problems on Riemannian manifolds and beyond.
- For g-convex optimization problems on Hadamard manifolds (Riemannian manifolds with global nonpositive sectional curvature), we prove iteration complexity upper bounds for several existing algorithms (Table 2.1). For the special case of smooth geodesically strongly convex optimization, a prior linear convergence result that uses line-search is known [Udriste, 1994]; our results do not require line search. Moreover, as far as we are aware, ours are the first global complexity results for general g-convex optimization.

[2]Here for simplicity only the dependencies on c and t are shown, while other factors are considered constant and thus omitted. Please refer to the theorems for complete results.

[3]"Yes": result holds for proper averaging of the iterates; "No": result holds for the last iterate.

f	Algorithm	Stepsize	Rate[2]	Averaging[3]	Theorem
g-convex, Lipschitz	projected subgradient	$\frac{D}{L_f\sqrt{ct}}$	$O\left(\sqrt{\frac{c}{t}}\right)$	Yes	2.1
g-convex, bounded subgradient	projected stochastic subgradient	$\frac{D}{G\sqrt{ct}}$	$O\left(\sqrt{\frac{c}{t}}\right)$	Yes	2.2
g-strongly convex, Lipschitz	projected subgradient	$\frac{2}{\mu(s+1)}$	$O\left(\frac{c}{t}\right)$	Yes	2.3
g-strongly convex, bounded subgradient	projected stochastic subgradient	$\frac{2}{\mu(s+1)}$	$O\left(\frac{c}{t}\right)$	Yes	2.4
g-convex, smooth	projected gradient	$\frac{1}{L_g}$	$O\left(\frac{c}{c+t}\right)$	No	2.5
g-convex, smooth bounded variance	projected stochastic gradient	$\frac{1}{L_g+\frac{\sigma}{D}\sqrt{ct}}$	$O\left(\frac{c+\sqrt{ct}}{c+t}\right)$	Yes	2.6
g-strongly convex, smooth	projected gradient	$\frac{1}{L_g}$	$O\left(\left(1-\min\left\{\frac{1}{c},\frac{\mu}{L_g}\right\}\right)^t\right)$	No	2.7

Table 2.1: Summary of results. This table summarizes the non-asymptotic convergence rates we have proved for various geodesically convex optimization algorithms. s: iterate index; t: total number of iterates; D: diameter of domain; L_f: Lipschitz constant of f; c: a constant dependent on D and on the sectional curvature lower bound κ; G: upper bound of gradient norms; μ: strong convexity constant of f; L_g: Lipschitz constant of the gradient; σ: square root variance of the gradient.

2.2 Background

Before we describe the algorithms and analyze their properties, we would like to introduce some concepts in metric geometry and Riemannian geometry that generalize concepts in Euclidean space. For a more detailed treatise of the background, please refer to Section 1.2.1 or a standard text [e.g. Jost, 2011].

2.2.1 Metric Geometry

For generalization of nonlinear optimization methods to metric space, we now recall some basic concepts in metric geometry, which cover vector spaces and Riemannian manifolds as special cases. A *metric space* is a pair (X, d) of set X and distance function d that satisfies positivity, symmetry, and the triangle inequality [Burago et al., 2001]. A continuous mapping from the interval $[0, 1]$ to X is called a *path*. The *length* of a path $\gamma : [0, 1] \to X$ is defined as $\text{length}(\gamma) := \sup \sum_{i=1}^{n} d(\gamma(t_{i-1}), \gamma(t_i))$, where the supremum is taken over the set of all partitions $0 = t_0 < \cdots < t_n = 1$ of the interval $[0, 1]$, with an arbitrary $n \in \mathbb{N}$. A metric space is a *length space* if for any $x, y \in X$ and $\epsilon > 0$ there exists a path $\gamma : [0, 1] \to X$ joining x and y such that $\text{length}(\gamma) \leq d(x, y) + \epsilon$. A path $\gamma : [0, 1] \to X$ is called a *geodesic* if it is parametrized by the arc length. If every two points $x, y \in X$ are connected by a geodesic, we say (X, d) is a *geodesic space*. If the geodesic connecting every $x, y \in X$ is unique, the space is called *uniquely geodesic* [Bacák, 2014].

The properties of *geodesic triangles* will be central to our analysis of optimization algorithms. A geodesic triangle $\triangle pqr$ with vertices $p, q, r \in X$ consists of three geodesics $\overline{pq}, \overline{qr}, \overline{rp}$. Given $\triangle pqr \in X$, a *comparison triangle* $\triangle \bar{p}\bar{q}\bar{r}$ in k-plane is a corresponding triangle with the same side lengths in two-dimensional space of constant Gaussian curvature k. A length space with curvature bound is called an Alexandrov space. The notion of angle is defined in the following sense. Let $\gamma : [0, 1] \to X$ and $\eta : [0, 1] \to X$ be two geodesics in (X, d) with $\gamma_0 = \eta_0$, we define the *angle* between γ and η as $\alpha(\gamma, \eta) := \limsup_{s,t \to 0_+} \measuredangle \bar{\gamma}_s \bar{\gamma}_0 \bar{\eta}_t$ where $\measuredangle \bar{\gamma}_s \bar{\gamma}_0 \bar{\eta}_t$ is the

Please refer to the theorems for complete results.

angle at $\bar{\gamma}_0$ of the corresponding triangle $\triangle\bar{\gamma}_s\bar{\gamma}_0\bar{\eta}_t$. We use Toponogov's theorem to relate the angles and lengths of any geodesic triangle in a geodesic space to those of a comparison triangle in a space of constant curvature [Burago et al., 2001, 1992].

2.2.2 Riemannian Geometry

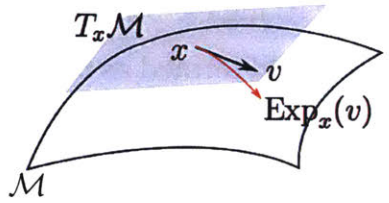


Figure 2-1: Illustration of a manifold. Also shown are tangent space, geodesic and exponential map.

An n-dimensional *manifold* is a topological space where each point has a neighborhood that is homeomorphic to the n-dimensional Euclidean space. At any point x on a manifold, tangent vectors are defined as the tangents of parametrized curves passing through x. The *tangent space* $T_x\mathcal{M}$ of a manifold $\mathcal{M}$ at x is defined as the set of all tangent vectors at the point x. An exponential map at $x \in \mathcal{M}$ is a mapping from the tangent space $T_x\mathcal{M}$ to $\mathcal{M}$ with the requirement that a vector $v \in T_x\mathcal{M}$ is mapped to the point $y := \mathrm{Exp}_x(v) \in \mathcal{M}$ such that there exists a geodesic $\gamma : [0, 1] \to \mathcal{M}$ satisfying $\gamma(0) = x, \gamma(1) = y$ and $\gamma'(0) = v$.

As tangent vectors at two different points $x, y \in \mathcal{M}$ lie in different tangent spaces, we cannot compare them directly. To meaningfully compare vectors in different tangent spaces, one needs to define a way to move a tangent vector along the geodesics, while "preserving" its length and orientation. We thus need to use an inner product structure on tangent spaces, which is called a Riemannian metric. A *Riemannian manifold* $(\mathcal{M}, \mathfrak{g})$ is a real smooth manifold equipped with an inner product $\mathfrak{g}_x$ on the tangent space $T_x\mathcal{M}$ of every point x, such that if u, v are two vector fields on $\mathcal{M}$ then $x \mapsto \langle u, v\rangle_x := \mathfrak{g}_x(u, v)$ is a smooth function. On a Riemannian manifold, the notion of *parallel transport* (parallel displacement) provides a sensible

way to transport a vector along a geodesic. Intuitively, a tangent vector $v \in T_x\mathcal{M}$ at x of a geodesic γ is still a tangent vector $\Gamma(\gamma)_x^y v$ of γ after being transported to a point y along γ. Furthermore, parallel transport preserves inner products, i.e. $\langle u, v\rangle_x = \langle \Gamma(\gamma)_x^y u, \Gamma(\gamma)_x^y v\rangle_y$.

The curvature of a Riemannian manifold is characterized by its Riemannian metric tensor at each point. For worst-case analysis, it is sufficient to consider the trigonometry of geodesic triangles. *Sectional curvature* is the Gauss curvature of a two dimensional submanifold formed as the image of a two dimensional subspace of a tangent space after exponential mapping. A two dimensional submanifold with positive, zero or negative sectional curvature is locally isometric to a two dimensional sphere, a Euclidean plane, or a hyperbolic plane with the same Gauss curvature.

2.2.3 Function Classes on a Riemannian Manifold

We first define some key terms. $\mathcal{X} \subset \mathcal{M}$ is called a geodesically convex set if for any $x, y \in \mathcal{X}$ the minimal distance geodesic γ connecting x, y lies within $\mathcal{X}$. Throughout the paper, we assume that the function f is defined on a Riemannian manifold $\mathcal{M}$, f assumes at least a global minimum point within $\mathcal{X}$, and $x^* \in \mathcal{X}$ is a minimizer of f, unless stated otherwise.

Definition 2.1 (Geodesic convexity). A function $f : \mathcal{M} \to \mathbb{R}$ is said to be geodesically convex if for any $x, y \in \mathcal{M}$, a geodesic γ such that $\gamma(0) = x$ and $\gamma(1) = y$, and $t \in [0, 1]$, it holds that

$$f(\gamma(t)) \leq (1 - t)f(x) + tf(y).$$

It can be shown that an equivalent definition for geodesic convexity is that for any $x, y \in \mathcal{M}$, there exists a tangent vector $g_x \in T_x\mathcal{M}$ such that

$$f(y) \geq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y)\rangle_x,$$

and g_x is called a *subgradient* of f at x, or the *gradient* if f is differentiable, and $\langle \cdot, \cdot \rangle_x$ denotes the inner product in the tangent space of x induced by the Riemannian

metric. In the rest of the paper we will omit the index of tangent space when it is clear from the context.

Definition 2.2 (Strong convexity). A function $f : \mathcal{M} \to \mathbb{R}$ is said to be geodesically μ-strongly convex if for any $x, y \in \mathcal{M}$,

$$f(y) \geq f(x) + \langle g_x, \operatorname{Exp}_x^{-1}(y) \rangle_x + \frac{\mu}{2} d^2(x, y).$$

or, equivalently, for any geodesic γ such that $\gamma(0) = x, \gamma(1) = y$ and $t \in [0, 1]$,

$$f(\gamma(t)) \leq (1 - t)f(x) + tf(y) - \frac{\mu}{2} t(1 - t) d^2(x, y).$$

Definition 2.3 (Lipschitzness). A function $f : \mathcal{M} \to \mathbb{R}$ is said to be geodesically L_f-Lipschitz if for any $x, y \in \mathcal{M}$,

$$|f(x) - f(y)| \leq L_f d(x, y).$$

Definition 2.4 (Smoothness). A differentiable function $f : \mathcal{M} \to \mathbb{R}$ is said to be geodesically L_g-smooth if its gradient is L_g-Lipschitz, i.e. for any $x, y \in \mathcal{M}$,

$$\|g_x - \Gamma_y^x g_y\| \leq L_g d(x, y)$$

where Γ_y^x is the parallel transport from y to x.

Observe that compared to the Euclidean setup, the above definition requires a parallel transport operation to "transport" g_y to g_x. It can be proved that if f is L_g-smooth, then for any $x, y \in \mathcal{M}$,

$$f(y) \leq f(x) + \langle g_x, \operatorname{Exp}_x^{-1}(y) \rangle_x + \frac{L_g}{2} d^2(x, y).$$

2.3 Convergence Rates of First-order Methods

In this section, we analyze the global complexity of deterministic and stochastic gradient methods for optimizing various classes of g-convex functions on Hadamard manifolds. We assume access to a projection oracle $\mathcal{P}_{\mathcal{X}}$ that maps a point $x \in \mathcal{M}$ to $\mathcal{P}_{\mathcal{X}}(x) \in \mathcal{X} \subset \mathcal{M}$ such that

$$d(x, \mathcal{P}_{\mathcal{X}}(x)) < d(x, y), \quad \forall y \in \mathcal{X} \setminus \{\mathcal{P}_{\mathcal{X}}(x)\},$$

where $\mathcal{X}$ is a geodesically convex set and $\max_{y,z\in\mathcal{X}} d(y,z) \leq D$. General projected subgradient / gradient algorithms on Riemannian manifolds take the form

$$x_{s+1} = \mathcal{P}_{\mathcal{X}}(\mathrm{Exp}_{x_s}(-\eta_s g_s)), \tag{2.2}$$

where s is the iterate index, g_s is a subgradient of the objective function, and η_s is a step-size. For brevity, we will use the word 'gradient' to refer to both subgradient and gradient, deterministic or stochastic; the meaning should be apparent from the context.

While it is easy to translate first-order optimization algorithms from Euclidean space to Riemannian manifolds, and similarly to prove asymptotic convergence rates (since locally Riemannian manifolds resemble Euclidean space), it is much harder to carry out *non-asymptotic* analysis, at least due to the following two difficulties:

- **Non-Euclidean trigonometry is difficult to use.** Trigonometric geometry in nonlinear spaces is fundamentally different from Euclidean space. In particular, for analyzing optimization algorithms, the law of cosines in Euclidean space

  $$a^2 = b^2 + c^2 - 2bc\cos(A), \tag{2.3}$$

  where a, b, c are the sides of a Euclidean triangle with A the angle between sides b and c, is an essential tool for bounding the squared distance between the iterates and the minimizer(s). Indeed, consider the Euclidean update $x_{s+1} = x_s - \eta_s g_s$. Applying (2.3) to the triangle $\triangle x_s x x_{s+1}$, with $a = \overline{x x_{s+1}}$, $b = \overline{x_s x_{s+1}}$, $c = \overline{x x_s}$, and $A = \measuredangle x x_s x_{s+1}$, we get the frequently used formula

  $$\|x_{s+1} - x\|^2 = \|x_s - x\|^2 - 2\eta_s\langle g_s, x_s - x\rangle + \eta_s^2\|g_s\|^2$$

  However, this nice equality does not exist for nonlinear spaces.

- **Linearization does not work.** Another key technique used in bounding squared distances is inspired by the proximal algorithms. Here, gradient-like updates are seen as proximal steps for minimizing a series of *linearizations* of the objective function. Specifically, let $\psi(x; x_s) = f(x_s) + \langle g_s, x - x_s\rangle$ be the lin-

earization of the convex function f, and let $g_s \in \partial f(x_s)$. Then, $x_{s+1} = x_s - \eta_s g_s$ is the unique solution to the following minimization problem

$$\min_x \left\{ \psi(x; x_s) + \frac{1}{2\eta_s} \|x - x_s\|^2 \right\}.$$

Since $\psi(x; x_s)$ is convex, we thus have (see e.g. Tseng [2009]) the recursively useful bound

$$\psi(x_{s+1}; x_s) + \frac{1}{2\eta_s} \|x_{s+1} - x\|^2 \leq \psi(x; x_s) + \frac{1}{2\eta_s} \|x_s - x\|^2 - \frac{\eta_s}{2} \|g_s\|^2.$$

But in nonlinear space there is no trivial analogy of a linear function. For example, for any given $y \in \mathcal{M}$ and $g_y \in T_y\mathcal{M}$, the function

$$\psi(x; y) = f(y) + \langle g_y, \operatorname{Exp}_y^{-1}(x) \rangle,$$

is geodesically both star-concave and star-convex in y, but neither convex nor concave in general. Thus a nonlinear analogue of the above result does not hold.

We address the first difficulty by developing an easy-to-use trigonometric distance bound for Alexandrov space with curvature bounded below. When specialized to Hadamard manifolds, our result reduces to the analysis in [Bonnabel, 2013], which in turn relies on [Cordero-Erausquin et al., 2001, Lemma 3.12]. However, unlike [Cordero-Erausquin et al., 2001], our proof assumes no manifold structure on the geodesic space of interest, and is fundamentally different in its techniques.

2.3.1 Trigonometric Distance Bound

As noted above, a main hurdle in analyzing non-asymptotic convergence of first-order methods in geodesic spaces is that the Euclidean law of cosines does not hold any more. For general nonlinear spaces, there are no corresponding analytical expressions. Even for the (hyperbolic) space of constant negative curvature -1, perhaps the simplest and most studied nonlinear space, the law of cosines is replaced by the *hyperbolic law of cosines*:

$$\cosh a = \cosh b \cosh c - \sinh b \sinh c \cos(A), \tag{2.4}$$

which does not align well with the standard techniques of convergence rate analysis. With the goal of developing analysis for nonlinear space optimization algorithms, our first contribution is the following trigonometric distance bound for Alexandrov space with curvature bounded below. Owing to its fundamental nature, we believe that this lemma may be of broader interest too.

Lemma 2.1. *If a, b, c are the sides (i.e., side lengths) of a geodesic triangle in an Alexandrov space with curvature lower bounded by κ, and A is the angle between sides b and c, then*

$$a^2 \leq \frac{\sqrt{|\kappa|}c}{\tanh(\sqrt{|\kappa|}c)} b^2 + c^2 - 2bc\cos(A). \tag{2.5}$$

Proof. **sketch.** The complete proof contains technical details that digress from the main focus of the main text, so we leave them in the end of this chapter. Below we sketch the main steps.

Our first observation is that by the famous Toponogov's theorem [Burago et al., 2001, 1992], we can upper bound the side lengths of a geodesic triangle in an Alexandrov space with curvature bounded below by the side lengths of a comparison triangle in the hyperbolic plane, which satisfies (*cf.* equation 2.4):

$$\cosh(\sqrt{|\kappa|}a) = \cosh(\sqrt{|\kappa|}b)\cosh(\sqrt{|\kappa|}c) - \sinh(\sqrt{|\kappa|}b)\sinh(\sqrt{|\kappa|}c)\cos(A). \tag{2.6}$$

Second, we observe that it suffices to study $\kappa = -1$, which corresponds to equation 2.4, since Eqn. (2.6) can be seen as Eqn. equation 2.4 with side lengths $a = \sqrt{|\kappa|}a', b = \sqrt{|\kappa|}b', c = \sqrt{|\kappa|}c'$ (see Lemma 2.2).

Finally, we observe that in equation 2.4, $\frac{\partial^2}{\partial b^2}\cosh(a) = \cosh(a)$. Letting $g(b, c, A) := \cosh(\sqrt{\text{rhs}(b, c, A)})$, where $\text{rhs}(b, c, A)$ is the right hand side of equation 2.5, we then see that it is sufficient to prove the following:

1. $\cosh(a)$ and $g(b, c, A)$ are equal at $b = 0$.
2. the first partial derivatives of $\cosh(a)$ and $g(b, c, A)$ w.r.t. b agree at $b = 0$.
3. $\frac{\partial^2}{\partial b^2} g(b, c, A) \geq g(b, c, A)$ for $b, c \geq 0$ (Lemma 2.3).

These three steps, if true, lead to the proof of $\cosh(a) \leq g(b, c, A)$ for $b, c \geq 0$, thus proving a special case of Lemma 2.1 for space with constant sectional curvature -1

as shown in Lemma 2.4, 2.5. Combing this special case with our first two observations concludes the proof of the lemma. □

Remark 2.1. Inequality (2.5) provides an upper bound on the side lengths of a geodesic triangle in an Alexandrov space with curvature bounded below. Some examples of such spaces are Riemannian manifolds, including hyperbolic space, Euclidean space, sphere, orthogonal groups, and compact sets on a PSD manifold. However, our derivation does not rely on any manifold structure, thus it also applies to certain cones and convex hypersurfaces [Burago et al., 2001].

We now recall a lemma showing that metric projection in Hadamard manifold is nonexpansive.

Lemma 2.2 (Bacák [2014]). *Let $(\mathcal{M}, \mathfrak{g})$ be a Hadamard manifold. Let $\mathcal{X} \subset \mathcal{X}$ be a closed convex set. Then the mapping $\mathcal{P}_{\mathcal{X}}(x) := \{y \in \mathcal{X} : d(x,y) = \inf_{z\in\mathcal{X}} d(x,z)\}$ is single-valued and nonexpansive, that is, we have for every $x, y \in \mathcal{M}$*

$$d(\mathcal{P}_{\mathcal{X}}(x), \mathcal{P}_{\mathcal{X}}(y)) \leq d(x,y).$$

In the sequel, we use the notation $\zeta(\kappa, c) \triangleq \frac{\sqrt{|\kappa|}c}{\tanh(\sqrt{|\kappa|}c)}$ for the curvature dependent quantity from inequality equation 2.5. From Lemma 2.1 and Lemma 2.2 it is straightforward to prove the following corollary, which characterizes an important relation between two consecutive updates of an iterative optimization algorithm on Riemannian manfiold with curvature bounded below.

Corollary 2.1. *For any Riemannian manifold $\mathcal{M}$ where the sectional curvature is lower bounded by κ and any point x, $x_s \in \mathcal{X}$, the update $x_{s+1} = \mathcal{P}_{\mathcal{X}}(\mathrm{Exp}_{x_s}(-\eta_s g_s))$ satisfies*

$$\langle -g_s, \mathrm{Exp}_{x_s}^{-1}(x)\rangle \leq \frac{1}{2\eta_s}\left(d^2(x_s,x) - d^2(x_{s+1},x)\right) + \frac{\zeta(\kappa, d(x_s,x))\eta_s}{2}\|g_s\|^2. \tag{2.7}$$

Proof. Denote $\tilde{x}_{s+1} := \mathrm{Exp}_{x_s}(-\eta_s g_s)$. Note that for the geodesic triangle $\triangle x_s \tilde{x}_{s+1} x$, we have $d(x_s, \tilde{x}_{s+1}) = \eta_s\|g_s\|$, while $d(x_s, \tilde{x}_{s+1})d(x_s,x)\cos(\measuredangle \tilde{x}_{s+1}x_s x) = \langle -\eta_s g_s, \mathrm{Exp}_{x_s}^{-1}(x)\rangle$. Now let $a = \overline{\tilde{x}_{s+1}x}, b = \overline{\tilde{x}_{s+1}x_s}, c = \overline{x_s x}, A = \measuredangle \tilde{x}_{s+1}x_s x$, apply Lemma 2.1 and simplify to obtain

$$\langle -g_s, \mathrm{Exp}_{x_s}^{-1}(x)\rangle \leq \frac{1}{2\eta_s}\left(d^2(x_s,x) - d^2(\tilde{x}_{s+1},x)\right) + \frac{\zeta(\kappa, d(x_s,x))\eta_s}{2}\|g_s\|^2,$$

whereas by Lemma 2.2 we have $d^2(\tilde{x}_{s+1}, x) \geq d^2(x_{s+1}, x)$, which then implies (2.7). □

It is instructive to compare equation 2.7 with its Euclidean counterpart (for which actually $\zeta = 1$):

$$\langle -g_s, x - x_s \rangle = \frac{1}{2\eta_s} \left(\|x_s - x\|^2 - \|x_{s+1} - x\|^2 \right) + \frac{\eta_s}{2} \|g_s\|^2.$$

Note that as long as the diameter of the domain and the sectional curvature remain bounded, ζ is bounded, and we get a meaningful bound in a form similar to the law of cosines in Euclidean space.

Corollary 2.1 furnishes the missing tool for analyzing non-asymptotic convergence rates of manifold optimization algorithms. We now move to the analysis of several such first-order algorithms.

2.3.2 Convergence Rate Analysis

Nonsmooth convex optimization. The following two theorems show that both deterministic and stochastic subgradient methods achieve a *curvature-dependent* $O(1/\sqrt{t})$ rate of convergence for g-convex on Hadamard manifolds.

Theorem 2.1. *Let f be g-convex and L_f-Lipschitz, the diameter of domain be bounded by D, and the sectional curvature lower-bounded by $\kappa \leq 0$. Then, the projected subgradient method with a constant stepsize $\eta_s = \eta = \frac{D}{L_f \sqrt{\zeta(\kappa, D)t}}$ and $\overline{x}_1 = x_1, \overline{x}_{s+1} = \operatorname{Exp}_{\overline{x}_s} \left(\frac{1}{s+1} \operatorname{Exp}_{\overline{x}_s}^{-1}(x_{s+1}) \right)$ satisfies*

$$f\left(\overline{x}_t\right) - f(x^*) \leq D L_f \sqrt{\frac{\zeta(\kappa, D)}{t}}.$$

Proof. Since f is g-convex, it satisfies $f(x_s) - f(x^*) \leq \langle -g_s, \operatorname{Exp}_{x_s}^{-1}(x^*) \rangle$, which combined with Corollary 2.1 and the L_f-Lipschitz condition yields the upper bound

$$f(x_s) - f(x^*) \leq \frac{1}{2\eta} \left(d^2(x_s, x^*) - d^2(x_{s+1}, x^*) \right) + \frac{\zeta(\kappa, D) L_f^2 \eta}{2}. \tag{2.8}$$

Summing over s from 1 to t and dividing by t, we obtain

$$\frac{1}{t} \sum_{s=1}^{t} f(x_s) - f(x^*) \leq \frac{1}{2t\eta} \left(d^2(x_1, x^*) - d^2(x_{t+1}, x^*) \right) + \frac{\zeta(\kappa, D) L_f^2 \eta}{2}. \tag{2.9}$$

Plugging in $d(x_1, x^*) \leq D$ and $\eta = \frac{D}{L_f\sqrt{\zeta(\kappa,D)t}}$ we further obtain

$$\frac{1}{t}\sum_{s=1}^{t} f(x_s) - f(x^*) \leq DL_f\sqrt{\frac{\zeta(\kappa, D)}{t}}.$$

It remains to show that $f(\overline{x}_t) \leq \frac{1}{t}\sum_{s=1}^{t} f(x_s)$, which can be proved by an easy induction. □

We note that Theorem 2.1 and our following results are all generalizations of known results in Euclidean space. Indeed, setting curvature $\kappa = 0$ we can recover the Euclidean convergence rates (in some cases up to a difference in small constant factors). However, for Hadamard manifolds $\kappa < 0$ and the theorem implies that the algorithms may converge more slowly. Also worth noting is that we must be careful in how we obtain the "average" iterate $\overline{x}_t$ on the manifold.

Theorem 2.2. *If f is g-convex, the diameter of the domain is bounded by D, the sectional curvature of the manifold is lower bounded by $\kappa \leq 0$, and the stochastic subgradient oracle satisfies $\mathbb{E}[\widetilde{g}(x)] = g(x) \in \partial f(x), \mathbb{E}[\|\widetilde{g}_s\|^2] \leq G^2$, then the projected stochastic subgradient method with stepsize $\eta_s = \eta = \frac{D}{G\sqrt{\zeta(\kappa,D)t}}$, and $\overline{x}_1 = x_1, \overline{x}_{s+1} = \mathrm{Exp}_{\overline{x}_s}\left(\frac{1}{s+1}\mathrm{Exp}_{\overline{x}_s}^{-1}(x_{s+1})\right)$ satisfies the upper bound*

$$\mathbb{E}[f(\overline{x}_t) - f(x^*)] \leq DG\sqrt{\frac{\zeta(\kappa, D)}{t}}.$$

Proof. The proof structure is very similar, except that for each equation we take expectation with respect to the sequence $\{x_s\}_{s=1}^t$. Since f is g-convex, we have

$$\mathbb{E}[f(x_s) - f(x^*)] \leq \langle -\mathbb{E}[\widetilde{g}_s], \mathrm{Exp}_{x_s}^{-1}(x^*)\rangle,$$

which combined with Corollary 2.1 and $\mathbb{E}[\|\widetilde{g}_s\|^2] \leq G^2$ yields

$$\mathbb{E}[f(x_s) - f(x^*)] \leq \frac{1}{2\eta}\mathbb{E}\left[d^2(x_s, x^*) - d^2(x_{s+1}, x^*)\right] + \frac{\zeta(\kappa, D)G^2\eta}{2}. \tag{2.10}$$

Now arguing as in Theorem 2.1 the proof follows. □

Strongly convex nonsmooth functions. The following two theorems show that both subgradient method and stochastic subgradient method achieve a curvature

dependent $O(1/t)$ rate of convergence for g-strongly convex functions on Hadamard manifolds.

Theorem 2.3. *If f is geodesically μ-strongly convex and L_f-Lipschitz, and the sectional curvature of the manifold is lower bounded by $\kappa \leq 0$, then the projected subgradient method with $\eta_s = \frac{2}{\mu(s+1)}$ satisfies*

$$f\left(\overline{x}_t\right) - f(x^*) \leq \frac{2\zeta(\kappa, D)L_f^2}{\mu(t+1)},$$

where $\overline{x}_1 = x_1$, and $\overline{x}_{s+1} = \operatorname{Exp}_{\overline{x}_s}\left(\frac{2}{s+1}\operatorname{Exp}_{\overline{x}_s}^{-1}(x_{s+1})\right)$.

Proof. Since f is geodesically μ-strongly convex, we have

$$f(x_s) - f(x^*) \leq \langle -g_s, \operatorname{Exp}_{x_s}^{-1}(x^*)\rangle - \frac{\mu}{2}d^2(x_s, x^*),$$

which combined with Corollary 2.1 and L_f-Lipschitz condition yields

$$f(x_s) - f(x^*) \leq \left(\frac{1}{2\eta_s} - \frac{\mu}{2}\right)d^2(x_s, x^*) - \frac{1}{2\eta_s}d^2(x_{s+1}, x^*) + \frac{\zeta(\kappa, D)L_f^2\eta_s}{2} \tag{2.11}$$

$$= \frac{\mu(s-1)}{4}d^2(x_s, x^*) - \frac{\mu(s+1)}{4}d^2(x_{s+1}, x^*) + \frac{\zeta(\kappa, D)L_f^2}{\mu(s+1)}. \tag{2.12}$$

Multiply (2.12) by s and sum over s from 1 to t; then divide the result by $\frac{t(t+1)}{2}$ to obtain

$$\frac{2}{t(t+1)}\sum_{s=1}^{t} sf(x_s) - f(x^*) \leq \frac{2\zeta(\kappa, D)L_f^2}{\mu(t+1)}. \tag{2.13}$$

The final step is to show $f(\overline{x}_t) \leq \frac{2}{t(t+1)}\sum_{s=1}^{t} sf(x_s)$, which again follows by an easy induction. □

Theorem 2.4. *If f is geodesically μ-strongly convex, the sectional curvature of the manifold is lower bounded by $\kappa \leq 0$, and the stochastic subgradient oracle satisfies $\mathbb{E}[\widetilde{g}(x)] = g(x) \in \partial f(x), \mathbb{E}[\|\widetilde{g}_s\|^2] \leq G^2$, then the projected subgradient method with $\eta_s = \frac{2}{\mu(s+1)}$ satisfies*

$$\mathbb{E}[f\left(\overline{x}_t\right) - f(x^*)] \leq \frac{2\zeta(\kappa, D)G^2}{\mu(t+1)}$$

where $\overline{x}_1 = x_1$, and $\overline{x}_{s+1} = \operatorname{Exp}_{\overline{x}_s}\left(\frac{2}{s+1}\operatorname{Exp}_{\overline{x}_s}^{-1}(x_{s+1})\right)$.

Proof. The proof structure is very similar to the previous theorem, except that now

we take expectations over the sequence $\{x_s\}_{s=1}^t$. We leave the details to the appendix for brevity. □

Theorems 2.3 and 2.4 are generalizations of their Euclidean counterparts [Lacoste-Julien et al., 2012], and follow the same proof structures. Our upper bounds depend linearly on $\zeta(\kappa, D)$, which implies that with $\kappa < 0$ the algorithms may converge more slowly. However, note that for strongly convex problems, the distances from iterates to the minimizer are shrinking, thus the inequality (2.11) (or its stochastic version) may be too pessimistic, and better dependency on κ may be obtained with a more refined analysis. We leave this as an open problem for the future.

Smooth convex optimization. The following two theorems show that gradient descent algorithm achieves a curvature dependent $O(1/t)$ rate of convergence, whereas stochastic gradient achieves a curvature dependent $O(1/t + \sqrt{1/t})$ rate for smooth g-convex functions on Hadamard manifolds.

Theorem 2.5. *If $f : \mathcal{M} \to \mathbb{R}$ is g-convex with an L_g-Lipschitz gradient, the diameter of domain is bounded by D, and the sectional curvature of the manifold is bounded below by κ, then projected gradient descent with $\eta_s = \eta = \frac{1}{L_g}$ satisfies for $t > 1$ the upper bound*

$$f(x_t) - f(x^*) \leq \frac{\zeta(\kappa, D) L_g D^2}{2(\zeta(\kappa, D) + t - 2)}.$$

Proof. For simplicity we denote $\Delta_s = f(x_s) - f(x^*)$. First observe that with $\eta = \frac{1}{L_g}$ the algorithm is a descent method. Indeed, we have

$$\Delta_{s+1} - \Delta_s \leq \langle g_s, \operatorname{Exp}_{x_s}^{-1}(x_{s+1}) \rangle + \frac{L_g}{2} d^2(x_{s+1}, x_s) = -\frac{\|g_s\|^2}{2L_g}. \tag{2.14}$$

On the other hand, by the convexity of f and Corollary 2.1 we obtain

$$\Delta_s \leq \langle -g_s, \operatorname{Exp}_{x_s}^{-1}(x^*) \rangle \leq \frac{L_g}{2} \left(d^2(x_s, x^*) - d^2(x_{s+1}, x^*) \right) + \frac{\zeta(\kappa, D) \|g_s\|^2}{2L_g}. \tag{2.15}$$

Multiplying (2.14) by $\zeta(\kappa, D)$ and adding to (2.15), we get

$$\zeta(\kappa, D)\Delta_{s+1} - (\zeta(\kappa, D) - 1)\Delta_s \leq \frac{L_g}{2} \left(d^2(x_s, x^*) - d^2(x_{s+1}, x^*) \right). \tag{2.16}$$

Now summing over s from 1 to $t-1$, a brief manipulation shows that

$$\zeta(\kappa, D)\Delta_t + \sum_{s=2}^{t-1} \Delta_s \leq (\zeta(\kappa, D) - 1)\Delta_1 + \tfrac{L_g D^2}{2}. \tag{2.17}$$

Since for $s \leq t$ we proved $\Delta_t \leq \Delta_s$, and by assumption $\Delta_1 \leq \frac{L_g D^2}{2}$, for $t > 1$ we get

$$\Delta_t \leq \frac{\zeta(\kappa, D) L_g D^2}{2(\zeta(\kappa, D) + t - 2)},$$

yielding the desired bound. □

Theorem 2.6. *If $f : \mathcal{M} \to \mathbb{R}$ is g-convex with L_g-Lipschitz gradient, the diameter of domain is bounded by D, the sectional curvature of the manifold is bounded below by κ, and the stochastic gradient oracle satisfies $\mathbb{E}[\widetilde{g}(x)] = g(x) = \nabla f(x), \mathbb{E}[\|\nabla f(x) - \widetilde{g}_s\|^2] \leq \sigma^2$, then the projected stochastic gradient algorithm with $\eta_s = \eta = \frac{1}{L_g + 1/\alpha}$ where $\alpha = \frac{D}{\sigma}\sqrt{\frac{1}{\zeta(\kappa, D)t}}$ satisfies for $t > 1$*

$$\mathbb{E}[f(\overline{x}_t) - f(x^*)] \leq \frac{\zeta(\kappa, D) L_g D^2 + 2D\sigma\sqrt{\zeta(\kappa, D)t}}{2(\zeta(\kappa, D) + t - 2)},$$

where $\overline{x}_2 = x_2$, $\overline{x}_{s+1} = \mathrm{Exp}_{\overline{x}_s}\left(\frac{1}{s}\mathrm{Exp}^{-1}_{\overline{x}_s}(x_{s+1})\right)$ for $2 \leq s \leq t-2$, $\overline{x}_t = \mathrm{Exp}_{\overline{x}_{t-1}}\left(\frac{\zeta(\kappa, D)}{\zeta(\kappa, D)+t-2}\mathrm{Exp}^{-1}_{\overline{x}_{t-1}}(x_t)\right)$

Proof. As before we write $\Delta_s = f(x_s) - f(x^*)$. First we observe that

$$\Delta_{s+1} - \Delta_s \leq \langle g_s, \mathrm{Exp}^{-1}_{x_s}(x_{s+1})\rangle + \frac{L_g}{2} d^2(x_{s+1}, x_s) \tag{2.18}$$

$$= \langle \widetilde{g}_s, \mathrm{Exp}^{-1}_{x_s}(x_{s+1}\rangle + \langle g_s - \widetilde{g}_s, \mathrm{Exp}^{-1}_{x_s}(x_{s+1})\rangle + \frac{L_g}{2} d^2(x_{s+1}, x_s) \tag{2.19}$$

$$\leq \langle \widetilde{g}_s, \mathrm{Exp}^{-1}_{x_s}(x_{s+1})\rangle + \frac{\alpha}{2}\|g_s - \widetilde{g}_s\|^2 + \frac{1}{2}\left(L_g + \frac{1}{\alpha}\right) d^2(x_{s+1}, x_s) \tag{2.20}$$

Taking expectation, and letting $\eta = \frac{1}{L_g + 1/\alpha}$, we obtain

$$\mathbb{E}[\Delta_{s+1} - \Delta_s] \leq \frac{\alpha\sigma^2}{2} - \frac{\mathbb{E}[\|\widetilde{g}_s\|^2]}{2\left(L_g + \frac{1}{\alpha}\right)}. \tag{2.21}$$

On the other hand, using convexity of f and Corollary 2.1 we get

$$\Delta_s \leq \langle -g_s, \mathrm{Exp}^{-1}_{x_s}(x^*)\rangle \leq \frac{L_g + \frac{1}{\alpha}}{2}\mathbb{E}\left[d^2(x_s, x^*) - d^2(x_{s+1}, x^*)\right] + \frac{\zeta(\kappa, D)\mathbb{E}[\|\widetilde{g}_s\|^2]}{2\left(L_g + \frac{1}{\alpha}\right)}. \tag{2.22}$$

Multiply (2.21) by $\zeta(\kappa, D)$ and add to (2.22), we get

$$\mathbb{E}[\zeta(\kappa, D)\Delta_{s+1} - (\zeta(\kappa, D) - 1)\Delta_s] \leq \frac{L_g + \frac{1}{\alpha}}{2}\mathbb{E}\left[d^2(x_s, x^*) - d^2(x_{s+1}, x^*)\right] + \frac{\alpha\zeta(\kappa, D)\sigma^2}{2}.$$

Summing over s from 1 to $t-1$ and simplifying, we obtain

$$\mathbb{E}[\zeta(\kappa,D)\Delta_t+\sum_{s=2}^{t-1}\Delta_s] \leq \mathbb{E}[(\zeta(\kappa,D)-1)\Delta_1]+\frac{L_gD^2}{2}+\frac{1}{2}\left(\frac{D^2}{\alpha}+\alpha\zeta(\kappa,D)t\sigma^2\right). \quad (2.23)$$

Now set $\alpha = \frac{D}{\sigma\sqrt{\zeta(\kappa,D)t}}$, and note that $\Delta_1 \leq \frac{L_gD^2}{2}$; thus, for $t>1$ we get

$$\mathbb{E}\big[\zeta(\kappa,D)\Delta_t + \textstyle\sum_{s=2}^{t-1}\Delta_s\big] \leq \frac{\zeta(\kappa,D)L_gD^2}{2} + D\sigma\sqrt{\zeta(\kappa,D)t}.$$

Finally, due to g-convexity of f it is easy to verify by induction that

$$\mathbb{E}[f(\overline{x}_t) - f(x^*)] \leq \frac{\mathbb{E}[\zeta(\kappa,D)\Delta_t + \sum_{s=2}^{t-1}\Delta_s]}{\zeta(\kappa,D)+t-2}.$$

□

Smooth and strongly convex functions. Next we prove that gradient descent achieves a curvature dependent linear rate of convergence for geodesically strongly convex and smooth functions on Hadamard manifolds.

Theorem 2.7. *If $f:\mathcal{M}\to\mathbb{R}$ is geodesically μ-strongly convex with L_g-Lipschitz gradient, and the sectional curvature of the manifold is bounded below by κ, then the projected gradient descent algorithm with $\eta_s=\eta=\frac{1}{L_g}$, $\epsilon=\min\{\frac{1}{\zeta(\kappa,D)},\frac{\mu}{L_g}\}$ satisfies for $t>1$*

$$f(x_t)-f(x^*) \leq \frac{(1-\epsilon)^{t-2}L_gD^2}{2}.$$

Proof. As before we use $\Delta_s = f(x_s)-f(x^*)$. Observe that with $\eta=\frac{1}{L_g}$ we have descent:

$$\Delta_{s+1}-\Delta_s \leq \langle g_s, \operatorname{Exp}_{x_s}^{-1}(x_{s+1})\rangle + \frac{L_g}{2}d^2(x_{s+1},x_s) = -\frac{\|g_s\|^2}{2L_g}. \quad (2.24)$$

On the other hand, by the strong convexity of f and Corollary 2.1 we obtain the bounds

$$\Delta_s \leq \langle -g_s, \operatorname{Exp}_{x_s}^{-1}(x^*)\rangle - \frac{\mu}{2}d^2(x_t,x^*) \quad (2.25)$$

$$\leq \frac{L_g-\mu}{2}d^2(x_s,x^*) - \frac{L_g}{2}d^2(x_{s+1},x^*) + \frac{\zeta(\kappa,D)\|g_s\|^2}{2L_g}. \quad (2.26)$$

Multiply (2.24) by $\zeta(\kappa,D)$ and add to (2.25) to obtain

$$\zeta(\kappa,D)\Delta_{s+1} - (\zeta(\kappa,D)-1)\Delta_s \leq \frac{L_g-\mu}{2}d^2(x_s,x^*) - \frac{L_g}{2}d^2(x_{s+1},x^*) \quad (2.27)$$

Let $\epsilon = \min\{\frac{1}{\zeta(\kappa,D)}, \frac{\mu}{L_g}\}$, multiply (2.27) by $(1-\epsilon)^{-(s-1)}$ and sum over s from 1 to $t-1$, we get

$$\zeta(\kappa, D)(1-\epsilon)^{-(t-2)}\Delta_t \leq (\zeta(\kappa, D)-1)\Delta_1 + \frac{L_g - \mu}{2} d^2(x_1, x^*). \tag{2.28}$$

Observe that since $\Delta_1 \leq \frac{L_g D^2}{2}$, it follows that $\Delta_t \leq \frac{(1-\epsilon)^{t-2} L_g D^2}{2}$, as desired. □

It must be emphasized that the proofs of Theorems 2.5, 2.6, and 2.7 contain some additional difficulties beyond their Euclidean counterparts. In particular, the term Δ_s does not cancel nicely due to the presence of the curvature term $\zeta(\kappa, D)$, which necessitates use of a different Lyapunov function to ensure convergence. Consequently, the stochastic gradient algorithm in Theorem 2.6 requires some unusual looking averaging scheme. In Theorem 2.7, since the distance between iterates and the minimizer is shrinking, better dependency on κ may also be possible if one replaces $\zeta(\kappa, D)$ by a tighter constant.

Proximal gradient optimization. In this subsection, we assume the objective function f is geodesically strongly convex but nonsmooth, yet is the sum of a geodesically smooth and strongly convex function f_{ss} and a geodesically convex and nonsmooth function f_c. We show that if in addition to the gradient oracle of f_{ss}, we have access to a proximal oracle of f_c that solves

$$x_{s+1} = \arg\min_x f_c(x) + \frac{\lambda}{2} d^2(x, x_s^+) \tag{2.29}$$

then the minimization of f enjoys linear convergence.

2.4 Experiments

To empirically validate our results, we compare the performance of a stochastic gradient algorithm with a full gradient descent algorithm on the matrix Karcher mean problem. Averaging PSD matrices have applications in averaging data of anisotropic symmetric positive-definite tensors, such as in diffusion tensor imaging [Fletcher and Joshi, 2007, Pennec et al., 2006] and elasticity theory [Cowin

and Yang, 1997]. The computation and properties of various notions of geometric means have been studied by many (e.g. Bini and Iannazzo [2013], Moakher [2005], Sra and Hosseini [2015]). Specifically, the Karcher mean of a set of N symmetric positive definite matrices $\{A_i\}_{i=1}^N$ is defined as the PSD matrix that minimizes the sum of squared distance induced by the Riemannian metric:

$$d(X, Y) = \| \log(X^{-1/2} Y X^{-1/2}) \|_F$$

The loss function

$$f(X; \{A_i\}_{i=1}^N) = \sum_{i=1}^N \| \log(X^{-1/2} A_i X^{-1/2}) \|_F^2$$

is known to be nonconvex in Euclidean space but geometrically $2N$-strongly convex , enabling the use of geometrically convex optimization algorithms. The full gradient update step is

$$X_{s+1} = X_s^{1/2} \exp\left(-\eta_s \sum_{i=1}^N \log(X_s^{1/2} A_i^{-1} X_s^{1/2}) \right) X_s^{1/2}$$

For stochastic gradient update, we set

$$X_{s+1} = X_s^{1/2} \exp\left(-\eta_s N \log(X_s^{1/2} A_{i(s)}^{-1} X_s^{1/2}) \right) X_s^{1/2}$$

where each index $i(s)$ is drawn uniformly at random from $\{1, \ldots, N\}$. The step-sizes η_s for gradient descent and stochastic gradient method have to be chosen according to the smoothness constant or the strongly-convex constant of the loss function. Unfortunately, unlike the Euclidean square loss, there is no cheap way to compute the smoothness constant exactly. In [Bini and Iannazzo, 2013] the authors proposed an adaptive procedure to estimate the optimal step-size. Empirically, however, we observe that an L_g estimate of $5N$ always guarantees convergence. We compare the performance of three algorithms that can be applied to this problem:

- Gradient descent (GD) with $\eta_s = \frac{1}{5N}$ set according to the estimate of the smoothness constant (Theorem 2.7).

- Stochastic gradient method for smooth functions (SGD-sm) $\eta_s = \frac{1}{L_g + 1/\alpha}$ with

$\alpha = \frac{D}{\sigma}\sqrt{\frac{1}{\zeta(\kappa,D)t}}$, where the parameters are set according to the estimates of the smoothness constant, domain diameter and gradient variance (Theorem 2.6).

- Stochastic subgradient method for strongly convex functions (SGD-st) with $\eta_s = \frac{1}{N(s+1)}$ set according to the $2N$-strong convexity of the loss function (Theorem 2.4).

Our data are 100×100 random PSD matrices generated using the Matrix Mean Toolbox [Bini and Iannazzo, 2013]. All matrices are explicitly normalized so that their norms all equal 1. We compare the algorithms on four datasets with $N \in \{10^2, 10^3\}$ matrices to average and the condition number Q of each matrix being either 10^2 or 10^8. For all experiments we initialize X using the arithmetic mean of the dataset. Figure 2-2 shows $f(X) - f(X^*)$ as a function of number of passes through the dataset. We observe that the full gradient algorithm with fixed step-size achieves linear convergence, whereas the stochastic gradient algorithms have a sublinear convergence rate, but is much faster during the initial steps.

2.5 Discussion

In this chapter, we make contributions to the understanding of geodesically convex optimization on Hadamard manifolds. Our contributions are twofold: first, we develop a user-friendly trigonometric distance bound for Alexandrov space with curvature bounded below, which includes several commonly known Riemannian manifolds as special cases; second, we prove iteration complexity upper bounds for several first-order algorithms on Hadamard manifolds, which are the first such analyses up to the best of our knowledge. We believe that our analysis is a small step, yet in the right direction, towards understanding and realizing the power of optimization in nonlinear spaces.

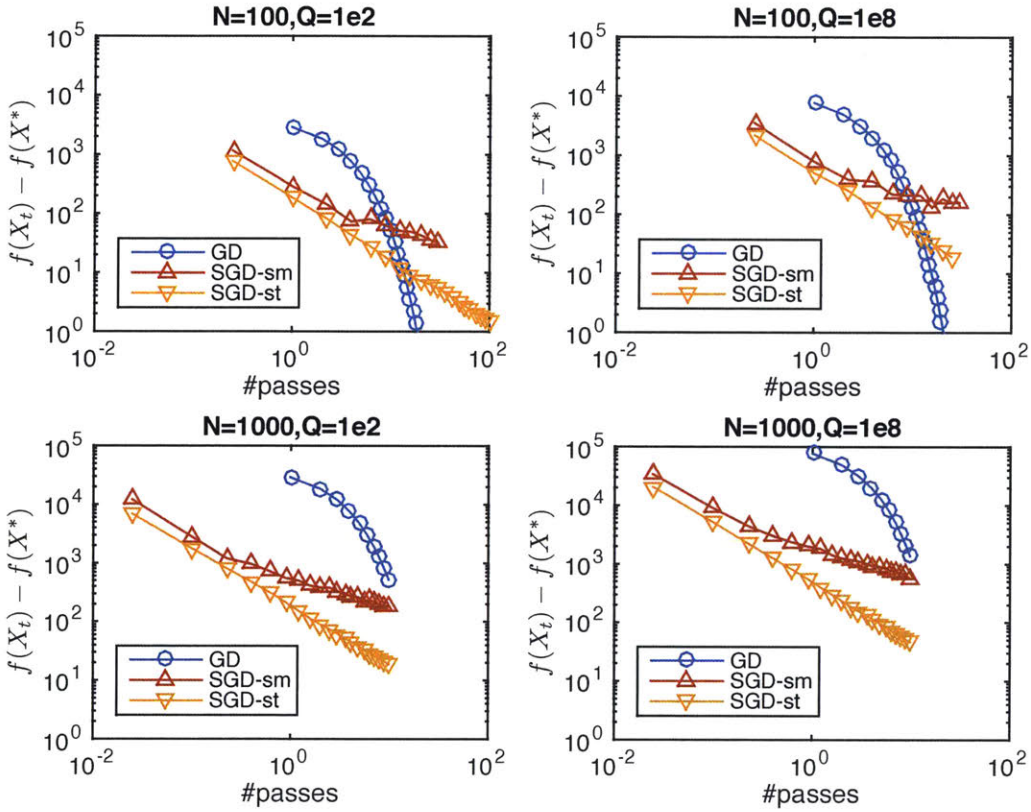


Figure 2-2: Comparing gradient descent and stochastic gradient methods in matrix Karcher mean problems. Shown are loglog plots of three algorithms on different datasets. GD: gradient descent (Theorem 2.7); SGD-sm: stochastic gradient method for smooth functions (Theorem 2.6); SGD-st: stochastic (sub)gradient method for strongly convex functions (Theorem 2.4). We varied two parameters: size of the dataset $n \in \{10^2, 10^3\}$ and conditional number $Q \in \{10^2, 10^8\}$. Data generating process, initialization and step-size are described in the main text. It is validated from the figures that GD converges at a linear rate, SGD-sm converges asymptotically at the $O(1/\sqrt{t})$ rate, and SGD-st converges at the $O(1/t)$ rate.

2.5.1 Future Directions

Many questions are not yet answered. We summarize some important ones in the following:

- A long-time question is whether the famous Nesterov's accelerated gradient descent algorithms have nonlinear space counterparts. The analysis of Nesterov's algorithms typically relies on a proximal gradient projection interpretation. In nonlinear space, we have not been able to find an analogy to such a projection. Further study is needed to see if similar analysis can be developed, or a different approach is required, or Nesterov's algorithms have no nonlinear space counterparts.

- Another interesting direction is variance reduced stochastic gradient methods for geodesically convex functions. For smooth and convex optimization in Euclidean space, these methods have recently drawn great interests and enjoyed remarkable empirical success. We hypothesize that similar algorithms can achieve faster convergence over naive incremental gradient methods on Hadamard manifolds.

- Finally, since in applications it is often favorable to replace exponential mapping with computationally cheap retractions, it is important to understand the effect of this approximation on convergence rate. Analyzing this effect is of both theoretical and practical interests.

2.6 Proofs

2.6.1 Proof of Lemma 2.1

Lemma 2.3. *Let*

$$g(b,c) = \cosh\sqrt{\frac{c}{\tanh(c)}b^2 + c^2 - 2bc\cos(A)}$$

then

$$\frac{\partial^2}{\partial b^2} g(b,c) \geq g(b,c), \quad b,c \geq 0$$

Proof. If $c = 0$, $g(b,c) = \cosh(b) = \frac{\partial^2}{\partial b^2} g(b,c)$. Now we focus on the case when $c > 0$. If $c > 0$, Let $u = \sqrt{(1+x)b^2 + c^2 - 2bc\cos(A)}$ where $x = x(c)$. We have

$$u^2 = (1+x)b^2 - 2bc\cos(A) + c^2 \geq \frac{c^2(x + \sin^2 A)}{1+x} = u_{\min}^2 > 0$$

$$\frac{\partial^2}{\partial b^2} g(b,c) = \left(1 + x - c^2\left(x + \sin^2 A\right)\frac{1}{u^2}\right)\cosh(u) + c^2\left(x + \sin^2 A\right)\frac{\sinh(u)}{u^3}$$

Since $g(b,c) = \cosh(u) > 0$, it suffices to prove

$$\frac{\frac{\partial^2}{\partial b^2} g(b,c)}{g(b,c)} - 1 = x\left(1 - \frac{c^2}{x}\left(x + \sin^2 A\right)\frac{1}{u^2} + \frac{c^2}{x}\left(x + \sin^2 A\right)\frac{\tanh(u)}{u^3}\right) \geq 0$$

so it suffices to prove

$$h_1(u) = \frac{u^3}{u - \tanh(u)} \geq \frac{c^2}{x}(x + \sin^2 A)$$

Solving for $h_1'(u) = 0$, we get $u = 0$. Since $\lim_{u \to 0_+} h_1(u) = 0$ and $h_1(u) > 0, \forall u > 0$, $h_1(u)$ is monotonically increasing on $u > 0$. Thus $h_1(u) \geq h_1(u_{\min}), \forall u > 0$. Note that $\frac{c^2}{x}(x + \sin^2 A) = \frac{1+x}{x} u_{\min}^2$, thus it suffices to prove

$$h_1(u_{\min}) = \frac{u_{\min}^3}{u_{\min} - \tanh(u_{\min})} \geq \frac{(1+x)u_{\min}^2}{x}$$

or equivalently

$$\frac{\tanh(u_{\min})}{u_{\min}} \geq \frac{1}{1+x}$$

Now fix c and notice that $\frac{\tanh(u_{\min})}{u_{\min}}$ as a function of $\sin^2 A$ is monotonically decreasing. Therefore its minimum is obtained at $\sin^2 A = 1$, where $u_{\min}^2 = u_*^2 = c^2$, i.e. $u_* = c$. So it only remains to show

$$\frac{\tanh(u_*)}{u_*} = \frac{\tanh(c)}{c} \geq \frac{1}{1+x}, \forall c > 0$$

or equivalently

$$1 + x \geq \frac{c}{\tanh(c)}, \forall c > 0$$

which is true by our definition of g. □

Lemma 2.4. *Suppose $h(x)$ is twice differentiable on $[r, +\infty)$ with three further assump-*

tions:

1. $h(r) \leq 0$,
2. $h'(r) \leq 0$,
3. $h''(x) \leq h(x), \forall x \in [r, +\infty)$,

then $h(x) \leq 0, \forall x \in [r, +\infty)$

Proof. It suffices to prove $h'(x) \leq 0, \forall x \in [r, +\infty)$. We prove this claim by contradiction.

Suppose the claim doesn't hold, then there exist some $t > s \geq r$ so that $h'(x) \leq 0$ for any x in $[r, s]$, $h'(s) = 0$ and $h'(x) > 0$ is monotonically increasing in $(s, t]$. It follows that for any $x \in [s, t]$ we have

$$h''(x) \leq h(x) \leq \int_r^x h'(u)du \leq \int_s^x h'(u)du \leq (x - s)h'(x) \leq (t - s)h'(x)$$

Thus by Grönwall's inequality,

$$h'(t) \leq h'(s)e^{(t-s)^2} = 0$$

which leads to a contradiction with our assumption $h'(t) > 0$. □

Lemma 2.5. *If a, b, c are the sides of a (geodesic) triangle in a hyperbolic space of constant curvature -1, and A is the angle between b and c, then*

$$a^2 \leq \frac{c}{\tanh(c)} b^2 + c^2 - 2bc\cos(A)$$

Proof. For a fixed but arbitrary $c \geq 0$, define $h_c(x) = f(x, c) - g(x, c)$. By Lemma 2.3 it is easy to verify that $h_c(x)$ satisfies the assumptions of Lemma 2.4. Apply Lemma 2.4 to h_c with $r = 0$ to show $h_c \leq 0$ in $[0, +\infty)$. Therefore $f(b, c) \leq g(b, c)$ for any $b, c \geq 0$. Finally use the fact that $\cosh(x)$ is monotonically increasing on $[0, +\infty)$. □

Corollary 2.2. *If a, b, c are the sides of a (geodesic) triangle in a hyperbolic space of constant curvature κ, and A is the angle between b and c, then*

$$a^2 \leq \frac{\sqrt{|\kappa|}c}{\tanh(\sqrt{|\kappa|}c)} b^2 + c^2 - 2bc\cos(A)$$

Proof. For hyperbolic space of constant curvature $\kappa < 0$, the law of cosines is

$$\cosh(\sqrt{|\kappa|}a) = \cosh(\sqrt{|\kappa|}b)\cosh(\sqrt{|\kappa|}c) - \sinh(\sqrt{|\kappa|}b)\sinh(\sqrt{|\kappa|}c)\cos A$$

which corresponds to the law of cosines of a geodesic triangle in hyperbolic space of curvature -1 with side lengths $\sqrt{|\kappa|}a, \sqrt{|\kappa|}b, \sqrt{|\kappa|}c$. Applying Lemma 2.5 we thus get

$$|\kappa|a^2 \leq \frac{\sqrt{|\kappa|}c}{\tanh(\sqrt{|\kappa|}c)}|\kappa|b^2 + |\kappa|c^2 - 2|\kappa|bc\cos(A)$$

and the corollary follows directly. □

2.6.2 Proof of Theorem 2.4

Theorem 2.4. If f is geodesically μ-strongly convex, the sectional curvature of the manifold is lower bounded by $\kappa \leq 0$, and the stochastic subgradient oracle satisfies $\mathbb{E}[\widetilde{g}(x)] = g(x) \in \partial f(x), \mathbb{E}[\|\widetilde{g}_s\|^2] \leq G^2$, then the projected subgradient method with $\eta_s = \frac{2}{\mu(s+1)}$ satisfies

$$\mathbb{E}[f(\overline{x}_t) - f(x^*)] \leq \frac{2\zeta(\kappa, D)G^2}{\mu(t+1)}$$

where $\overline{x}_1 = x_1$, and $\overline{x}_{s+1} = \mathrm{Exp}_{\overline{x}_s}\left(\frac{2}{s+1}\mathrm{Exp}_{\overline{x}_s}^{-1}(x_{s+1})\right)$.

Proof. Since f is geodesically μ-strongly convex, we have

$$\mathbb{E}[f(x_s) - f(x^*)] \leq \langle -\mathbb{E}[\widetilde{g}_s], \mathrm{Exp}_{x_s}^{-1}(x^*)\rangle - \frac{\mu}{2}\mathbb{E}[d^2(x_s, x^*)]$$

which combined with Corollary 2.1 and $\mathbb{E}[\|\widetilde{g}_s\|^2] \leq G^2$ yields

$$\begin{aligned}\mathbb{E}[f(x_s) - f(x^*)] &\leq \left(\frac{1}{2\eta_s} - \frac{\mu}{2}\right)\mathbb{E}\left[d^2(x_s, x^*)\right] - \frac{1}{2\eta_s}\mathbb{E}\left[d^2(x_{s+1}, x^*)\right] + \frac{\zeta(\kappa, D)G^2\eta_s}{2} \\ &= \frac{\mu(s-1)}{4}\mathbb{E}\left[d^2(x_s, x^*)\right] - \frac{\mu(s+1)}{4}\mathbb{E}\left[d^2(x_{s+1}, x^*)\right] + \frac{\zeta(\kappa, D)G^2}{\mu(s+1)}\end{aligned} \tag{2.30}$$

Multiply (2.30) by s and sum over s from 1 to t, then divide the result by $\frac{t(t+1)}{2}$ we get

$$\mathbb{E}\left[\frac{2}{t(t+1)}\sum_{s=1}^{t} sf(x_s) - f(x^*)\right] \leq \frac{2\zeta(\kappa, D)G^2}{\mu(t+1)} \tag{2.31}$$

The final step is to show $f(\overline{x}_t) \leq \frac{2}{t(t+1)}\sum_{s=1}^{t} sf(x_s)$ by induction. □

3

Iteration Complexity of Riemannian SVRG Methods

We study optimization of finite sums of *geodesically* smooth functions on Riemannian manifolds. Although variance reduction techniques for optimizing finite-sums have witnessed tremendous attention in the recent years, existing work is limited to vector space problems. We introduce *Riemannian SVRG* (RSVRG), a new variance reduced Riemannian optimization method. We analyze RSVRGfor both geodesically *convex* and *nonconvex* (smooth) functions. Our analysis reveals that RSVRGinherits advantages of the usual SVRG method, but with factors depending on curvature of the manifold that influence its convergence. To our knowledge, RSVRGis the first *provably fast* stochastic Riemannian method. Moreover, our paper presents the first non-asymptotic complexity analysis (novel even for the batch set-

ting) for nonconvex Riemannian optimization. Our results have several implications; for instance, they offer a Riemannian perspective on variance reduced PCA, which promises a short, transparent convergence analysis.

3.1 Introduction

We study the following rich class of (possibly nonconvex) finite-sum optimization problems:

$$\min_{x \in \mathcal{X} \subset \mathcal{M}} f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{3.1}$$

where $(\mathcal{M}, \mathfrak{g})$ is a Riemannian manifold with the Riemannian metric $\mathfrak{g}$, and $\mathcal{X} \subset \mathcal{M}$ is a geodesically convex set. We assume that each $f_i : \mathcal{M} \to \mathbb{R}$ is geodesically L-smooth (see §3.2). Problem equation 3.1 generalizes the fundamental machine learning problem of empirical risk minimization, which is usually cast in vector spaces, to a Riemannian setting. It also includes as special cases important problems such as principal component analysis (PCA), independent component analysis (ICA), dictionary learning, mixture modeling, among others (see e.g., the related work section).

The Euclidean version of equation 3.1 where $\mathcal{M} = \mathbb{R}^d$ and $\mathfrak{g}$ is the Euclidean inner-product has been the subject of intense algorithmic development in machine learning and optimization, starting with the classical work of Robbins and Monro [1951] to the recent spate of work on variance reduction [Defazio et al., 2014, Johnson and Zhang, 2013, Konečnỳ and Richtárik, 2013, Reddi et al., 2016, Schmidt et al., 2013]. However, when $(\mathcal{M}, \mathfrak{g})$ is a nonlinear Riemannian manifold, much less is known beyond [Bonnabel, 2013, Zhang and Sra, 2016].

When solving problems with manifold constraints, one common approach is to alternate between optimizing in the ambient Euclidean space and "projecting" onto the manifold. For example, two well-known methods to compute the leading eigenvector of symmetric matrices, power iteration and Oja's algorithm [Oja, 1992], are in essence projected gradient and projected stochastic gradient algorithms. For certain manifolds (e.g., positive definite matrices), projections can be

quite expensive to compute.

An effective alternative is to use *Riemannian optimization*[1], which directly operates on the manifold in question. This mode of operation allows Riemannian optimization to view the constrained optimization problem equation 3.1 as an unconstrained problem on a manifold, and thus, to be "projection-free." More important is its conceptual value: viewing a problem through the Riemannian lens, one can discover insights into problem geometry, which can translate into better optimization algorithms.

Although the Riemannian approach is appealing, our knowledge of it is fairly limited. In particular, there is little analysis about its global complexity (a.k.a. non-asymptotic convergence rate), in part due to the difficulty posed by the nonlinear metric. Only very recently Zhang and Sra [2016] developed the first global complexity analysis of batch and stochastic gradient methods for geodesically convex functions. However, the batch and stochastic gradient methods in Zhang and Sra [2016] suffer from problems similar to their vector space counterparts. For solving finite sum problems with n components, the full-gradient method requires n derivatives at each step; the stochastic method requires only one derivative but at the expense of slower $O(1/\epsilon^2)$ convergence to an ϵ-accurate solution.

These issues have motivated much of the recent progress on faster stochastic optimization in vector spaces by using variance reduction Defazio et al. [2014], Johnson and Zhang [2013], Schmidt et al. [2013] techniques. However, all ensuing methods critically rely on properties of vector spaces, whereby, adapting them to work in the context of Riemannian manifolds poses major challenges. Given the richness of Riemannian optimization (it includes vector space optimization as a special case) and its growing number of applications, developing fast stochastic Riemannian optimization is important. It will help us apply Riemannian optimization to large-scale problems, while offering a new set of algorithmic tools for the practitioner's repertoire.

[1]Riemannian optimization is optimization on a *known* manifold structure. Note the distinction from *manifold learning*, which attempts to learn a manifold structure from data. We briefly review some Riemannian optimization applications in the related work.

Contributions. We summarize the key contributions of this chapter below.

- We introduce Riemannian SVRG (RSVRG), a variance reduced Riemannian stochastic gradient method based on SVRG Johnson and Zhang [2013]. We analyze RSVRGfor geodesically strongly convex functions through a novel theoretical analysis that accounts for the nonlinear (curved) geometry of the manifold to yield linear convergence rates.
- Building on recent advances in variance reduction for nonconvex optimization [Allen-Zhu and Hazan, 2016, Reddi et al., 2016], we generalize the convergence analysis of RSVRGto (geodesically) nonconvex functions and also to gradient dominated functions (see Section 3.2 for the definition). Our analysis provides the first stochastic Riemannian method that is provably superior to both batch and stochastic (Riemannian) gradient methods for nonconvex finite-sum problems.
- Using a Riemannian formulation and applying our result for (geodesically) gradient-dominated functions, we provide new insights, and a short transparent analysis explaining fast convergence of variance reduced PCA for computing the leading eigenvector of a symmetric matrix.

This chapter describes the first stochastic gradient method with global linear convergence rates for geodesically strongly convex functions, as well as the first non-asymptotic convergence rates for geodesically nonconvex optimization (even in the batch case). Our analysis reveals how manifold geometry, in particular curvature, impacts convergence rates. We illustrate the benefits of RSVRGby showing an application to computing leading eigenvectors of a symmetric matrix and to the task of computing the Riemannian centroid of covariance matrices, a problem that has received great attention in the literature [Bhatia, 2007, Jeuris et al., 2012, Zhang and Sra, 2016].

Related Work. Variance reduction techniques, such as *control variates*, are widely used in Monte Carlo simulations Rubinstein and Kroese [2011]. In linear spaces,

variance reduced methods for solving finite-sum problems have recently witnessed a huge surge of interest [e.g. Bach and Moulines, 2013, Defazio et al., 2014, Gong and Ye, 2014, Johnson and Zhang, 2013, Konečnỳ and Richtárik, 2013, Schmidt et al., 2013, Xiao and Zhang, 2014]. They have been shown to accelerate stochastic optimization for strongly convex objectives, convex objectives, nonconvex f_i ($i \in [n]$), and even when both f and f_i ($i \in [n]$) are nonconvex [Allen-Zhu and Hazan, 2016, Reddi et al., 2016]. Reddi et al. [2016] further proved global linear convergence for gradient dominated nonconvex problems. Our analysis is inspired by Johnson and Zhang [2013], Reddi et al. [2016], but applies to the substantially more general Riemannian optimization setting.

References of Riemannian optimization can be found in Absil et al. [2009b], Udriste [1994], where analysis is limited to asymptotic convergence (except [Udriste, 1994, Theorem 4.2] which proved linear rate convergence for first-order line search method with bounded and positive definite hessian). Stochastic Riemannian optimization has been previously considered in Bonnabel [2013], Liu et al. [2004], though with only asymptotic convergence analysis, and without any rates. Many applications of Riemannian optimization are known, including matrix factorization on fixed-rank manifold Tan et al. [2014], Vandereycken [2013], dictionary learning Cherian and Sra [2015], Sun et al. [2015], optimization under orthogonality constraints Edelman et al. [1998], Moakher [2002], covariance estimation Wiesel [2012], learning elliptical distributions Sra and Hosseini [2013], Zhang et al. [2013], and Gaussian mixture models Hosseini and Sra [2015a]. Notably, some nonconvex Euclidean problems are geodesically convex, for which Riemannian optimization can provide similar guarantees to convex optimization. Zhang and Sra [2016] provide the first global complexity analysis for first-order Riemannian algorithms, but their analysis is restricted to geodesically convex problems with full or stochastic gradients. In contrast, we propose RSVRG, a variance reduced Riemannian stochastic gradient algorithm, and analyze its global complexity for both geodesically convex and nonconvex problems.

In parallel with our work, Kasai et al. [2016] also proposed and analyzed RSVRG

specifically for the Grassmann manifold. Their complexity analysis is restricted to *local* convergence to strict local minima, which essentially corresponds to our analysis of (locally) geodesically strongly convex functions.

3.2 Preliminaries

Before formally discussing Riemannian optimization, let us recall some foundational concepts of Riemannian geometry. For a thorough review one can refer to any classic text, e.g.,[Petersen, 2006].

A *Riemannian manifold* $(\mathcal{M}, \mathfrak{g})$ is a real smooth manifold $\mathcal{M}$ equipped with a Riemannain metric $\mathfrak{g}$. The metric $\mathfrak{g}$ induces an inner product structure in each tangent space $T_x\mathcal{M}$ associated with every $x \in \mathcal{M}$. We denote the inner product of $u, v \in T_x\mathcal{M}$ as $\langle u, v\rangle \triangleq \mathfrak{g}_x(u, v)$; and the norm of $u \in T_x\mathcal{M}$ is defined as $\|u\| \triangleq \sqrt{\mathfrak{g}_x(u, u)}$. The angle between u, v is defined as $\arccos \frac{\langle u,v\rangle}{\|u\|\|v\|}$. A geodesic is a constant speed curve $\gamma : [0, 1] \to \mathcal{M}$ that is locally distance minimizing. An exponential map $\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$ maps v in $T_x\mathcal{M}$ to y on $\mathcal{M}$, such that there is a geodesic γ with $\gamma(0) = x, \gamma(1) = y$ and $\dot{\gamma}(0) \triangleq \frac{d}{dt}\gamma(0) = v$. If between any two points in $\mathcal{X} \subset \mathcal{M}$ there is a unique geodesic, the exponential map has an inverse $\mathrm{Exp}_x^{-1} : \mathcal{X} \to T_x\mathcal{M}$ and the geodesic is the unique shortest path with $\|\mathrm{Exp}_x^{-1}(y)\| = \|\mathrm{Exp}_y^{-1}(x)\|$ the geodesic distance between $x, y \in \mathcal{X}$.

Parallel transport $\Gamma_x^y : T_x\mathcal{M} \to T_y\mathcal{M}$ maps a vector $v \in T_x\mathcal{M}$ to $\Gamma_x^y v \in T_y\mathcal{M}$, while preserving norm, and roughly speaking, "direction," analogous to translation in $\mathbb{R}^d$. A tangent vector of a geodesic γ remains tangent if parallel transported along γ. Parallel transport preserves inner products.

Figure 3-1: Illustration of manifold operations. (Left) A vector v in $T_x\mathcal{M}$ is mapped to $\mathrm{Exp}_x(v)$; (right) A vector v in $T_x\mathcal{M}$ is parallel transported to $T_y\mathcal{M}$ as $\Gamma_x^y v$.

The geometry of a Riemannian manifold is determined by its Riemannian metric tensor through various characterization of curvatures. Let $u, v \in T_x\mathcal{M}$ be linearly independent, so that they span a two dimensional subspace of $T_x\mathcal{M}$. Under the exponential map, this subspace is mapped to a two dimensional submanifold of $\mathcal{U} \subset \mathcal{M}$. The sectional curvature $\kappa(x, \mathcal{U})$ is defined as the Gauss curvature of $\mathcal{U}$ at x. As we will mainly analyze manifold trigonometry, for worst-case analysis, it is sufficient to consider sectional curvature.

Function Classes. We now define some key terms. A set $\mathcal{X}$ is called *geodesically convex* if for any $x, y \in \mathcal{X}$, there is a geodesic γ with $\gamma(0) = x, \gamma(1) = y$ and $\gamma(t) \in \mathcal{X}$ for $t \in [0, 1]$. Throughout the paper, we assume that the function f in (3.1) is defined on a geodesically convex set $\mathcal{X}$ on a Riemannian manifold $\mathcal{M}$.

We call a function $f : \mathcal{X} \to \mathbb{R}$ *geodesically convex* (g-convex) if for any $x, y \in \mathcal{X}$ and any geodesic γ such that $\gamma(0) = x$, $\gamma(1) = y$ and $\gamma(t) \in \mathcal{X}$ for $t \in [0, 1]$, it holds that

$$f(\gamma(t)) \leq (1 - t)f(x) + tf(y).$$

It can be shown that if the inverse exponential map is well-defined, an equivalent definition is that for any $x, y \in \mathcal{X}$, $f(y) \geq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y) \rangle$, where g_x is a subgradient of f at x (or the gradient if f is differentiable). A function $f : \mathcal{X} \to \mathbb{R}$ is called *geodesically μ-strongly convex* (μ-strongly g-convex) if for any $x, y \in \mathcal{X}$ and subgradient g_x, it holds that

$$f(y) \geq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y) \rangle + \tfrac{\mu}{2} \|\mathrm{Exp}_x^{-1}(y)\|^2.$$

We call a vector field $g : \mathcal{X} \to \mathbb{R}^d$ *geodesically L-Lipschitz* (L-g-Lipschitz) if for any $x, y \in \mathcal{X}$,

$$\|g(x) - \Gamma_y^x g(y)\| \leq L \|\mathrm{Exp}_x^{-1}(y)\|,$$

where Γ_y^x is the parallel transport from y to x. We call a differentiable function $f : \mathcal{X} \to \mathbb{R}$ *geodesically L-smooth* (L-g-smooth) if its gradient is L-g-Lipschitz, in which case we have

$$f(y) \leq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y) \rangle + \tfrac{L}{2} \|\mathrm{Exp}_x^{-1}(y)\|^2.$$

We say $f : \mathcal{X} \to \mathbb{R}$ is τ-*gradient dominated* if x^* is a global minimizer of f and for every $x \in \mathcal{X}$

$$f(x) - f(x^*) \leq \tau \|\nabla f(x)\|^2. \tag{3.2}$$

We recall the following trigonometric distance bound that is essential for our analysis:

Lemma 3.1 (Bonnabel [2013], Zhang and Sra [2016]). *If a, b, c are the side lengths of a geodesic triangle in a Riemannian manifold with sectional curvature lower bounded by $\kappa_{\min}$, and A is the angle between sides b and c (defined through inverse exponential map and inner product in tangent space), then*

$$a^2 \leq \frac{\sqrt{|\kappa_{\min}|}c}{\tanh(\sqrt{|\kappa_{\min}|}c)} b^2 + c^2 - 2bc\cos(A). \tag{3.3}$$

An *Incremental First-order Oracle (IFO)* Agarwal and Bottou [2015] in (3.1) takes an $i \in [n]$ and a point $x \in \mathcal{X}$, and returns a pair $(f_i(x), \nabla f_i(x)) \in \mathbb{R} \times T_x\mathcal{M}$. We measure non-asymptotic complexity in terms of IFO calls.

3.3 Riemannian SVRG

In this section we introduce RSVRGformally. We make the following standing assumptions: (a) f attains its optimum at $x^* \in \mathcal{X}$; (b) $\mathcal{X}$ is compact, and the diameter of $\mathcal{X}$ is bounded by D, that is, $\max_{x,y\in\mathcal{X}} d(x, y) \leq D$; (c) the sectional curvature in $\mathcal{X}$ is *upper* bounded by $\kappa_{\max}$, and within $\mathcal{X}$ the exponential map is invertible; and (d) the sectional curvature in $\mathcal{X}$ is *lower* bounded by $\kappa_{\min}$. We define the following key geometric constant that capture the impact of manifold curvature:

$$\boxed{\zeta = \begin{cases} \frac{\sqrt{|\kappa_{\min}|}D}{\tanh(\sqrt{|\kappa_{\min}|}D)}, & \text{if } \kappa_{\min} < 0, \\ 1, & \text{if } \kappa_{\min} \geq 0, \end{cases}} \tag{3.4}$$

We note that most (if not all) practical manifold optimization problems can satisfy these assumptions.

Our proposed RSVRGalgorithm is shown in Algorithm 1. Compared with the Euclidean SVRG, it differs in two key aspects: the variance reduction step uses

parallel transport to combine gradients from different tangent spaces; and the exponential map is used (instead of the update $x_t^{s+1} - \eta v_t^{s+1}$).

Algorithm 1: RSVRG(x^0, m, η, S)

Parameters: update frequency m, learning rate η, number of epochs S

initialize $\tilde{x}^0 = x^0$;
for $s = 0, 1, \ldots, S-1$ **do**
  $x_0^{s+1} = \tilde{x}^s$;
  $g^{s+1} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{x}^s)$;
  for $t = 0, 1, \ldots, m-1$ **do**
    Randomly pick $i_t \in \{1, \ldots, n\}$;
    $v_t^{s+1} = \nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}} \left(\nabla f_{i_t}(\tilde{x}^s) - g^{s+1}\right)$;
    $x_{t+1}^{s+1} = \mathrm{Exp}_{x_t^{s+1}}\left(-\eta v_t^{s+1}\right)$;
  end
  Set $\tilde{x}^{s+1} = x_m^{s+1}$;
end
Option I: output $x_a = \tilde{x}^S$;
Option II: output x_a chosen uniformly randomly from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$.

3.3.1 Convergence analysis for strongly g-convex functions

In this section, we analyze global complexity of RSVRGfor solving (3.1), where each f_i ($i \in [n]$) is g-smooth and f is strongly g-convex. In this case, we show that RSVRGhas linear convergence rate. This is in contrast with the $O(1/t)$ rate of Riemannian stochastic gradient algorithm for strongly g-convex functions Zhang and Sra [2016].

Theorem 3.1. *Assume in (3.1) each f_i is L-g-smooth, and f is μ-strongly g-convex, then if we run Algorithm 1 with Option I and parameters that satisfy*

$$\alpha = \frac{3\zeta\eta L^2}{\mu - 2\zeta\eta L^2} + \frac{(1 + 4\zeta\eta^2 - 2\eta\mu)^m(\mu - 5\zeta\eta L^2)}{\mu - 2\zeta\eta L^2} < 1$$

then with S outer loops, the Riemannian SVRG algorithm produces an iterate x_a that satisfies

$$\mathbb{E}d^2(x_a, x^*) \leq \alpha^S d^2(x^0, x^*).$$

The proof of Theorem 3.1 is in the appendix, and takes a different route compared with the original SVRG proof Johnson and Zhang [2013]. Specifically, due

to the nonlinear Riemannian metric, we are not able to bound the squared norm of the variance reduced gradient by $f(x) - f(x^*)$. Instead, we bound this quantity by the squared distances to the minimizer, and show linear convergence of the iterates. A bound on $\mathbb{E}[f(x) - f(x^*)]$ is then implied by L-g-smoothness, albeit with a stronger dependence on the condition number. Theorem 3.1 leads to the following more digestible corollary on the global complexity of the algorithm:

Corollary 3.1. *With assumptions as in Theorem 3.1 and properly chosen parameters, after* $O\left((n + \frac{\zeta L^2}{\mu^2}) \log(\frac{1}{\epsilon})\right)$ *IFO calls, the output* x_a *satisfies*

$$\mathbb{E}[f(x_a) - f(x^*)] \leq \epsilon.$$

We give a proof with specific parameter choices in the appendix. Observe the dependence on ζ in our result: for $\kappa_{\min} < 0$, we have $\zeta > 1$, which implies that negative space curvature adversarially affects convergence rate; while for $\kappa_{\min} \geq 0$, we have $\zeta = 1$, which implies that for nonnegatively curved manifolds, the impact of curvature is not explicit. In the rest of our analysis we will see a similar effect of sectional curvature; this phenomenon seems innate to manifold optimization (also see Zhang and Sra [2016]).

In the analysis we do not assume each f_i to be g-convex, which resulted in a worse dependence on the condition number. We note that a similar result was obtained in linear space Garber and Hazan [2015]. However, we will see in the next section that by generalizing the analysis for gradient dominated functions in Reddi et al. [2016], we are able to greatly improve this dependence.

3.3.2 Convergence analysis for geodesically nonconvex functions

In this section, we analyze global complexity of RSVRGfor solving (3.1), where each f_i is only required to be L-g-smooth, and neither f_i nor f need be g-convex. We measure convergence to a stationary point using $\|\nabla f(x)\|^2$ following Ghadimi and Lan [2013]. Note, however, that here $\nabla f(x) \in T_x\mathcal{M}$ and $\|\nabla f(x)\|$ is defined via the inner product in $T_x\mathcal{M}$. We first note that Riemannian-SGD on nonconvex L-

g-smooth problems attains $O(1/\epsilon^2)$ convergence as SGD Ghadimi and Lan [2013] holds; we relegate the details to the appendix.

Recently, two groups independently proved that variance reduction also benefits stochastic gradient methods for nonconvex smooth finite-sum optimization problems, with different analysis Allen-Zhu and Hazan [2016], Reddi et al. [2016]. Our analysis for nonconvex RSVRGis inspired by Reddi et al. [2016]. Our main result for this section is Theorem 3.2.

Theorem 3.2. *Assume in (3.1) each f_i is L-g-smooth, the sectional curvature in $\mathcal{X}$ is lower bounded by $\kappa_{\min}$, and we run Algorithm 1 with Option II. Then there exist universal constants $\mu_0 \in (0,1), \nu > 0$ such that if we set $\eta = \mu_0/(Ln^{\alpha_1}\zeta^{\alpha_2})$ ($0 < \alpha_1 \leq 1$ and $0 \leq \alpha_2 \leq 2$), $m = \lfloor n^{3\alpha_1/2}/(3\mu_0\zeta^{1-2\alpha_2})\rfloor$ and $T = mS$, we have*

$$\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{Ln^{\alpha_1}\zeta^{\alpha_2}[f(x^0)-f(x^*)]}{T\nu},$$

where x^* is an optimal solution to equation 3.1.

The key challenge in proving Theorem 3.2 in the Riemannian setting is to incorporate the impact of using a nonlinear metric. Similar to the g-convex case, the nonlienar metric impacts the convergence, notably through the constant ζ that depends on a lower-bound on sectional curvature.

Reddi et al. [2016] suggested setting $\alpha_1 = 2/3$, in which case we obtain the following corollary.

Corollary 3.2. *With assumptions and parameters in Theorem 3.2, choosing $\alpha_1 = 2/3$, the IFO complexity for achieving an ϵ-accurate solution is:*

$$\textit{IFO calls} = \begin{cases} O\left(n + (n^{2/3}\zeta^{1-\alpha_2}/\epsilon)\right), & \textit{if } \alpha_2 \leq 1/2, \\ O\left(n\zeta^{2\alpha_2-1} + (n^{2/3}\zeta^{\alpha_2}/\epsilon)\right), & \textit{if } \alpha_2 > 1/2. \end{cases}$$

Setting $\alpha_2 = 1/2$ in Corollary 3.2 immediately leads to Corollary 3.3:

Corollary 3.3. *With assumptions in Theorem 3.2 and $\alpha_1 = 2/3, \alpha_2 = 1/2$, the IFO complexity for achieving an ϵ-accurate solution is $O\left(n + (n^{2/3}\zeta^{1/2}/\epsilon)\right)$.*

Algorithm 2: GD-SVRG(x^0, m, η, S, K)

Parameters: update frequency m, learning rate η, number of epochs S, K, x^0

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

2 | $x^{k+1} = \text{RSVRG}(x^k, m, \eta, S)$ with Option II;

3 **end**

Output: x^K

The same reasoning allows us to also capture the class of gradient dominated functions (3.2), for which Reddi et al. [2016] proved that SVRG converges linearly to a global optimum. We have the following corresponding theorem for RSVRG:

Theorem 3.3. *Suppose that in addition to the assumptions in Theorem 3.2, f is τ-gradient dominated. Then there exist universal constants $\mu_0 \in (0,1), \nu > 0$ such that if we run Algorithm 2 with $\eta = \mu_0/(Ln^{2/3}\zeta^{1/2}), m = \lfloor n/(3\mu_0) \rfloor, S = \lceil (6 + \frac{18\mu_0}{n-3}) L\tau\zeta^{1/2}\mu_0/(\nu n^{1/3}) \rceil$, we have*

$$\mathbb{E}[\|\nabla f(x^K)\|^2] \leq 2^{-K}\|\nabla f(x^0)\|^2,$$
$$\mathbb{E}[f(x^K) - f(x^*)] \leq 2^{-K}[f(x^0) - f(x^*)].$$

We summarize the implication of Theorem 3.3 as follows (note the dependence on curvature):

Corollary 3.4. *With Algorithm 2 and the parameters in Theorem 3.3, the IFO complexity to compute an ϵ-accurate solution for a gradient dominated function f is $O((n + L\tau\zeta^{1/2}n^{2/3})\log(1/\epsilon))$.*

A typical example of gradient dominated function is a strongly g-convex function (see appendix). Specifically, we have the following corollary, which prove linear convergence rate of RSVRGwith the same assumptions as in Theorem 3.1, improving the dependence on the condition number.

Corollary 3.5. *With Algorithm 2 and the parameters in Theorem 3.3, the IFO complexity to compute an ϵ-accurate solution for a μ-strongly g-convex function f is $O((n + \mu^{-1}L\zeta^{1/2}n^{2/3})\log(1/\epsilon))$.*

3.4 Applications

3.4.1 Computing the leading eigenvector

In this section, we apply our analysis of RSVRGfor gradient dominated functions (Theorem 3.3) to fast eigenvector computation, a fundamental problem that is still being actively researched in the big-data setting Garber and Hazan [2015], Jin et al. [2015], Shamir [2015]. For the problem of computing the leading eigenvector, i.e.,

$$\min_{x^\top x=1} \quad -x^\top \left(\sum_{i=1}^n z_i z_i^\top\right) x \quad \triangleq \quad -x^\top A x = f(x), \tag{3.5}$$

existing analyses for state-of-the-art algorithms typically result in $O(1/\delta^2)$ dependence on the eigengap δ of A, as opposed to the conjectured $O(1/\delta)$ dependence Shamir [2015], as well as the $O(1/\delta)$ dependence of power iteration. Here we give new support for the $O(1/\delta)$ conjecture. Note that Problem (3.5) seen as one in $\mathbb{R}^d$ is nonconvex, with negative semidefinite Hessian everywhere, and has nonlinear constraints. However, we show that on the hypersphere $\mathbb{S}^{d-1}$ Problem (3.5) is unconstrained, and has *gradient dominated* objective. In particular we have the following result:

Theorem 3.4. *Suppose A has eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_d$ and $\delta = \lambda_1 - \lambda_2$, and x^0 is drawn uniformly randomly on the hypersphere. Then with probability $1 - p$, x^0 falls in a Riemannian ball of a global optimum of the objective function, within which the objective function is $O(\frac{d}{p^2\delta})$-gradient dominated.*

We provide the proof of Theorem 3.4 in appendix. Theorem 3.4 gives new insights for why the conjecture might be true – once it is shown that with a constant stepsize and with high probability (both independent of δ) the iterates remain in such a Riemannian ball, applying Corollary 3.4 one can immediately prove the $O(1/\delta)$ dependence conjecture. We leave this analysis as future work.

Next we show that variance reduced PCA (VR-PCA) Shamir [2015] is closely related to RSVRG. We implement Riemannian SVRG for PCA, and use the code for VR-PCA in Shamir [2015]. Analytic forms for exponential map and parallel

transport on hypersphere can be found in [Absil et al., 2009b, Example 5.4.1; Example 8.1.1]. We conduct well-controlled experiments comparing the performance of two algorithms. Specifically, to investigate the dependence of convergence on δ, for each $\delta = 10^{-3}/k$ where $k = 1, \ldots, 25$, we generate a $d \times n$ matrix $Z = (z_1, \ldots, z_n)$ where $d = 10^3, n = 10^4$ using the method $Z = UDV^\top$ where U, V are orthonormal matrices and D is a diagonal matrix, as described in Shamir [2015]. Note that A has the same eigenvalues as D^2. All the data matrices share the same U, V and only differ in δ (thus also in D). We also fix the same random initialization x^0 and random seed. We run both algorithms on each matrix for 50 epochs. For every five epochs, we estimate the number of epochs required to double its accuracy [2]. This number can serve as an indicator of the global complexity of the algorithm. We plot this number for different epochs against $1/\delta$, shown in Figure 3-2. Note that the performance of RSVRG and VR-PCA with the same stepsize is very similar, which implies a close connection of the two. Indeed, the update $\frac{x+v}{\|x+v\|}$ used in Shamir [2015] and others is a well-known approximation to the exponential map $\mathrm{Exp}_x(v)$ with small stepsize (a.k.a. retraction). Also note the complexity of both algorithms seems to have an asymptotically linear dependence on $1/\delta$.

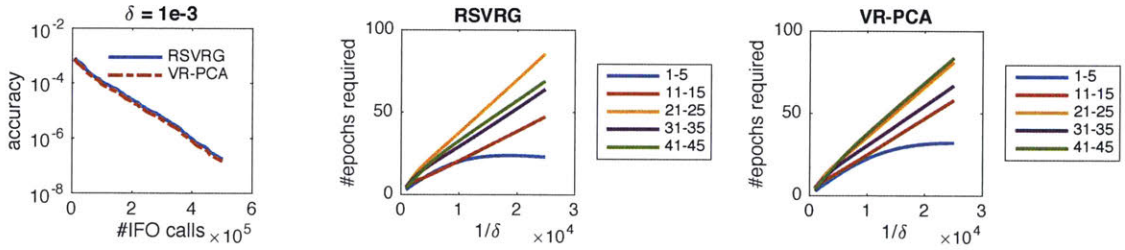


Figure 3-2: Computing the leading eigenvector. Left: RSVRG and VR-PCA are indistinguishable in terms of IFO complexity. Middle and right: Complexity appears to depend on $1/\delta$. x-axis shows the inverse of eigengap δ, y-axis shows the estimated number of epochs required to double the accuracy. Lines represent different epoch index. All variables are controlled except for δ.

[2] Accuracy is measured by $\frac{f(x)-f(x^*)}{|f(x^*)|}$, i.e. the relative error between the objective value and the optimum. We measure how much the error has been reduced after each five epochs, which is a multiplicative factor $c < 1$ on the error at the start of each five epochs. Then we use $\log(2)/\log(1/c) * 5$ as the estimate, assuming c stays constant.

3.4.2 Computing the Riemannian centroid

In this subsection we validate that RSVRG converges linearly for averaging PSD matrices under the Riemannian metric. The problem for finding the Riemannian centroid of a set of PSD matrices $\{A_i\}_{i=1}^n$ is

$$X^* = \arg\min_{X \succeq 0} \left\{ f(X; \{A_i\}_{i=1}^n) \triangleq \sum_{i=1}^n \| \log(X^{-1/2} A_i X^{-1/2}) \|_F^2 \right\}$$

where X is also a PSD matrix. This is a geodesically strongly convex problem, yet nonconvex in Euclidean space. It has been studied both in matrix computation and in various applications Bhatia [2007], Jeuris et al. [2012]. We use the same experiment setting as described in Zhang and Sra [2016] [3], and compare RSVRG against Riemannian full gradient (RGD) and stochastic gradient (RSGD) algorithms (Figure 3-3). Other methods for this problem include the relaxed Richardson iteration algorithm Bini and Iannazzo [2013], the approximated joint diagonalization algorithm Congedo et al. [2015], and Riemannian Newton and quasi-Newton type methods, notably the limited-memory Riemannian BFGS Yuan et al. [2016]. However, none of these methods were shown to greatly outperform RGD, especially in data science applications where n is large and extremely small optimization error is not required.

Note that the objective is sum of squared Riemannian distances in a nonpositively curved space, thus is $(2n)$-strongly g-convex and $(2n\zeta)$-g-smooth. According to the proof of Corollary 3.1 (see appendix) the optimal stepsize for RSVRG is $O(1/(\zeta^3 n))$. For all the experiments, we initialize all the algorithms using the arithmetic mean of the matrices. We set $\eta = \frac{1}{100n}$, and choose $m = n$ in Algorithm 1 for RSVRG, and use suggested parameters from Zhang and Sra [2016] for other algorithms. The results suggest RSVRGhas clear advantage in the large scale setting.

[3] We generate 100×100 random PSD matrices using the Matrix Mean Toolbox Bini and Iannazzo [2013] with normalization so that the norm of each matrix equals 1.

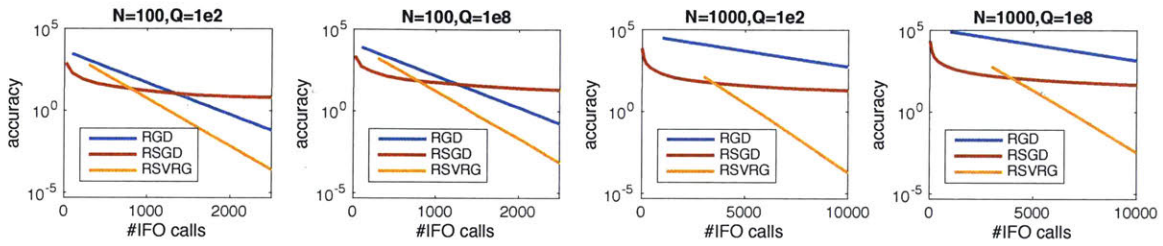


Figure 3-3: Riemannian mean of PSD matrices. N: number of matrices, Q: conditional number of each matrix. x-axis shows the actual number of IFO calls, y-axis show $f(X) - f(X^*)$ in log scale. Lines show the performance of different algorithms in colors. Note that RSVRG achieves linear convergence and is especially advantageous for large dataset.

3.5 Discussion

We introduce Riemannian SVRG, the first variance reduced stochastic gradient algorithm for Riemannian optimization. In addition, we analyze its global complexity for optimizing geodesically strongly convex, convex, and nonconvex functions, explicitly showing their dependence on sectional curvature. Our experiments validate our analysis that Riemannian SVRG is much faster than full gradient and stochastic gradient methods for solving finite-sum optimization problems on Riemannian manifolds.

Our analysis of computing the leading eigenvector as a Riemannian optimization problem is also worth noting: a nonconvex problem with nonpositive Hessian and nonlinear constraints in the ambient space turns out to be gradient dominated on the manifold. We believe this shows the promise of theoretical study of Riemannian optimization, and geometric optimization in general, and we hope it encourages other researchers in the community to join this endeavor.

Our work also has limitations — most practical Riemannian optimization algorithms use retraction and vector transport to efficiently approximate the exponential map and parallel transport, which we do not analyze in this work. A systematic study of retraction and vector transport is an important topic for future research. For other applications of Riemannian optimization such as low-rank matrix completion Vandereycken [2013], covariance matrix estimation Wiesel [2012] and subspace tracking Edelman et al. [1998], we believe it would also be promising to apply fast incremental gradient algorithms in the large scale setting.

3.6 Proofs

3.6.1 Proofs for Section 3.3.1

Theorem 3.1. Assume in (3.1) each f_i is L-g-smooth, and f is μ-strongly g-convex, then if we run Algorithm 1 with Option I and parameters that satisfy

$$\alpha = \frac{3\zeta\eta L^2}{\mu - 2\zeta\eta L^2} + \frac{(1 + 4\zeta\eta^2 - 2\eta\mu)^m(\mu - 5\zeta\eta L^2)}{\mu - 2\zeta\eta L^2} < 1$$

then with S outer loops, the Riemannian SVRG algorithm produces an iterate x_a that satisfies

$$\mathbb{E}d^2(x_a, x^*) \leq \alpha^S d^2(x^0, x^*).$$

Proof. We start by bounding the squared norm of the variance reduced gradient. Since $v_t^{s+1} = \nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}(\nabla f_{i_t}(\tilde{x}^s) - g^{s+1})$, conditioned on x_t^{s+1} and taking expectation with respect to i_t, we obtain:

$$\begin{aligned}
\mathbb{E}\|v_t^{s+1}\|^2 &= \mathbb{E}\left\|\nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\left(\nabla f_{i_t}(\tilde{x}^s) - g^{s+1}\right)\right\|^2 \\
&= \mathbb{E}\left\|\left(\nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\nabla f_{i_t}(\tilde{x}^s)\right) + \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\left(\nabla f(\tilde{x}^s) - \Gamma_{x^*}^{\tilde{x}^s}\nabla f(x^*)\right)\right\|^2 \\
&\leq 2\mathbb{E}\left\|\nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\nabla f_{i_t}(\tilde{x}^s)\right\|^2 + 2\mathbb{E}\left\|\Gamma_{\tilde{x}^s}^{x_t^{s+1}}\left(\nabla f(\tilde{x}^s) - \Gamma_{x^*}^{\tilde{x}^s}\nabla f(x^*)\right)\right\|^2 \\
&= 2\mathbb{E}\left\|\nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\nabla f_{i_t}(\tilde{x}^s)\right\|^2 + 2\mathbb{E}\left\|\nabla f(\tilde{x}^s) - \Gamma_{x^*}^{\tilde{x}^s}\nabla f(x^*)\right\|^2 \\
&\leq 2L^2\left\|\mathrm{Exp}_{x_t^{s+1}}^{-1}(\tilde{x}^s)\right\|^2 + 2L^2\left\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x^*)\right\|^2 \\
&\leq 2L^2\left(\left\|\mathrm{Exp}_{x_t^{s+1}}^{-1}(x^*)\right\| + \left\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x^*)\right\|\right)^2 + 2L^2\left\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x^*)\right\|^2 \\
&\leq 4L^2\left\|\mathrm{Exp}_{x_t^{s+1}}^{-1}(x^*)\right\|^2 + 6L^2\left\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x^*)\right\|^2
\end{aligned}$$

We use $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ twice, in the first and fourth inequalities. The second equality is due to $\nabla f(x^*) = 0$. The second inequality is due to the L-g-smoothness assumption. The third inequality is due to triangle inequality.

Notice that $\mathbb{E}v_t^{s+1} = \nabla f(x_t^{s+1})$ and $x_{t+1}^{s+1} = \mathrm{Exp}_{x_t^{s+1}}(-\eta v_t^{s+1})$, we thus have

$$\begin{aligned}
\mathbb{E}d^2(x_{t+1}^{s+1}, x^*) &\leq d^2(x_t^{s+1}, x^*) + 2\eta\langle \mathrm{Exp}_{x_t^{s+1}}^{-1}(x^*), \mathbb{E}v_t\rangle + \zeta\eta^2\mathbb{E}\|v_t\|^2 \\
&\leq d^2(x_t^{s+1}, x^*) + 2\eta\langle \mathrm{Exp}_{x_t^{s+1}}^{-1}(x^*), \nabla f(x_t^{s+1})\rangle \\
&\qquad + \zeta\eta^2 L^2\left(4d^2(x_t^{s+1}, x^*) + 6d^2(\tilde{x}^s, x^*)\right) \\
&\leq \left(1 + 4\zeta\eta^2L^2 - \eta\mu\right) d^2(x_t^{s+1}, x^*) + 6\zeta\eta^2L^2d^2(\tilde{x}^s, x^*) \\
&\qquad + 2\eta\left(f(x^*) - f(x_t^{s+1})\right) \\
&\leq \left(1 + 4\zeta\eta^2L^2 - 2\eta\mu\right) d^2(x_t^{s+1}, x^*) + 6\zeta\eta^2L^2d^2(\tilde{x}^s, x^*)
\end{aligned}$$

The first inequality uses the trigonometric distance lemma, the second one uses previously obtained bound for $\mathbb{E}\|v_t\|^2$, the third and fourth use the μ-strong g-convexity of $f(x)$.

We now denote $u_t \triangleq \mathbb{E}d^2(x_t^{s+1}, x^*), q \triangleq 1 + 4\zeta\eta^2L^2 - 2\eta\mu, p \triangleq 6\zeta\eta^2L^2/(1-q)$. Hence by taking expectation with all the history, and noting $\tilde{x}^s = x_0^{s+1}$, we have $u_{t+1} \leq qu_t + p(1-q)u_0$, i.e. $u_{t+1} - pu_0 \leq q(u_t - pu_0)$. Therefore, $u_m - pu_0 \leq q^m(u_0 - pu_0)$, hence we get

$$u_m \leq \left(p + q^m(1-p)\right) u_0,$$

where $p + q^m(1-p) = \frac{3\zeta\eta L^2}{\mu - 2\zeta\eta L^2} + \frac{(1+4\zeta\eta^2L^2-2\eta\mu)^m(\mu - 5\zeta\eta L^2)}{\mu - 2\zeta\eta L^2} = \alpha$. It follows directly from the algorithm that after S outer loops, $\mathbb{E}d^2(x_a, x^*) = \mathbb{E}d^2(\tilde{x}^S, x^*) \leq \alpha^S d^2(x^0, x^*)$. □

Corollary 3.1. With assumptions as in Theorem 3.1 and properly chosen parameters, after $O\left((n + \frac{\zeta L^2}{\mu^2})\log(\frac{1}{\epsilon})\right)$ IFO calls, the output x_a satisfies

$$\mathbb{E}[f(x_a) - f(x^*)] \leq \epsilon.$$

Proof. Assume we choose $\eta = \mu/(17\zeta L^2)$ and $m \geq 10\zeta L^2/\mu^2$, it follows that $q = 1 - 30\mu^2/(289\zeta L^2) \leq 1 - \mu^2/(10\zeta L^2), p = 1/5$ and therefore

$$u_m \leq \left(\frac{1}{5} + \frac{4}{5}\left(1 - \mu^2/(10\zeta L^2)\right)^{10\zeta L^2/\mu^2}\right) u_0 \leq \left(\frac{1}{5} + \frac{4}{5e}\right) u_0 \leq \frac{u_0}{2},$$

where the second inequality is due to $(1-x)^{1/x} \leq 1/e$ for $x \in (0,1)$. Applying Theorem 3.1 with $\alpha = 1/2$, we have $\mathbb{E}d^2(x_a, x^*) \leq 2^{-S}d^2(x^0, x^*)$. Note that by using the L-g-smooth assumption, we also get $\mathbb{E}[f(x_a) - f(x^*)] \leq \mathbb{E}\left[\frac{1}{2}Ld^2(x_a, x^*)\right] \leq 2^{-S-1}Ld^2(x^0, x^*)$. It thus suffices to run $\log_2(Ld^2(x^0, x^*)/\epsilon) - 1$ outer loops to guar-

antee $\mathbb{E}[f(x_a) - f(x^*)] \leq \epsilon$.

For the s-th outer loop, we need n IFO calls to evaluate the full gradient at $\tilde{x}^s$, and $2m$ IFO calls when calculating each variance reduced gradient. Hence the total number of IFO calls to reach ϵ accuracy is $O\left((n + \frac{\zeta L^2}{\mu^2})\log(\frac{1}{\epsilon})\right)$. □

3.6.2 Proofs for Section 3.3.2

Theorem 3.5. *Assuming the inverse exponential map is well-defined on $\mathcal{X}$, $f : \mathcal{X} \to \mathbb{R}$ is a geodesically L-smooth function, stochastic first-order oracle $\nabla \tilde{f}(x)$ satisfies $\mathbb{E}[\nabla \tilde{f}(x^t)] = \nabla f(x^t), \|\nabla \tilde{f}(x^t)\|^2 \leq \sigma^2$, then the SGD algorithm $x^{t+1} = \mathrm{Exp}_{x^t}(-\eta \nabla \tilde{f}(x^t))$ with $\eta = c/\sqrt{T}, c = \sqrt{\frac{2(f(x^0)-f(x^*))}{L\sigma^2}}$ satisfies*

$$\min_{0\leq t\leq T-1} \mathbb{E}[\|\nabla f(x^t)\|^2] \leq \sqrt{\frac{2(f(x^0) - f(x^*))L}{T}}\sigma.$$

Proof.

$$\begin{aligned}
\mathbb{E}[f(x^{t+1})] &\leq \mathbb{E}[f(x^t) + \langle \nabla f(x^t), \mathrm{Exp}_{x^t}^{-1}(x^{t+1})\rangle + \frac{L}{2}\|\mathrm{Exp}_{x^t}^{-1}(x^{t+1})\|^2] \\
&\leq \mathbb{E}[f(x^t)] - \eta\mathbb{E}[\|\nabla f(x^t)\|^2] + \frac{L\eta^2}{2}\mathbb{E}[\|\nabla \tilde{f}(x^t)\|^2] \\
&\leq \mathbb{E}[f(x^t)] - \eta\mathbb{E}[\|\nabla f(x^t)\|^2] + \frac{L\eta^2}{2}\sigma^2
\end{aligned}$$

After rearrangement, we obtain

$$\mathbb{E}[\|\nabla f(x^t)\|^2] \leq \frac{1}{\eta}\mathbb{E}[f(x^t) - f(x^{t+1})] + \frac{L\eta}{2}\sigma^2$$

Summing up the above equation from $t = 0$ to $T - 1$ and using $\eta = c/\sqrt{T}$ where

$$c = \sqrt{\frac{2(f(x^0) - f(x^*))}{L\sigma^2}}$$

we obtain

$$\begin{aligned}
\min_t \mathbb{E}[\|\nabla f(x^t)\|^2] \leq \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|f(x^t)\|^2] &\leq \frac{1}{T\eta}\mathbb{E}[f(x^0) - f(x^T)] + \frac{L\eta}{2}\sigma^2 \\
&\leq \frac{1}{T\eta}(f(x^0) - f(x^*)) + \frac{L\eta}{2}\sigma^2 \\
&\leq \sqrt{\frac{2(f(x^0) - f(x^*))L}{T}}\sigma
\end{aligned}$$

□

Lemma 3.2. *Assume in (3.1) each f_i is L-g-smooth, the sectional curvature in $\mathcal{X}$ is lower bounded by $\kappa_{\min}$, and we run Algorithm 1 with Option II. For $c_t, c_{t+1}, \beta, \eta > 0$, suppose we have*

$$c_t = c_{t+1}\left(1 + \beta\eta + 2\zeta L^2\eta^2\right) + L^3\eta^2$$

and

$$\delta(t) = \eta - \frac{c_{t+1}\eta}{\beta} - L\eta^2 - 2c_{t+1}\zeta\eta^2 > 0,$$

then the iterate x_t^{s+1} satisfies the bound:

$$\mathbb{E}\left[\|\nabla f(x_t^{s+1})\|^2\right] \leq \frac{R_t^{s+1} - R_{t+1}^{s+1}}{\delta_t}$$

where $R_t^{s+1} := \mathbb{E}[f(x_t^{s+1}) + c_t\|\mathrm{Exp}_{\tilde{x}^s}(x_t^{s+1})\|^2]$ for $0 \leq s \leq S-1$.

Proof. Since f is L-smooth we have

$$\begin{aligned}\mathbb{E}[f(x_{t+1}^{s+1})] &\leq \mathbb{E}[f(x_t^{s+1}) + \langle\nabla f(x_t^{s+1}), \mathrm{Exp}_{x_t^{s+1}}^{-1}(x_{t+1}^{s+1})\rangle + \frac{L}{2}\|\mathrm{Exp}_{x_t^{s+1}}^{-1}(x_{t+1}^{s+1})\|^2] \\ &\leq \mathbb{E}[f(x_t^{s+1}) - \eta\|\nabla f(x_t^{s+1})\|^2 + \frac{L\eta^2}{2}\|v_t^{s+1}\|^2] \end{aligned} \tag{3.6}$$

Consider now the Lyapunov function

$$R_t^{s+1} := \mathbb{E}[f(x_t^{s+1}) + c_t\|\mathrm{Exp}_{\tilde{x}^s}(x_t^{s+1})\|^2]$$

For bounding it we will require the following:

$$\begin{aligned}\mathbb{E}[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_{t+1}^{s+1})\|^2] &\leq \mathbb{E}[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2 + \zeta\|\mathrm{Exp}_{x_t^{s+1}}^{-1}(x_{t+1}^{s+1})\|^2 \\ &\quad - 2\langle\mathrm{Exp}_{x_t^{s+1}}^{-1}(x_{t+1}^{s+1}), \mathrm{Exp}_{x_t^{s+1}}^{-1}(\tilde{x}^s)\rangle] \\ &= \mathbb{E}[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2 + \zeta\eta^2\|v_t^{s+1}\|^2 \\ &\quad + 2\eta\langle\nabla f(x_t^{s+1}), \mathrm{Exp}_{x_t^{s+1}}^{-1}(\tilde{x}^s)\rangle] \\ &\leq \mathbb{E}[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2 + \zeta\eta^2\|v_t^{s+1}\|^2] \\ &\quad + 2\eta\mathbb{E}\left[\frac{1}{2\beta}\|\nabla f(x_t^{s+1})\|^2 + \frac{\beta}{2}\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2\right] \end{aligned} \tag{3.7}$$

where the first inequality is due to Lemma 2.1, the second due to $2\langle a, b\rangle \leq \frac{1}{\beta}\|a\|^2 + \beta\|b\|^2$. Plugging Equation (3.6) and Equation (3.7) into R_{t+1}^{s+1}, we obtain the follow-

ing bound:

$$
\begin{aligned}
R_{t+1}^{s+1} &\leq \mathbb{E}[f(x_t^{s+1}) - \eta\|\nabla f(x_t^{s+1})\|^2 + \frac{L\eta^2}{2}\|v_t^{s+1}\|^2] \\
&\quad + c_{t+1}\mathbb{E}[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2 + \zeta\eta^2\|v_t^{s+1}\|^2] \\
&\quad + 2c_{t+1}\eta\mathbb{E}\left[\frac{1}{2\beta}\|\nabla f(x_t^{s+1})\|^2 + \frac{\beta}{2}\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2\right] \\
&= \mathbb{E}\left[f(x_t^{s+1}) - \left(\eta - \frac{c_{t+1}\eta}{\beta}\right)\|\nabla f(x_t^{s+1})\|^2\right] \\
&\quad + \left(\frac{L\eta^2}{2} + c_{t+1}\zeta\eta^2\right)\mathbb{E}\left[\|v_t^{s+1}\|^2\right] \\
&\quad + (c_{t+1} + c_{t+1}\eta\beta)\,\mathbb{E}\left[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2\right] \qquad (3.8)
\end{aligned}
$$

It remains to bound $\mathbb{E}\left[\|v_t^{s+1}\|^2\right]$. Denoting $\Delta_t^{s+1} = \nabla f_{i_t}(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\nabla f_{i_t}(\tilde{x}^s)$, we have $\mathbb{E}[\Delta_t^{s+1}] = \nabla f(x_t^{s+1}) - \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\nabla f(\tilde{x}^s)$, and thus

$$
\begin{aligned}
\mathbb{E}\left[\|v_t^{s+1}\|^2\right] &= \mathbb{E}\left[\|\Delta_t^{s+1} + \Gamma_{\tilde{x}^s}^{x_t^{s+1}}\nabla f(\tilde{x}^s)\|^2\right] \\
&= \mathbb{E}\left[\|\Delta_t^{s+1} - \mathbb{E}[\Delta_t^{s+1}] + \nabla f(x_t^{s+1})\|^2\right] \\
&\leq 2\mathbb{E}[\|\Delta_t^{s+1} - \mathbb{E}[\Delta_t^{s+1}]\|^2] + 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \\
&\leq 2\mathbb{E}[\|\Delta_t^{s+1}\|^2] + 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \\
&\leq 2L^2\mathbb{E}[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2] + 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \qquad (3.9)
\end{aligned}
$$

where the first inequality is due to $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, the second due to $\mathbb{E}\|\xi - \mathbb{E}\xi\|^2 = \mathbb{E}\|\xi\|^2 - \|\mathbb{E}\xi\|^2 \leq \mathbb{E}\|\xi\|^2$ for any random vector ξ in any tangent space, the third due to L-g-smooth assumption. Substituting Equation (3.9) into Equation (3.8) we get

$$
\begin{aligned}
R_{t+1}^{s+1} &\leq \mathbb{E}\left[f(x_t^{s+1}) - \left(\eta - \frac{c_{t+1}\eta}{\beta} - L\eta^2 - 2c_{t+1}\zeta\eta^2\right)\|\nabla f(x_t^{s+1})\|^2\right] \\
&\quad + \left(c_{t+1}\left(1 + \beta\eta + 2\zeta L^2\eta^2\right) + L^3\eta^2\right)\mathbb{E}\left[\|\mathrm{Exp}_{\tilde{x}^s}^{-1}(x_t^{s+1})\|^2\right] \\
&= R_t^{s+1} - \left(\eta - \frac{c_{t+1}\eta}{\beta} - L\eta^2 - 2c_{t+1}\zeta\eta^2\right)\mathbb{E}\left[\|\nabla f(x_t^{s+1})\|^2\right] \qquad (3.10)
\end{aligned}
$$

Rearranging terms completes the proof. □

Theorem 3.6. *With assumptions as in Lemma 3.2, let* $c_m = 0, \eta > 0, \beta > 0$, *and* $c_t = c_{t+1}\left(1 + \beta\eta + 2\zeta L^2\eta^2\right) + L^3\eta^2$ *such that* $\delta(t) > 0$ *for* $0 \leq t \leq m-1$. *Define the quantity*

$\delta_n := \min_t \delta(t)$, *and let* $T = mS$. *Then for the output* x_a *from Option II we have*

$$\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{f(x^0) - f(x^*)}{T\delta_n}$$

Proof. Using Lemma 3.2 and telescoping the sum, we obtain

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{R_0^{s+1} - R_m^{s+1}}{\delta_n}$$

Since $c_m = 0$ and $x_0^{s+1} = \tilde{x}^s$, we thus have

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{\mathbb{E}[f(\tilde{x}^s) - f(\tilde{x}^{s+1})]}{\delta_n}, \tag{3.11}$$

Now sum over all epochs to obtain

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{f(\tilde{x}^0) - f(x^*)}{T\delta_n} \tag{3.12}$$

Note the definition of x_a implies that the left hand side of (3.12) is exactly $\mathbb{E}[\|\nabla f(x_a)\|^2]$.

□

Theorem 3.2. Assume in (3.1) each f_i is L-g-smooth, the sectional curvature in $\mathcal{X}$ is lower bounded by $\kappa_{\min}$, and we run Algorithm 1 with Option II. Then there exist universal constants $\mu_0 \in (0,1), \nu > 0$ such that if we set $\eta = \mu_0/(Ln^{\alpha_1}\zeta^{\alpha_2})$ ($0 < \alpha_1 \leq 1$ and $0 \leq \alpha_2 \leq 2$), $m = \lfloor n^{3\alpha_1/2}/(3\mu_0\zeta^{1-2\alpha_2})\rfloor$ and $T = mS$, we have

$$\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{Ln^{\alpha_1}\zeta^{\alpha_2}[f(x^0) - f(x^*)]}{T\nu},$$

where x^* is an optimal solution to the problem in (1).

Proof. Let $\beta = L\zeta^{1-\alpha_2}/n^{\alpha_1/2}$. From the recurrence relation $c_t = c_{t+1}(1 + \beta\eta + 2\zeta L^2\eta^2) + L^3\eta^2$ and $c_m = 0$ we have

$$c_0 = \frac{\mu_0^2 L}{n^{2\alpha_1}\zeta^{2\alpha_2}} \frac{(1+\theta)^m - 1}{\theta},$$

where

$$\theta = \eta\beta + 2\zeta\eta^2L^2 = \frac{\mu_0\zeta^{1-2\alpha_2}}{n^{3\alpha_1/2}} + \frac{2\mu_0^2\zeta^{1-2\alpha_2}}{n^{2\alpha_1}} \in \left(\frac{\mu_0\zeta^{1-2\alpha_2}}{n^{3\alpha_1/2}}, \frac{3\mu_0\zeta^{1-2\alpha_2}}{n^{3\alpha_1/2}}\right).$$

Notice that $\theta < 1/m$ so that $(1+\theta)^m < e$. We can thus bound c_0 by

$$c_0 \leq \frac{\mu_0 L}{n^{\alpha_1/2}\zeta}(e-1)$$

and in turn bound δ_n by

$$\begin{aligned}
\delta_n &= \min_t \left(\eta - \frac{c_{t+1}\eta}{\beta} - \eta^2 L - 2c_{t+1}\zeta\eta^2 \right) \\
&\geq \left(\eta - \frac{c_0\eta}{\beta} - \eta^2 L - 2c_0\zeta\eta^2 \right) \\
&\geq \eta \left(1 - \frac{\mu_0(e-1)}{\zeta^{2-\alpha_2}} - \frac{\mu_0}{n^{\alpha_1}\zeta^{\alpha_2}} - \frac{2\mu_0^2(e-1)}{n^{3\alpha_1/2}\zeta^{\alpha_2}} \right) \\
&\geq \frac{\nu}{L n^{\alpha_1}\zeta^{\alpha_2}}
\end{aligned}$$

where the last inequality holds for small enough μ_0, as $\zeta, n \geq 1$. For example, it holds for $\mu_0 = 1/10, \nu = 1/20$. Substituting the above bound in Theorem 3.6 concludes the proof. □

Corollary 3.2. With assumptions and parameters in Theorem 3.2, choosing $\alpha_1 = 2/3$, the IFO complexity for achieving an ϵ-accurate solution is:

$$\text{IFO calls} = \begin{cases} O\left(n + (n^{2/3}\zeta^{1-\alpha_2}/\epsilon)\right), & \text{if } \alpha_2 \leq 1/2, \\ O\left(n\zeta^{2\alpha_2-1} + (n^{2/3}\zeta^{\alpha_2}/\epsilon)\right), & \text{if } \alpha_2 > 1/2. \end{cases}$$

Proof. Note that to reach an ϵ-accurate solution, $O(n^{\alpha_1}\zeta^{\alpha_2}/(m\epsilon)) = O(1+n^{-1/3}\zeta^{1-\alpha_2}/\epsilon)$ epochs are required. On the other hand, one epoch takes $O\left(n(1+\zeta^{2\alpha_2-1})\right)$ IFO calls. Thus the total amount of IFO calls is $O\left(n(1+\zeta^{2\alpha_2-1})(1+n^{-1/3}\zeta^{1-\alpha_2}/\epsilon)\right)$. Simplify to get the stated result. □

Theorem 3.3. Suppose that in addition to the assumptions in Theorem 3.2, f is τ-gradient dominated. Then there exist universal constants $\mu_0 \in (0,1), \nu > 0$ such that if we run Algorithm 2 with $\eta = \mu_0/(Ln^{2/3}\zeta^{1/2}), m = \lfloor n/(3\mu_0) \rfloor, S = \lceil (6 + \frac{18\mu_0}{n-3}) L\tau\zeta^{1/2}\mu_0/(\nu n^{1/3}) \rceil$, we have

$$\begin{aligned}
\mathbb{E}[\|\nabla f(x^K)\|^2] &\leq 2^{-K}\|\nabla f(x^0)\|^2, \\
\mathbb{E}[f(x^K) - f(x^*)] &\leq 2^{-K}[f(x^0) - f(x^*)].
\end{aligned}$$

Proof. Apply Theorem 3.2. Observe that for each run of Algorithm 1 with Option II we now have $T = mS \geq 2L\tau n^{2/3}\zeta^{1/2}/\nu$, which implies

$$\frac{1}{\tau}\mathbb{E}[f(x^{k+1}) - f(x^*)] \leq \mathbb{E}[\|\nabla f(x^{k+1})\|^2] \leq \frac{1}{2\tau}\mathbb{E}[f(x^k) - f(x^*)] \leq \frac{1}{2}\mathbb{E}[\|\nabla f(x^k)\|^2]$$

The theorem follows by recursive application of the above inequality. □

Corollary 3.4. With Algorithm 2 and the parameters in Theorem 3.3, the IFO complexity to compute an ϵ-accurate solution for gradient dominated function f is $O((n + L\tau\zeta^{1/2}n^{2/3})\log(1/\epsilon))$.

Proof. We need $O((n+m)S) = O(n + L\tau\zeta^{1/2}n^{2/3})$ IFO calls in a run of Algorithm 1 to double the accuracy, thus in Algorithm 2, $K = O(\log(1/\epsilon))$ runs are needed to reach ϵ-accuracy. □

Corollary 3.5. With Algorithm 2 and the parameters in Theorem 3.3, the IFO complexity to compute an ϵ-accurate solution for a μ-strongly g-convex function f is $O((n + \mu^{-1}L\zeta^{1/2}n^{2/3})\log(1/\epsilon))$.

Proof. Assume x^* is the minimizer of f and f is μ-strongly g-convex, then we have

$$\begin{aligned} f(x^*) &= \min_y f(y) \\ &\geq \min_y f(x) + \langle \nabla f(x), \mathrm{Exp}_x^{-1}(y)\rangle + \frac{\mu}{2}\|\mathrm{Exp}_x^{-1}(y)\|^2 \\ &= f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2 + \min_y \frac{1}{2\mu}\|\nabla f(x) + \mu \mathrm{Exp}_x^{-1}(y)\|^2 \\ &\geq f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2 \end{aligned}$$

where we get the first inequality by strong g-convexity, the second equality by completing the squares, and the second inequality by choosing $y = \mathrm{Exp}_x\left(-\frac{1}{\mu}\nabla f(x)\right)$. Thus $f(x)$ is $(1/(2\mu))$-gradient dominated, and choosing $\tau = 1/(2\mu)$ in Corollary 3.4 concludes the proof. □

3.6.3 Proof for Section 3.4.1

Theorem 3.4. Suppose A has eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_d$ and $\delta = \lambda_1 - \lambda_2$, and x^0 is drawn uniformly randomly on the hypersphere. Then with probability $1 - p$, x^0 falls in a Riemannian ball of a global optimum of the objective function, within which the objective function is $O(\frac{d}{p^2\delta})$-gradient dominated.

Proof. We write x in the basis of A's eigenvectors $\{v_i\}_{i=1}^d$ with corresponding eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_d$, i.e. $x = \sum_{i=1}^d \alpha_i v_i$. Thus $Ax = \sum_{i=1}^d \alpha_i\lambda_i v_i$ and

$f(x) = -\sum_{i=1}^{d} \alpha_i^2 \lambda_i$. The Riemannian gradient of $f(x)$ is $P_x \nabla f(x) = -2(I - xx^\top)Ax = -2(Ax + f(x)x) = -2\sum_{i=1}^{d} \alpha_i(\lambda_i - \sum_{j=1}^{d} \alpha_j^2 \lambda_j)v_i$ (see [Absil et al., 2009b, Example 3.6.1]). Now consider a Riemannian ball on the hypersphere defined by $\mathcal{B}_\epsilon \triangleq \{x : x \in \mathbb{S}^{d-1}, \alpha_1 \geq \epsilon\}$, note that the center of $\mathcal{B}_\epsilon$ is the first eigenvector. We apply a case by case argument with respect to $f(x) - f(x^*)$. If $f(x) - f(x^*) \geq \frac{\delta}{2}$, we can lower bound the gradient by

$$\begin{aligned}\tfrac{1}{4}\|P_x \nabla f(x)\|^2 &= \sum_{i=1}^{d} \alpha_i^2 \Big(\lambda_i - \sum_{j=1}^{d} \alpha_j^2 \lambda_j\Big)^2 \geq \alpha_1^2 \Big(\lambda_1 - \sum_{j=1}^{d} \alpha_j^2 \lambda_j\Big)^2 = \alpha_1^2 \left(f(x) - f(x^*)\right)^2 \\ &\geq \tfrac{1}{2}\alpha_1^2 \delta (f(x) - f(x^*)) \geq \tfrac{1}{2}\epsilon^2 \delta (f(x) - f(x^*))\end{aligned}$$

The last equality follows from the fact that $f(x^*) = -\lambda_1$ and $f(x) = -\sum_{i=1}^{d} \alpha_i^2 \lambda_i$. On the other hand, if $f(x) - f(x^*) < \frac{\delta}{2}$, for $i = 2, \ldots, d$, since $-\lambda_i - f(x^*) \geq \delta$, we have $-\lambda_i - f(x) > \frac{1}{2}(-\lambda_i - f(x^*)) \geq \delta/2$. We can, again, lower bound the gradient by

$$\begin{aligned}\|P_x \nabla f(x)\|^2 &= 4\sum_{i=1}^{d} \alpha_i^2 \Big(\lambda_i - \sum_{j=1}^{d} \alpha_j^2 \lambda_j\Big)^2 \geq 4\sum_{i=2}^{d} \alpha_i^2 \Big(\lambda_i - \sum_{j=1}^{d} \alpha_j^2 \lambda_j\Big)^2 \\ &\geq \sum_{i=2}^{d} \alpha_i^2 (\lambda_1 - \lambda_i)^2 \geq \delta \sum_{i=2}^{d} \alpha_i^2 (\lambda_1 - \lambda_i) = \delta(f(x) - f(x^*))\end{aligned}$$

Combining the two cases, we have that within $\mathcal{B}_\epsilon$ the objective function (3.5) is $\max\{\frac{1}{2\epsilon^2\delta}, \frac{1}{\delta}\}$-gradient dominated. Finally, observe that if x^0 is chosen uniformly at random on $\mathbb{S}^{d-1}$, then with probability at least $1 - p$, $\alpha_1^2 = \Omega(\frac{p^2}{d})$, i.e. there exists some constant $c > 0$ such that $\frac{1}{\epsilon^2} \leq \frac{cd}{p^2}$. □

4

Towards Riemannian Accelerated Gradient Method

We propose a Riemannian version of Nesterov's Accelerated Gradient algorithm (RAGD), and show that for *geodesically* smooth and strongly convex problems, within a neighborhood of the minimizer whose radius depends on the condition number as well as the sectional curvature of the manifold, RAGD converges to the minimizer with acceleration. Unlike the algorithm in [Liu et al., 2017] that requires the exact solution to a nonlinear equation which in turn may be intractable, our algorithm is constructive and computationally tractable[1]. Our proof exploits a new estimate sequence and a novel bound on the nonlinear metric distortion, both ideas

[1] as long as Riemannian gradient, exponential map and its inverse are computationally tractable, which is the case for many matrix manifolds [Absil et al., 2009b].

may be of independent interest.

4.1 Introduction

Convex optimization theory has been a fruitful area of research for decades, with classic work such as the ellipsoid algorithm [Khachiyan, 1980] and the interior point methods [Karmarkar, 1984]. However, with the rise of machine learning and data science, growing problem sizes have shifted the community's focus to first-order methods such as gradient descent and stochastic gradient descent. Over the years, impressive theoretical progress has also been made here, helping elucidate problem characteristics and bringing insights that drive the discovery of provably faster algorithms, notably Nesterov's accelerated gradient descent [Nesterov, 1983] and variance reduced incremental gradient methods [e.g., Defazio et al., 2014, Johnson and Zhang, 2013, Schmidt et al., 2013].

Outside convex optimization, however, despite some recent progress on nonconvex optimization our theoretical understanding remains limited. Nonetheless, nonconvexity pervades machine learning applications and motivates identification and study of specialized structure that enables sharper theoretical analysis, e.g., optimality bounds, global complexity, or faster algorithms. Some examples include, problems with low-rank structure [Boumal et al., 2016b, Ge et al., 2017, Kawaguchi, 2016, Sun et al., 2017]; local convergence rates [Agarwal et al., 2016, Carmon et al., 2016, Ghadimi and Lan, 2013, Reddi et al., 2016]; growth conditions that enable fast convergence [Attouch et al., 2013, Polyak, 1963, Shamir, 2015, Zhang et al., 2016]; and nonlinear constraints based on Riemannian manifolds [Boumal et al., 2016a, Mishra and Sepulchre, 2016, Zhang and Sra, 2016, Zhang et al., 2016], or more general metric spaces [Ambrosio et al., 2014, Bacák, 2014].

In this chapter, we focus on nonconvexity from a Riemannian viewpoint and consider gradient based optimization. In particular, we are motivated by Nesterov's accelerated gradient method [Nesterov, 1983], a landmark result in the the-

ory of first-order optimization. By introducing an ingenious "estimate sequence" technique, Nesterov [1983] devised a first-order algorithm that provably outperforms gradient descent, and is *optimal* (in a first-order oracle model) up to constant factors. This result bridges the gap between the lower and upper complexity bounds in smooth first-order convex optimization [Nemirovsky and Yudin, 1983, Nesterov, 2004].

Following this seminal work, other researchers also developed different analyses to explain the phenomenon of acceleration. However, both the original proof of Nesterov and all other existing analyses rely heavily on the linear structure of vector spaces. Therefore, our central question is:

Is linear space structure necessary to achieve acceleration?

Given that the iteration complexity theory of gradient descent generalizes to Riemannian manifolds [Zhang and Sra, 2016], it is tempting to hypothesize that a Riemannian generalization of accelerated gradient methods also works. However, the nonlinear nature of Riemannian geometry poses significant obstructions to either verify or refute such a hypothesis. The aim of this chapter is to study existence of accelerated gradient methods on Riemannian manifolds, while identifying and tackling key obstructions and obtaining new tools for global analysis of optimization on Riemannian manifolds as a byproduct.

It is important to note that in a recent work [Liu et al., 2017], the authors claimed to have developed Nesterov-style methods on Riemannian manifolds and analyzed their convergence rates. Unfortunately, this is *not* the case, since their algorithm requires the *exact* solution to a nonlinear equation [Liu et al., 2017, (4) and (5)] on the manifold at every iteration. In fact, solving this nonlinear equation itself can be as difficult as solving the original optimization problem.

4.1.1 Related work

The first accelerated gradient method in vector space along with the concept of estimate sequence is proposed by Nesterov [1983]; [Nesterov, 2004, Chapter 2.2.1]

contains an expository introduction. In recent years, there has been a surging interest to either develop new analysis for Nesterov's algorithm or invent new accelerated gradient methods. In particular, Flammarion and Bach [2015], Su et al. [2014], Wibisono et al. [2016] take a dynamical system viewpoint, modeling the continuous time limit of Nesterov's algorithm as a second-order ordinary differential equation. Allen-Zhu and Orecchia [2014] reinterpret Nesterov's algorithm as the linear coupling of a gradient step and a mirror descent step, which also leads to accelerated gradient methods for smoothness defined with non-Euclidean norms. Arjevani et al. [2015] reinvent Nesterov's algorithm by considering optimal methods for optimizing polynomials. Bubeck et al. [2015] develop an alternative accelerated method with a geometric explanation. Lessard et al. [2016] use theory from robust control to derive convergence rates for Nesterov's algorithm.

The design and analysis of Riemannian optimization algorithms as well as some historical perspectives were covered in details in [Absil et al., 2009b], although the analysis only focused on local convergence. The first global convergence result was derived in [Udriste, 1994] under the assumption that the Riemannian Hessian is positive definite. Zhang and Sra [2016] established the globally convergence rate of Riemannian gradient descent algorithm for optimizing geodesically convex functions on Riemannian manifolds. Other nonlocal analyses of Riemannian optimization algorithms include stochastic gradient algorithm [Zhang and Sra, 2016], fast incremental algorithm [Kasai et al., 2016, Zhang et al., 2016], proximal point algorithm [Ferreira and Oliveira, 2002] and trust-region algorithm [Boumal et al., 2016a]. Absil et al. [2009b, Chapter 2] also surveyed some important applications of Riemannian optimization.

4.1.2 Summary of results

In this chapter, we make the following contributions:

1. We propose the first *computationally tractable* accelerated gradient algorithm that, within a radius from the minimizer that depends on the condition num-

ber and sectional curvature bounds, is provably faster than gradient descent methods on Riemannian manifolds with bounded sectional curvatures. (Algorithm 4, Theorem 4.3)

2. We analyze the convergence of this algorithm using a new estimate sequence, which relaxes Nesterov's original assumption and also generalizes to Riemannian optimization. (Lemma 4.3)

3. We develop a novel bound related to the bi-Lipschitz property of exponential maps on Riemannian manifolds. This fundamental geometric result is essential for our convergence analysis, but should also have other interesting applications. (Theorem 4.2)

4.2 Background

We briefly review concepts in Riemannian geometry that are related to our analysis; for a thorough introduction one standard text is [e.g. Jost, 2011]. A *Riemannian manifold* $(\mathcal{M}, \mathfrak{g})$ is a real smooth manifold $\mathcal{M}$ equipped with a Riemannain metric $\mathfrak{g}$. The metric $\mathfrak{g}$ induces an inner product structure on each tangent space $T_x\mathcal{M}$ associated with every $x \in \mathcal{M}$. We denote the inner product of $u, v \in T_x\mathcal{M}$ as $\langle u, v \rangle \triangleq \mathfrak{g}_x(u, v)$; and the norm of $u \in T_x\mathcal{M}$ is defined as $\|u\|_x \triangleq \sqrt{\mathfrak{g}_x(u, u)}$; we omit the index x for brevity wherever it is obvious from the context. The angle between u, v is defined as $\arccos \frac{\langle u,v \rangle}{\|u\|\|v\|}$. A geodesic is a constant speed curve $\gamma : [0, 1] \to \mathcal{M}$ that is locally distance minimizing. An exponential map $\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$ maps v in $T_x\mathcal{M}$ to y on $\mathcal{M}$, such that there is a geodesic γ with $\gamma(0) = x, \gamma(1) = y$ and $\dot{\gamma}(0) \triangleq \frac{d}{dt}\gamma(0) = v$. If between any two points in $\mathcal{X} \subset \mathcal{M}$ there is a unique geodesic, the exponential map has an inverse $\mathrm{Exp}_x^{-1} : \mathcal{X} \to T_x\mathcal{M}$ and the geodesic is the unique shortest path with $\|\mathrm{Exp}_x^{-1}(y)\| = \|\mathrm{Exp}_y^{-1}(x)\|$ the geodesic distance between $x, y \in \mathcal{X}$. Parallel transport is the Riemannian analogy of vector translation, induced by the Riemannian metric.

Let $u, v \in T_x\mathcal{M}$ be linearly independent, so that they span a two dimensional

subspace of $T_x\mathcal{M}$. Under the exponential map, this subspace is mapped to a two dimensional submanifold of $\mathcal{U} \subset \mathcal{M}$. The sectional curvature $\kappa(x, \mathcal{U})$ is defined as the Gauss curvature of $\mathcal{U}$ at x, and is a critical concept in the comparison theorems involving geodesic triangles [Burago et al., 2001].

The notion of geodesically convex sets, geodesically (strongly) convex functions and geodesically smooth functions are defined as straightforward generalizations of the corresponding vector space objects to Riemannian manifolds. In particular,

- A set $\mathcal{X}$ is called *geodesically convex* if for any $x, y \in \mathcal{X}$, there is a geodesic γ with $\gamma(0) = x, \gamma(1) = y$ and $\gamma(t) \in \mathcal{X}$ for $t \in [0, 1]$.

- We call a function $f : \mathcal{X} \to \mathbb{R}$ *geodesically convex* (g-convex) if for any $x, y \in \mathcal{X}$ and any geodesic γ such that $\gamma(0) = x$, $\gamma(1) = y$ and $\gamma(t) \in \mathcal{X}$ for all $t \in [0, 1]$, it holds that
$$f(\gamma(t)) \leq (1 - t)f(x) + tf(y).$$
It can be shown that if the inverse exponential map is well-defined, an equivalent definition is that for any $x, y \in \mathcal{X}$, $f(y) \geq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y) \rangle$, where g_x is the gradient of f at x (in this work we assume f is differentiable). A function $f : \mathcal{X} \to \mathbb{R}$ is called *geodesically μ-strongly convex* (μ-strongly g-convex) if for any $x, y \in \mathcal{X}$ and gradient g_x, it holds that
$$f(y) \geq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y) \rangle + \tfrac{\mu}{2} \|\mathrm{Exp}_x^{-1}(y)\|^2.$$

- We call a vector field $g : \mathcal{X} \to \mathbb{R}^d$ *geodesically L-Lipschitz* (L-g-Lipschitz) if for any $x, y \in \mathcal{X}$,
$$\|g(x) - \Gamma_y^x g(y)\| \leq L \|\mathrm{Exp}_x^{-1}(y)\|,$$
where Γ_y^x is the parallel transport from y to x. We call a differentiable function $f : \mathcal{X} \to \mathbb{R}$ *geodesically L-smooth* (L-g-smooth) if its gradient is L-g-Lipschitz, in which case we have
$$f(y) \leq f(x) + \langle g_x, \mathrm{Exp}_x^{-1}(y) \rangle + \tfrac{L}{2} \|\mathrm{Exp}_x^{-1}(y)\|^2.$$

Throughout our analysis, for simplicity, we make the following standing assump-

tions:

Assumption 4.1. *$\mathcal{X} \subset \mathcal{M}$ is a geodesically convex set where the exponential map* Exp *and its inverse* Exp^{-1} *are well defined.*

Assumption 4.2. *The sectional curvature in $\mathcal{X}$ is bounded, i.e. $|\kappa(x, \cdot)| \leq K, \forall x \in \mathcal{X}$.*

Assumption 4.3. *f is geodesically L-smooth, μ-strongly convex, and assumes its minimum inside $\mathcal{X}$.*

Assumption 4.4. *All the iterates remain in $\mathcal{X}$.*

With these assumptions, the problem being solved can be stated formally as $\min_{x \in \mathcal{X} \subset \mathcal{M}} f(x)$.

4.3 Proposed algorithm: RAGD

Algorithm 3: Riemannian-Nesterov($x_0, \gamma_0, \{h_k\}_{k=0}^{T-1}, \{\beta_k\}_{k=0}^{T-1}$)

Parameters: initial point $x_0 \in \mathcal{X}$, $\gamma_0 > 0$, step sizes $\{h_k \leq \frac{1}{L}\}$, shrinkage parameters $\{\beta_k > 0\}$

initialize $v_0 = x_0$
for $k = 0, 1, \ldots, T-1$ **do**
  Compute $\alpha_k \in (0, 1)$ from the equation $\alpha_k^2 = h_k \cdot ((1 - \alpha_k)\gamma_k + \alpha_k \mu)$
  Set $\overline{\gamma}_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k \mu$
  Choose $y_k = \text{Exp}_{x_k}\left(\frac{\alpha_k \gamma_k}{\gamma_k + \alpha_k \mu} \text{Exp}_{x_k}^{-1}(v_k)\right)$
  Compute $f(y_k)$ and $\nabla f(y_k)$
  Set $x_{k+1} = \text{Exp}_{y_k}(-h_k \nabla f(y_k))$
  Set $v_{k+1} = \text{Exp}_{y_k}\left(\frac{(1-\alpha_k)\gamma_k}{\overline{\gamma}_{k+1}} \text{Exp}_{y_k}^{-1}(v_k) - \frac{\alpha_k}{\overline{\gamma}_{k+1}} \nabla f(y_k)\right)$
  Set $\gamma_{k+1} = \frac{1}{1+\beta_k} \overline{\gamma}_{k+1}$
end
Output: x_T

Our proposed optimization procedure is shown in Algorithm 3. We assume the algorithm is granted access to oracles that can efficiently compute the exponential map and its inverse, as well as the Riemannian gradient of function f. In comparison with Nesterov's accelerated gradient method in vector space [Nesterov, 2004, p.76], we note two important differences: first, instead of linearly combining

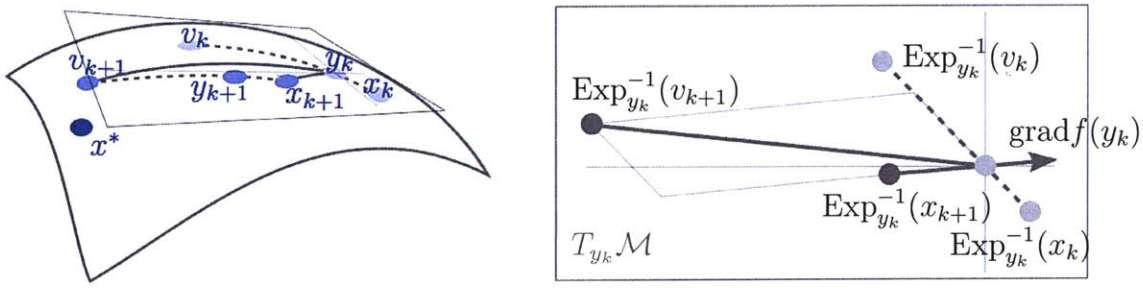


Figure 4-1: Illustration of the geometric quantities in Algorithm 3. **Left:** iterates and minimizer x^* with y_k's tangent space shown schematically. **Right:** the inverse exponential maps of relevant iterates in y_k's tangent space. Note that y_k is on the geodesic from x_k to v_k (Algorithm 3, Line 6); $\mathrm{Exp}_{y_k}^{-1}(x_{k+1})$ is in the opposite direction of $\mathrm{grad} f(y_k)$ (Algorithm 3, Line 9); also note how $\mathrm{Exp}_{y_k}^{-1}(v_{k+1})$ is constructed (Algorithm 3, Line 11).

vectors, the update for iterates is computed via exponential maps; second, we introduce a paired sequence of parameters $\{(\gamma_k, \overline{\gamma}_k)\}_{k=0}^{T-1}$, for reasons that will become clear when we analyze the convergence of the algorithm.

Algorithm 3 provides a general scheme for Nesterov-style algorithms on Riemannian manifolds, leaving the choice of many parameters to users' preference. To further simplify the parameter choice as well as the analysis, we note that the following specific choice of parameters

$$\gamma_0 \equiv \gamma = \frac{\sqrt{\beta^2 + 4(1+\beta)\mu h} - \beta}{\sqrt{\beta^2 + 4(1+\beta)\mu h} + \beta} \cdot \mu, \qquad h_k \equiv h, \forall k \geq 0, \qquad \beta_k \equiv \beta > 0, \forall k \geq 0,$$

which leads to Algorithm 4, a constant step instantiation of the general scheme. We leave the proof of this claim as a lemma in the Appendix.

Algorithm 4: Constant Step Riemannian-Nesterov(x_0, h, β)

Parameters: initial point $x_0 \in \mathcal{X}$, step size $h \leq \frac{1}{L}$, shrinkage parameter $\beta > 0$

1 initialize $v_0 = x_0$

2 set $\alpha = \frac{\sqrt{\beta^2+4(1+\beta)\mu h}-\beta}{2}$, $\gamma = \frac{\sqrt{\beta^2+4(1+\beta)\mu h}-\beta}{\sqrt{\beta^2+4(1+\beta)\mu h}+\beta} \cdot \mu$, $\overline{\gamma} = (1+\beta)\gamma$

3 **for** $k = 0, 1, \ldots, T-1$ **do**

4 Choose $y_k = \mathrm{Exp}_{x_k}\left(\frac{\alpha\gamma}{\gamma+\alpha\mu}\mathrm{Exp}_{x_k}^{-1}(v_k)\right)$

5 Set $x_{k+1} = \mathrm{Exp}_{y_k}\left(-h\nabla f(y_k)\right)$

6 Set $v_{k+1} = \mathrm{Exp}_{y_k}\left(\frac{(1-\alpha)\gamma}{\overline{\gamma}}\mathrm{Exp}_{y_k}^{-1}(v_k) - \frac{\alpha}{\overline{\gamma}}\nabla f(y_k)\right)$

7 **end**

8 **Output:** x_T

We move forward to analyzing the convergence properties of these two algorithms in the following two sections. In Section 4.4, we first provide a novel generalization of Nesterov's estimate sequence to Riemannian manifolds, then show that if a specific tangent space distance comparison inequality (4.8) always holds, then Algorithm 3 converges similarly as its vector space counterpart. In Section 4.5, we establish sufficient conditions for this tangent space distance comparison inequality to hold, specifically for Algorithm 4, and show that under these conditions Algorithm 4 converges in $O\left(\sqrt{\frac{L}{\mu}}\log(1/\epsilon)\right)$ iterations, a faster rate than the $O\left(\frac{L}{\mu}\log(1/\epsilon)\right)$ complexity of Riemannian gradient descent.

4.4 Analysis of a new estimate sequence

First introduced in [Nesterov, 1983], estimate sequences are central tools in establishing the acceleration of Nesterov's method. We first note a weaker notion of estimate sequences for functions whose domain is not necessarily a vector space.

Definition 4.1. A pair of sequences $\{\Phi_k(x) : \mathcal{X} \to \mathbb{R}\}_{k=0}^{\infty}$ and $\{\lambda_k\}_{k=0}^{\infty}$ is called a (weak) estimate sequence of a function $f(x) : \mathcal{X} \to \mathbb{R}$, if $\lambda_k \to 0$ and for all $k \geq 0$ we have:

$$\Phi_k(x^*) \leq (1-\lambda_k)f(x^*) + \lambda_k\Phi_0(x^*). \tag{4.1}$$

This definition relaxes the original definition proposed by Nesterov [2004, def. 2.2.1], in that the latter requires $\Phi_k(x) \leq (1-\lambda_k)f(x)+\lambda_k\Phi_0(x)$ to hold for all $x \in \mathcal{X}$, whereas our definition only assumes it holds at the minimizer x^*. We note that similar observations have been made, e.g., in [Carmon et al., 2017]. This relaxation is essential for sparing us from fiddling with the global geometry of Riemannian manifolds.

However, there is one major obstacle in the analysis – Nesterov's construction of quadratic function sequence critically relies on the linear metric and does not generalize to nonlinear space. An example is given in Figure 4-2, where we illustrate the distortion of distance (hence quadratic functions) in tangent spaces. The

key novelty in our construction is inequality (4.4) which allows a broader family of estimate sequences, as well as inequality (4.8) which handles nonlinear metric distortion and fulfills inequality (4.4). Before delving into the analysis of our specific construction, we recall how to construct estimate sequences and note their use in the following two lemmas.

Lemma 4.1. *Let us assume that:*

1. *f is geodesically L-smooth and μ-strongly geodesically convex on domain $\mathcal{X}$.*
2. *$\Phi_0(x)$ is an arbitrary function on $\mathcal{X}$.*
3. *$\{y_k\}_{k=0}^{\infty}$ is an arbitrary sequence in $\mathcal{X}$.*
4. *$\{\alpha_k\}_{k=0}^{\infty}$: $\alpha_k \in (0,1)$, $\sum_{k=0}^{\infty} \alpha_k = \infty$.*
5. *$\lambda_0 = 1$.*

Then the pair of sequences $\{\Phi_k(x)\}_{k=0}^{\infty}$, $\{\lambda_k\}_{k=0}^{\infty}$ which satisfy the following recursive rules:

$$\lambda_{k+1} = (1-\alpha_k)\lambda_k, \tag{4.2}$$

$$\overline{\Phi}_{k+1}(x) = (1-\alpha_k)\Phi_k(x) + \alpha_k \left[f(y_k) + \langle \nabla f(y_k), \operatorname{Exp}_{y_k}^{-1}(x) \rangle + \frac{\mu}{2} \|\operatorname{Exp}_{y_k}^{-1}(x)\|^2 \right], \tag{4.3}$$

$$\Phi_{k+1}(x^*) \leq \overline{\Phi}_{k+1}(x^*), \tag{4.4}$$

is a (weak) estimate sequence.

The proof is similar to [Nesterov, 2004, Lemma 2.2.2] which we include in Appendix 4.7.2.

Lemma 4.2. *If for a (weak) estimate sequence $\{\Phi_k(x) : \mathcal{X} \to \mathbb{R}\}_{k=0}^{\infty}$ and $\{\lambda_k\}_{k=0}^{\infty}$ we can find a sequence of iterates $\{x_k\}$, such that*

$$f(x_k) \leq \Phi_k^* \equiv \min_{x \in \mathcal{X}} \Phi_k(x),$$

then $f(x_k) - f(x^*) \leq \lambda_k(\Phi_0(x^*) - f(x^*)) \to 0$.

Proof. By Definition 4.1 we have $f(x_k) \leq \Phi_k^* \leq \Phi_k(x^*) \leq (1-\lambda_k)f(x^*) + \lambda_k \Phi_0(x^*)$. Hence $f(x_k) - f(x^*) \leq \lambda_k(\Phi_0(x^*) - f(x^*)) \to 0$. □

Lemma 4.2 immediately suggest the use of (weak) estimate sequences in establishing the convergence and analyzing the convergence rate of certain iterative algorithms. The following lemma shows that a weak estimate sequence exists for Algorithm 3. Later in Lemma 4.5, we prove that the sequence $\{x_k\}$ in Algorithm 3 satisfies the requirements in Lemma 4.2 for our estimate sequence.

Lemma 4.3. *Let* $\Phi_0(x) = \Phi_0^* + \frac{\gamma_0}{2}\|\mathrm{Exp}_{y_0}^{-1}(x)\|^2$. *Assume for all* $k \geq 0$, *the sequences* $\{\gamma_k\}$, $\{\overline{\gamma}_k\}$, $\{v_k\}$, $\{\Phi_k^*\}$ *and* $\{\alpha_k\}$ *satisfy*

$$\overline{\gamma}_{k+1} = (1-\alpha_k)\gamma_k + \alpha_k \mu, \tag{4.5}$$

$$v_{k+1} = \mathrm{Exp}_{y_k}\left(\frac{(1-\alpha_k)\gamma_k}{\overline{\gamma}_{k+1}}\mathrm{Exp}_{y_k}^{-1}(v_k) - \frac{\alpha_k}{\overline{\gamma}_{k+1}}\nabla f(y_k)\right) \tag{4.6}$$

$$\begin{aligned}\Phi_{k+1}^* = {} & (1-\alpha_k)\,\Phi_k^* + \alpha_k f(y_k) - \frac{\alpha_k^2}{2\overline{\gamma}_{k+1}}\|\nabla f(y_k)\|^2 \\ & + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\overline{\gamma}_{k+1}}\left(\frac{\mu}{2}\|\mathrm{Exp}_{y_k}^{-1}(v_k)\|^2 + \langle \nabla f(y_k), \mathrm{Exp}_{y_k}^{-1}(v_k)\rangle\right),\end{aligned} \tag{4.7}$$

$$\gamma_{k+1}\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|^2 \leq \overline{\gamma}_{k+1}\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2, \tag{4.8}$$

$$\alpha_k \in (0,1), \quad \sum_{k=0}^{\infty}\alpha_k = \infty, \tag{4.9}$$

then the pair of sequence $\{\Phi_k(x)\}_{k=0}^{\infty}$ *and* $\{\lambda_k\}_{k=0}^{\infty}$, *defined by*

$$\Phi_{k+1}(x) = \Phi_{k+1}^* + \frac{\gamma_{k+1}}{2}\|\mathrm{Exp}_{y_{k+1}}^{-1}(x) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|^2, \tag{4.10}$$

$$\lambda_0 = 1, \quad \lambda_{k+1} = (1-\alpha_k)\lambda_k. \tag{4.11}$$

is a (weak) estimate sequence.

Proof. Recall the definition of $\overline{\Phi}_{k+1}(x)$ in Equation (4.3). We claim that if $\Phi_k(x) = \Phi_k^* + \frac{\gamma_k}{2}\|\mathrm{Exp}_{y_k}^{-1}(x) - \mathrm{Exp}_{y_k}^{-1}(v_k)\|^2$, then we have $\overline{\Phi}_{k+1}(x) \equiv \Phi_{k+1}^* + \frac{\overline{\gamma}_{k+1}}{2}\|\mathrm{Exp}_{y_k}^{-1}(x) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2$. The proof of this claim requires a simple algebraic manipulation as is noted as Lemma 4.4. Now using the assumption (4.8) we immediately get $\Phi_{k+1}(x^*) \leq \overline{\Phi}_{k+1}(x^*)$. By Lemma 4.1 the proof is complete. □

We verify the specific form of $\overline{\Phi}_{k+1}(x)$ in Lemma 4.4, whose proof can be found in the Appendix 4.7.3.

Lemma 4.4. *For all $k \geq 0$, if $\Phi_k(x) = \Phi_k^* + \frac{\gamma_k}{2}\|\mathrm{Exp}_{y_k}^{-1}(x) - \mathrm{Exp}_{y_k}^{-1}(v_k)\|^2$, then with $\overline{\Phi}_{k+1}$ defined as in (4.3), $\overline{\gamma}_{k+1}$ as in (4.5), v_{k+1} as in Algorithm 3 and Φ_{k+1}^* as in (4.7) we have $\overline{\Phi}_{k+1}(x) \equiv \Phi_{k+1}^* + \frac{\overline{\gamma}_{k+1}}{2}\|\mathrm{Exp}_{y_k}^{-1}(x) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2$.*

The next lemma asserts that the iterates $\{x_k\}$ of Algorithm 3 satisfy the requirement that the function values $f(x_k)$ are upper bounded by Φ_k^* defined in our estimate sequence.

Lemma 4.5. *Assume $\Phi_0^* = f(x_0)$, and $\{\Phi_k^*\}$ be defined as in (4.7) with $\{x_k\}$ and other terms defined as in Algorithm 3. Then we have $\Phi_k^* \geq f(x_k)$ for all $k \geq 0$.*

The proof is standard. We include it in Appendix 4.7.4 for completeness. Finally, we are ready to state the following theorem on the convergence rate of Algorithm 3.

Theorem 4.1 (Convergence of Algorithm 3). *For any given $T \geq 0$, assume (4.8) is satisfied for all $0 \leq k \leq T$, then Algorithm 3 generates a sequence $\{x_k\}_{k=0}^{\infty}$ such that*

$$f(x_T) - f(x^*) \leq \lambda_T \left(f(x_0) - f(x^*) + \frac{\gamma_0}{2}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2 \right) \tag{4.12}$$

where $\lambda_0 = 1$ and $\lambda_k = \prod_{i=0}^{k-1}(1 - \alpha_i)$.

Proof. The proof is similar to [Nesterov, 2004, Theorem 2.2.1]. We choose $\Phi_0(x) = f(x_0) + \frac{\gamma_0}{2}\|\mathrm{Exp}_{y_0}^{-1}(x)\|^2$, hence $\Phi_0^* = f(x_0)$. By Lemma 4.3 and Lemma 4.5, the assumptions in Lemma 4.2 hold. It remains to use Lemma 4.2. □

4.5 Local fast rate with a constant step scheme

By now we see that almost all the analysis of Nesterov's generalizes, except that the assumption in (4.8) is not necessarily satisfied. In vector space, the two expressions both reduce to $x^* - v_{k+1}$ and hence (4.8) trivially holds with $\gamma = \overline{\gamma}$. On Riemannian manifolds, however, due to the nonlinear Riemannian metric and the associated

exponential maps, $\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|$ and $\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|$ in general do not equal (illustrated in Figure 4-2). Bounding the difference between these two quantities points the way forward for our analysis, which is also our main contribution in this section. We start with two lemmas comparing a geodesic triangle and the triangle formed by the preimage of its vertices in the tangent space, in two constant curvature spaces: hyperbolic space and the hypersphere.

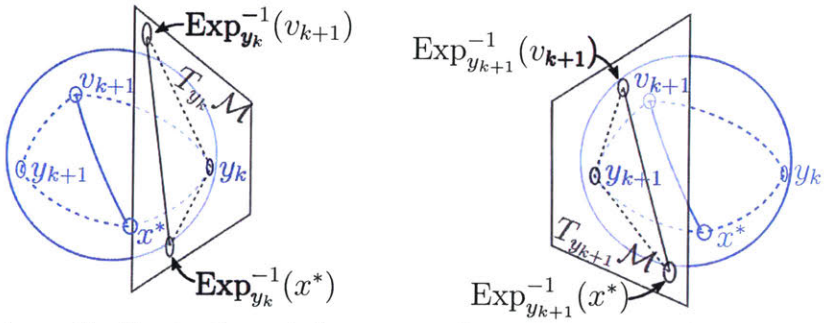


Figure 4-2: A schematic illustration of the geometric quantities in Theorem 4.2. Tangent spaces of y_k and y_{k+1} are shown in separate figures to reduce cluttering. Note that even on a sphere (which has constant positive sectional curvature), $d(x^*, v_{k+1})$, $\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|$ and $\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|$ generally do not equal.

Lemma 4.6 (bi-Lipschitzness of the exponential map in hyperbolic space). *Let a, b, c be the side lengths of a geodesic triangle in a hyperbolic space with constant sectional curvature -1, and A is the angle between sides b and c. Furthermore, assume $b \leq \frac{1}{4}, c \geq 0$. Let $\triangle \bar{a}\bar{b}\bar{c}$ be the comparison triangle in Euclidean space, with $\bar{b} = b, \bar{c} = c, \bar{A} = A$, then*

$$\bar{a}^2 \leq a^2 \leq (1 + 2b^2)\bar{a}^2. \tag{4.13}$$

Proof. The proof of this lemma contains technical details that deviate from our main focus; so we defer them to the appendix. The first inequality is well known. To show the second inequality, we have Lemma 4.9 and Lemma 4.10 (in Appendix) which in combination complete the proof. □

We also state without proof that by the same techniques one can show the following result holds.

Lemma 4.7 (bi-Lipschitzness of the exponential map on hypersphere). *Let a, b, c be the side lengths of a geodesic triangle in a hypersphere with constant sectional curvature* 1,

and A is the angle between sides b and c. Furthermore, assume $b \leq \frac{1}{4}, c \in [0, \frac{\pi}{2}]$. Let $\triangle \bar{a}\bar{b}\bar{c}$ be the comparison triangle in Euclidean space, with $\bar{b} = b, \bar{c} = c, \bar{A} = A$, then

$$a^2 \leq \bar{a}^2 \leq (1 + 2b^2)a^2. \tag{4.14}$$

Albeit very much simplified, spaces of constant curvature are important objects to study, because often their properties can be generalized to general Riemannian manifolds with bounded curvature, specifically via the use of powerful comparison theorems in metric geometry [Burago et al., 2001]. In our case, we use these two lemmas to derive a tangent space distance comparison theorem for Riemannian manifolds with bounded sectional curvature.

Theorem 4.2 (Multiplicative distortion of squared distance on Riemannian manifold). *Let x^*, v_{k+1}, y_k, $y_{k+1} \in \mathcal{X}$ be four points in a g-convex, uniquely geodesic set $\mathcal{X}$ where the sectional curvature is bounded within $[-K, K]$, for some nonnegative number K. Define $b_{k+1} = \max\left\{\|\mathrm{Exp}_{y_k}^{-1}(x^*)\|, \|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*)\|\right\}$. Assume $b_{k+1} \leq \frac{1}{4\sqrt{K}}$ for $K > 0$ (otherwise $b_{k+1} < \infty$), then we have*

$$\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|^2 \leq (1 + 5Kb_{k+1}^2)\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2. \tag{4.15}$$

Proof. The high level idea is to think of the tangent space distance distortion on Riemannian manifolds of bounded curvature as a consequence of bi-Lipschitzness of the exponential map. Specifically, note that $\triangle y_k x^* v_{k+1}$ and $\triangle y_{k+1} x^* v_{k+1}$ are two geodesic triangles in $\mathcal{X}$, whereas $\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|$ and $\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|$ are side lengths of two comparison triangles in vector space. Since $\mathcal{X}$ is of bounded sectional curvature, we can apply comparison theorems.

First, we consider bound on the distortion of squared distance in a Riemannian manifold with constant curvature $-K$. Note that in this case, the hyperbolic law of cosines becomes

$$\cosh(\sqrt{K}a) = \cosh(\sqrt{K}b)\cosh(\sqrt{K}c) - \sinh(\sqrt{K}b)\sinh(\sqrt{K}c)\cos(A),$$

which corresponds to the geodesic triangle in hyperbolic space with side lengths $\sqrt{K}a, \sqrt{K}b, \sqrt{K}c$, with the corresponding comparison triangle in Euclidean space having lengths $\sqrt{K}\bar{a}, \sqrt{K}\bar{b}, \sqrt{K}\bar{c}$. Apply Lemma 4.6 we have $(\sqrt{K}a)^2 \leq (1 +$

$2(\sqrt{K}b)^2)(\sqrt{K}\bar{a})^2$, i.e. $a^2 \leq (1+2Kb^2)\bar{a}^2$. Now consider the geodesic triangle $\triangle y_k x^* v_{k+1}$. Let $\tilde{a} = \|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|, b = \|\mathrm{Exp}_{y_k}^{-1}(v_{k+1})\| \leq b_{k+1}, c = \|\mathrm{Exp}_{y_k}^{-1}(x^*)\|, A = \angle x^* y_k v_{k+1}$, so that $\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2 = b^2 + c^2 - 2bc\cos(A)$. By Toponogov's comparison theorem [Burago et al., 2001], we have $\tilde{a} \leq a$ hence

$$\|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|^2 \leq \left(1 + 2Kb_{k+1}^2\right) \|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2. \tag{4.16}$$

Similarly, using the spherical law of cosines for a space of constant curvature K

$$\cos(\sqrt{K}a) = \cos(\sqrt{K}b)\cos(\sqrt{K}c) + \sin(\sqrt{K}b)\sin(\sqrt{K}c)\cos(A)$$

and Lemma 4.7 we can show $\bar{a}^2 \leq (1+2Kb^2)a^2$, where $\bar{a}$ is the side length in Euclidean space corresponding to a. Hence by our uniquely geodesic assumption and [Meyer, 1989, Theorem 2.2, Remark 7], with similar reasoning for the geodesic triangle $\triangle y_{k+1} x^* v_{k+1}$, we have $a \leq \|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|$, so that

$$\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|^2 \leq \left(1 + 2Kb_{k+1}^2\right) a^2 \leq \left(1 + 2Kb_{k+1}^2\right) \|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|^2. \tag{4.17}$$

Finally, combining inequalities (4.16) and (4.17), and noting that $(1+2Kb_{k+1}^2)^2 = 1 + 4Kb_{k+1}^2 + (4Kb_{k+1}^2)Kb^2 \leq 1 + 5Kb_{k+1}^2$, the proof is complete. □

Theorem 4.2 suggests that if $b_{k+1} \leq \frac{1}{4\sqrt{K}}$, we could choose $\beta \geq 5Kb_{k+1}^2$ and $\gamma \leq \frac{1}{1+\beta}\overline{\gamma}$ to guarantee $\Phi_{k+1}(x^*) \leq \overline{\Phi}_{k+1}(x^*)$. It then follows that the analysis holds for k-th step. Still, it is unknown that under what conditions can we guarantee $\Phi_{k+1}(x^*) \leq \overline{\Phi}_{k+1}(x^*)$ hold for all $k \geq 0$, which would lead to a convergence proof. We resolve this question in the next theorem.

Theorem 4.3 (Local fast convergence). *With Assumptions 4.1, 4.2, 4.3, 4.4, denote* $D = \frac{1}{20\sqrt{K}}\left(\frac{\mu}{L}\right)^{\frac{3}{4}}$ *and assume* $\mathcal{B}_{x^*,D} := \{x \in \mathcal{M} : d(x, x^*) \leq D\} \subseteq \mathcal{X}$. *If we set* $h = \frac{1}{L}, \beta = \frac{1}{5}\sqrt{\frac{\mu}{L}}$ *and* $x_0 \in \mathcal{B}_{x^*,D}$, *then Algorithm 4 converges; moreover, we have*

$$f(x_k) - f(x^*) \leq \left(1 - \frac{9}{10}\sqrt{\frac{\mu}{L}}\right)^k \left(f(x_0) - f(x^*) + \frac{\mu}{2}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2\right). \tag{4.18}$$

Proof. **sketch.** Recall that in Theorem 4.1 we already establish that if the tangent space distance comparison inequality (4.8) holds, then the general Riemannian

Nesterov iteration (Algorithm 3) and hence its constant step size special case (Algorithm 4) converge with a guaranteed rate. By the tangent space distance comparison theorem (Theorem 4.2), the comparison inequality should hold if y_k and x^* are close enough. Indeed, we use induction to assert that with a good initialization, (4.8) holds for each step. Specifically, for every $k > 0$, if y_k is close to x^* and the comparison inequality holds until the $(k-1)$-th step, then y_{k+1} is also close to x^* and the comparison inequality holds until the k-th step. We postpone the complete proof until Appendix 4.7.6. □

4.6 Discussion

In this work, we proposed a Riemannian generalization of the accelerated gradient algorithm and developed its convergence and complexity analysis. For the first time (to the best of our knowledge), we show gradient based algorithms on Riemannian manifolds can be accelerated, at least in a neighborhood of the minimizer. Central to our analysis are the two main technical contributions of our work: a new estimate sequence (Lemma 4.3), which relaxes the assumption of Nesterov's original construction and handles metric distortion on Riemannian manifolds; a tangent space distance comparison theorem (Theorem 4.2), which provides sufficient conditions for bounding the metric distortion and could be of interest for a broader range of problems on Riemannian manifolds.

Despite not matching the standard convex results, our result exposes the key difficulty of analyzing Nesterov-style algorithms on Riemannian manifolds, an aspect missing in previous work. Critically, the convergence analysis relies on bounding a new distortion term per each step. Furthermore, we observe that the side length sequence $d(y_k, v_{k+1})$ can grow much greater than $d(y_k, x^*)$, even if we reduce the "step size" h_k in Algorithm 1, defeating any attempt to control the distortion globally by modifying the algorithm parameters. This is a benign feature in vector space analysis, since (4.8) trivially holds nonetheless; however it poses a great difficulty for analysis in nonlinear space. Note the stark contrast to (stochas-

tic) gradient descent, where the step length can be effectively controlled by reducing the step size, hence bounding the distortion terms globally [Zhang and Sra, 2016].

A topic of future interest is to study whether assumption (4.8) can be further relaxed, while maintaining that overall the algorithm still converges. By bounding the squared distance distortion in every step, our analysis provides guarantee for the worst-case scenario, which seems unlikely to happen in practice. It would be interesting to conduct experiments to see how often (4.8) is violated versus how often it is loose. It would also be interesting to construct some adversarial problem case (if any) and study the complexity lower bound of gradient based Riemannian optimization, to see if geodesically convex optimization is strictly more difficult than convex optimization. Generalizing the current analysis to non-strongly g-convex functions is another interesting direction.

4.7 Proofs

4.7.1 Proof of constant step scheme

Lemma 4.8. *Pick $\beta_k \equiv \beta > 0$. If in Algorithm 3 we set*

$$h_k \equiv h, \forall k \geq 0, \qquad \gamma_0 \equiv \gamma = \frac{\sqrt{\beta^2 + 4(1+\beta)\mu h} - \beta}{\sqrt{\beta^2 + 4(1+\beta)\mu h} + \beta} \cdot \mu,$$

then we have

$$\alpha_k \equiv \alpha = \frac{\sqrt{\beta^2 + 4(1+\beta)\mu h} - \beta}{2}, \qquad \overline{\gamma}_{k+1} \equiv (1+\beta)\gamma, \qquad \gamma_{k+1} \equiv \gamma, \qquad \forall k \geq 0. \tag{4.19}$$

Proof. Suppose that $\gamma_k = \gamma$, then from Algorithm 3 we have α_k is the positive root of

$$\alpha_k^2 - (\mu - \gamma)h\alpha_k - \gamma h = 0.$$

Also note

$$\mu - \gamma = \frac{\beta\alpha}{(1+\beta)h}, \quad \text{and} \quad \gamma = \frac{\alpha^2}{(1+\beta)h}, \tag{4.20}$$

hence

$$\begin{aligned}\alpha_k &= \frac{(\mu-\gamma)h + \sqrt{(\mu-\gamma)^2h^2 + 4\gamma h}}{2} \\ &= \frac{\beta\alpha}{2(1+\beta)} + \frac{1}{2}\sqrt{\frac{\beta^2\alpha^2}{(1+\beta)^2} + \frac{4\alpha^2}{1+\beta}} \\ &= \alpha\end{aligned}$$

Furthermore, we have

$$\begin{aligned}\overline{\gamma}_{k+1} &= (1-\alpha_k)\gamma_k + \alpha_k\mu = (1-\alpha)\gamma + \alpha\mu \\ &= \gamma + (\mu-\gamma)\alpha = \gamma + \beta\frac{\alpha^2}{(1+\beta)h} \\ &= (1+\beta)\gamma\end{aligned}$$

and $\gamma_{k+1} = \frac{1}{1+\beta}\overline{\gamma}_{k+1} = \gamma$. Since $\gamma_k = \gamma$ holds for $k = 0$, by induction the proof is complete. □

4.7.2 Proof of Lemma 4.1

Proof. The proof is similar to [Nesterov, 2004, Lemma 2.2.2] except that we introduce $\overline{\Phi}_{k+1}$ as an intermediate step in constructing $\Phi_{k+1}(x)$. In fact, to start we have $\Phi_0(x) \leq (1-\lambda_0)f(x) + \lambda_0\Phi_0(x) \equiv \Phi_0(x)$. Moreover, assume (4.1) holds for some $k \geq 0$, i.e. $\Phi_k(x^*) - f(x^*) \leq \lambda_k(\Phi_0(x^*) - f(x^*))$, then

$$\begin{aligned}\Phi_{k+1}(x^*) - f(x^*) &\leq \overline{\Phi}_{k+1}(x^*) - f(x^*) \\ &\leq (1-\alpha_k)\Phi_k(x^*) + \alpha_k f(x^*) - f(x^*) \\ &= (1-\alpha_k)(\Phi_k(x^*) - f(x^*)) \\ &\leq (1-\alpha_k)\lambda_k(\Phi_0(x^*) - f(x^*)) \\ &= \lambda_{k+1}(\Phi_0(x^*) - f(x^*)),\end{aligned}$$

where the first inequality is due to our construction of $\Phi_{k+1}(x)$ in (4.4), the second inequality due to strong convexity of f. By induction we have $\Phi_k(x^*) \leq (1-\lambda_k)f(x^*) + \lambda_k\Phi_0(x^*)$ for all $k \geq 0$. It remains to note that condition 4 ensures $\lambda_k \to 0$. □

4.7.3 Proof of Lemma 4.4

Proof. We prove this lemma by completing the square:

$$\begin{aligned}
\overline{\Phi}_{k+1}(x) =& \ (1-\alpha_k)\left(\Phi_k^* + \frac{\gamma_k}{2}\|\mathrm{Exp}_{y_k}^{-1}(x) - \mathrm{Exp}_{y_k}^{-1}(v_k)\|^2\right) \\
&+ \alpha_k\left(f(y_k) + \langle \nabla f(y_k), \mathrm{Exp}_{y_k}^{-1}(x)\rangle + \frac{\mu}{2}\|\mathrm{Exp}_{y_k}^{-1}(x)\|^2\right) \\
=& \ \frac{\overline{\gamma}_{k+1}}{2}\|\mathrm{Exp}_{y_k}^{-1}(x)\|^2 + \left\langle \alpha_k \nabla f(y_k) - (1-\alpha_k)\,\gamma_k \mathrm{Exp}_{y_k}^{-1}(v_k), \mathrm{Exp}_{y_k}^{-1}(x)\right\rangle \\
&+ (1-\alpha_k)\left(\Phi_k^* + \frac{\gamma_k}{2}\|\mathrm{Exp}_{y_k}^{-1}(v_k)\|^2\right) + \alpha_k f(y_k) \\
=& \ \frac{\overline{\gamma}_{k+1}}{2}\left\|\mathrm{Exp}_{y_k}^{-1}(x) - \left(\frac{(1-\alpha_k)\gamma_k}{\overline{\gamma}_{k+1}}\mathrm{Exp}_{y_k}^{-1}(v_k) - \frac{\alpha_k}{\overline{\gamma}_{k+1}}\nabla f(y_k)\right)\right\|^2 + \Phi_{k+1}^* \\
=& \ \Phi_{k+1}^* + \frac{\overline{\gamma}_{k+1}}{2}\left\|\mathrm{Exp}_{y_k}^{-1}(x) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\right\|^2
\end{aligned}$$

where the third equality is by completing the square with respect to $\mathrm{Exp}_{y_k}^{-1}(x)$ and use the definition of Φ_{k+1}^* in (4.7), the last equality is by the definition of y_k in Algorithm 3, and $\overline{\Phi}_{k+1}(x)$ is minimized if and only if $x = \mathrm{Exp}_{y_k}\left(\frac{(1-\alpha_k)\gamma_k}{\overline{\gamma}_{k+1}}\mathrm{Exp}_{y_k}^{-1}(v_k) - \frac{\alpha_k}{\overline{\gamma}_{k+1}}\nabla f(y_k)\right) = v_{k+1}$. □

4.7.4 Proof of Lemma 4.5

Proof. For $k = 0$, $\Phi_k^* \geq f(x_k)$ trivially holds. Assume for iteration k we have $\Phi_k^* \geq f(x_k)$, then from definition (4.7) we have

$$\begin{aligned}
\Phi_{k+1}^* \geq& \ (1-\alpha_k)\, f(x_k) + \alpha_k f(y_k) - \frac{\alpha_k^2}{2\overline{\gamma}_{k+1}}\|\nabla f(y_k)\|^2 + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\overline{\gamma}_{k+1}}\langle \nabla f(y_k), \mathrm{Exp}_{y_k}^{-1}(v_k)\rangle \\
\geq& \ f(y_k) - \frac{\alpha_k^2}{2\overline{\gamma}_{k+1}}\|\nabla f(y_k)\|^2 + (1-\alpha_k)\left\langle \nabla f(y_k), \frac{\alpha_k\gamma_k}{\overline{\gamma}_{k+1}}\mathrm{Exp}_{y_k}^{-1}(v_k) + \mathrm{Exp}_{y_k}^{-1}(x_k)\right\rangle \\
=& \ f(y_k) - \frac{\alpha_k^2}{2\overline{\gamma}_{k+1}}\|\nabla f(y_k)\|^2 \\
=& \ f(y_k) - \frac{h_k}{2}\|\nabla f(y_k)\|^2,
\end{aligned}$$

where the first inequality is due to $\Phi_k^* \geq f(x_k)$, the second due to $f(x_k) \geq f(y_k) + \langle \nabla f(y_k), \mathrm{Exp}_{y_k}^{-1}(x_k)\rangle$ by g-convexity, and the equalities follow from Algorithm 3.

On the other hand, we have the bound

$$\begin{aligned} f(x_{k+1}) &\leq f(y_k) + \langle \nabla f(y_k), \mathrm{Exp}_{y_k}^{-1}(x_{k+1}) \rangle + \frac{L}{2} \|\mathrm{Exp}_{y_k}^{-1}(x_{k+1})\|^2 \\ &= f(y_k) - h_k \left(1 - \frac{Lh_k}{2}\right) \|\nabla f(y_k)\|^2 \\ &\leq f(y_k) - \frac{h_k}{2} \|\nabla f(y_k)\|^2 \leq \Phi_{k+1}^*, \end{aligned}$$

where the first inequality is by the L-smoothness assumption, the equality from the definition of x_{k+1} in Algorithm 3 Line 9, and the second inequality from the assumption that $h_k \leq \frac{1}{L}$. Hence by induction, $\Phi_k^* \geq f(x_k)$ for all $k \geq 0$. □

4.7.5 Proof of Lemma 4.6

Lemma 4.9. *Let a, b, c be the side lengths of a geodesic triangle in a hyperbolic space with constant sectional curvature -1, and A is the angle between sides b and c. Furthermore, assume $b \leq \frac{1}{4}, c \geq \frac{1}{2}$. Let $\triangle \bar{a}\bar{b}\bar{c}$ be the comparison triangle in Euclidean space, with $\bar{b} = b, \bar{c} = c, \bar{A} = A$, then*

$$a^2 \leq (1 + 2b^2)\bar{a}^2 \tag{4.21}$$

Proof. We first apply [Zhang and Sra, 2016, Lemma 5] with $\kappa = -1$ to get

$$a^2 \leq \frac{c}{\tanh(c)} b^2 + c^2 - 2bc\cos(A).$$

We also have

$$\bar{a}^2 = b^2 + c^2 - 2bc\cos(A).$$

Hence we get

$$a^2 - \bar{a}^2 \leq \left(\frac{c}{\tanh(c)} - 1\right) b^2.$$

It remains to note that for $b \leq \frac{1}{4}, c \geq \frac{1}{2}$,

$$2a^2 \geq 2(c-b)^2 \geq 2\left(c - \frac{1}{4}\right) \geq \frac{c}{\tanh(1/2)} - 1 \geq \frac{c}{\tanh(c)} - 1,$$

which implies $a^2 \leq (1 + 2b^2)\bar{a}^2$. □

Lemma 4.10. *Let a, b, c be the side lengths of a geodesic triangle in a hyperbolic space with constant sectional curvature -1, and A is the angle between sides b and c. Furthermore, assume $b \leq \frac{1}{4}, c \leq \frac{1}{2}$. Let $\triangle \bar{a}\bar{b}\bar{c}$ be the comparison triangle in Euclidean space, with*

$\bar{b} = b, \bar{c} = c, \bar{A} = A$, *then*

$$a^2 \leq (1 + b^2)\bar{a}^2 \tag{4.22}$$

Proof. Recall the law of cosines in Euclidean space and hyperbolic space:

$$\bar{a}^2 = \bar{b}^2 + \bar{c}^2 - 2\bar{b}\bar{c}\cos\bar{A}, \tag{4.23}$$

$$\cosh a = \cosh b \cosh c - \sinh b \sinh c \cos A, \tag{4.24}$$

and the Taylor series expansion:

$$\cosh x = \sum_{n=0}^{\infty} \frac{1}{(2n)!} x^{2n}, \qquad \sinh x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)!} x^{2n+1}. \tag{4.25}$$

We let $\bar{b} = b, \bar{c} = c, \bar{A} = A$, from Eq. (4.23) we have

$$\cosh \bar{a} = \cosh\left(\sqrt{b^2 + c^2 - 2bc\cos A}\right) \tag{4.26}$$

It is widely known that $\bar{a} \leq a$. Now we use Eq. (4.25) to expand the RHS of Eq. (2.6) and Eq. (4.26), and compare the coefficients for each corresponding term $b^i c^j$ in the two series. Without loss of generality, we assume $i \geq j$; the results for condition $i < j$ can be easily obtained by the symmetry of b, c. We expand Eq. (2.6) as

$$\begin{aligned}\cosh a = &\left(\sum_{n=0}^{\infty} \frac{1}{(2n)!} b^{2n}\right)\left(\sum_{n=0}^{\infty} \frac{1}{(2n)!} c^{2n}\right) \\ &- \left(\sum_{n=0}^{\infty} \frac{1}{(2n+1)!} b^{2n+1}\right)\left(\sum_{n=0}^{\infty} \frac{1}{(2n+1)!} c^{2n+1}\right)\cos A\end{aligned}$$

where the coefficient $\alpha(i, j)$ of $b^i c^j$ is

$$\alpha(i,j) = \begin{cases} \frac{1}{(2p)!(2q)!}, & \text{if } p, q \in \mathbb{N} \text{ and } i = 2p, j = 2q, \\ \frac{\cos A}{(2p+1)!(2q+1)!}, & \text{if } p, q \in \mathbb{N} \text{ and } i = 2p+1, j = 2q+1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.27}$$

Similarly, we expand Eq. (4.26) as

$$\cosh \bar{a} = \sum_{n=0}^{\infty} \frac{1}{(2n)!} \left(b^2 + c^2 - 2bc\cos A\right)^n$$

where the coefficient $\bar{\alpha}(i,j)$ of $b^i c^j$ is

$$\bar{\alpha}(i,j) = \begin{cases} \frac{\sum_{k=0}^{q} \binom{p+q}{p-k,q-k,2k} (2\cos A)^{2k}}{(2p+2q)!}, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p, j = 2q, \\ \frac{\sum_{k=0}^{q} \binom{p+q+1}{p-k,q-k,2k+1} (2\cos A)^{2k+1}}{(2p+2q+2)!}, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p+1, j = 2q+1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.28}$$

We hence calculate their absolute difference

$$\begin{aligned} & |\alpha(i,j) - \bar{\alpha}(i,j)| \\ = & \begin{cases} \frac{\sum_{k=0}^{q} \binom{p+q}{p-k,q-k,2k} 2^{2k} \left(1-(\cos A)^{2k}\right)}{(2p+2q)!}, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p, j = 2q, \\ \frac{\sum_{k=0}^{q} \binom{p+q+1}{p-k,q-k,2k+1} 2^{2k+1} \left(1-(\cos A)^{2k}\right) |\cos A|}{(2p+2q+2)!}, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p+1, j = 2q+1, \\ 0, & \text{otherwise.} \end{cases} \\ \leq & \begin{cases} \frac{\sum_{k=0}^{q} \binom{p+q}{p-k,q-k,2k} 2^{2k} k}{(2p+2q)!} \sin^2 A, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p, j = 2q, \\ \frac{\sum_{k=0}^{q} \binom{p+q+1}{p-k,q-k,2k+1} 2^{2k+1} k}{(2p+2q+2)!} \sin^2 A, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p+1, j = 2q+1, \\ 0, & \text{otherwise.} \end{cases} \\ \leq & \begin{cases} \frac{q \sum_{k=0}^{q} \binom{p+q}{p-k,q-k,2k} 2^{2k}}{(2p+2q)!} \sin^2 A, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p, j = 2q, \\ \frac{q \sum_{k=0}^{q} \binom{p+q+1}{p-k,q-k,2k+1} 2^{2k+1}}{(2p+2q+2)!} \sin^2 A, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p+1, j = 2q+1, \\ 0, & \text{otherwise.} \end{cases} \\ = & \begin{cases} \frac{q}{(2p)!(2q)!} \sin^2 A, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p, j = 2q, \\ \frac{q}{(2p+1)!(2q+1)!} \sin^2 A, & \text{if } p,q \in \mathbb{N} \text{ and } i = 2p+1, j = 2q+1, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4.29}$$

where the two equalities are due to Lemma 4.11, the first inequality due to the following fact

$$\begin{aligned} 1 - (\cos A)^{2m} &= \left(1 - (\cos A)^2\right) \left(1 + (\cos A)^2 + (\cos A)^4 + \cdots + (\cos A)^{2(m-1)}\right) \\ &= \sin^2 A \left(1 + (\cos A)^2 + (\cos A)^4 + \cdots + (\cos A)^{2(m-1)}\right) \leq m \sin^2 A \end{aligned}$$

By setting $q = 0$, we see that in the Taylor series of $\cosh a - \cosh \bar{a}$, any term that does not include a factor of c^2 cancels out. By the symmetry of b, c, any term that does not include a factor of b^2 also cancels out. The term with the lowest order of power is thus $\frac{1}{4} b^2 c^2 \sin^2 A$. Since we have $c \leq \frac{1}{2}, b \leq \frac{1}{4}$, the terms $|\alpha(i,j) - \bar{\alpha}(i,j)| b^i c^j$

must satisfy

$$\begin{aligned}\sum_{i,j} |\alpha(i,j) - \bar{\alpha}(i,j)| b^i c^j &\leq \left(\frac{1}{4} + \sum_{\substack{i+j=2k,\\ i,j\geq 2, k\geq 3}} \frac{i+j}{2(i!)(j!)} \frac{1}{2^{2k-4}} \right) b^2 c^2 \sin^2 A \\ &\leq \left(\frac{1}{4} + \sum_{k\geq 3} \frac{1}{2^{2k-3}} \right) b^2 c^2 \sin^2 A \leq \frac{1}{2} b^2 c^2 \sin^2 A \\ &= \frac{1}{2} b^2 \bar{a}^2 \sin^2 C \leq \frac{1}{2} \bar{a}^2 b^2\end{aligned}$$

where the first inequality follows from Eq. (4.29) and is due to $\min(p,q) \leq \frac{i+j}{2}$, the second inequality is due to $\sum_{\substack{i+j=2k\\ i\geq 2, j\geq 2}} \frac{i+j}{(i!)(j!)} \leq \frac{(2k)^2}{(k!)^2} \leq 1$ for $k \geq 3$ and the last equality is due to Euclidean law of sines. We thus get

$$\cosh a - \cosh \bar{a} \leq \sum_{i,j} |\alpha(i,j) - \bar{\alpha}(i,j)| b^i c^j \sin^2 A \leq \frac{1}{2} b^2 \bar{a}^2 \tag{4.30}$$

On the other hand, from the Taylor series of $\cosh$ we have

$$\cosh a - \cosh \bar{a} = \sum_{n=0}^{\infty} \frac{a^{2n} - \bar{a}^{2n}}{(2n)!} \geq \frac{1}{2}(a^2 - \bar{a}^2),$$

hence $a^2 \leq (1+b^2)\bar{a}^2$.

□

Lemma 4.11 (Two multinomial identities). *For $p, q \in \mathbb{N}, p \geq q$, we have*

$$\frac{(2p+2q)!}{(2p)!(2q)!} = \sum_{k=0}^{q} \binom{p+q}{p-k, q-k, 2k} 2^{2k} \tag{4.31}$$

$$\frac{(2p+2q+2)!}{(2p+1)!(2q+1)!} = \sum_{k=0}^{q} \binom{p+q+1}{p-k, q-k, 2k+1} 2^{2k+1} \tag{4.32}$$

Proof. We prove the identities by showing that the LHS and RHS correspond to two equivalent ways of counting the same quantity. For the first identity, consider a set of $2p+2q$ balls b_i each with a unique index $i = 1, \ldots, 2p+2q$, we count how many ways we can put them into boxes B_1 and B_2, such that B_1 has $2p$ balls and B_2 has $2q$ balls. The LHS is obviously a correct count. To get the RHS, note that we can first put balls in pairs, then decide what to do with each pair. Specifically, there are $p+q$ pairs $\{b_{2i-1}, b_{2i}\}$, and we can partition the counts by the number of pairs of which we put one of the two balls in B_2. Note that this number must be even.

If there are $2k$ such pairs, which gives us $2k$ balls in B_2, we still need to choose $2(q-k)$ pairs of which both balls are put in B_2, and the left are $p-k$ pairs of which both balls are put in B_1. The total number of counts given k is thus

$$\binom{p+q}{p-k, q-k, 2k} 2^{2k}$$

because we can choose either ball in each of the $2k$ pairs leading to 2^{2k} possible choices. Summing over k we get the RHS. Hence the LHS and the RHS equal. The second identity can be proved with essentially the same argument. □

4.7.6 Proof of Theorem 4.3

Proof. The base case. First we verify that y_0, y_1 is sufficiently close to x^* so that the comparison inequality (4.8) holds at step $k = 0$. In fact, since $y_0 = x_0$ by construction, we have

$$\|\mathrm{Exp}_{y_0}^{-1}(x^*)\| = \|\mathrm{Exp}_{x_0}^{-1}(x^*)\| \leq \frac{1}{4\sqrt{K}}, \qquad 5K\|\mathrm{Exp}_{y_0}^{-1}(x^*)\|^2 \leq \frac{1}{80}\left(\frac{\mu}{L}\right)^{\frac{3}{2}} \leq \beta. \tag{4.33}$$

To bound $\|\mathrm{Exp}_{y_1}^{-1}(x^*)\|$, observe that y_1 is on the geodesic between x_1 and v_1. So first we bound $\|\mathrm{Exp}_{x_1}^{-1}(x^*)\|$ and $\|\mathrm{Exp}_{v_1}^{-1}(x^*)\|$. Bound on $\|\mathrm{Exp}_{x_1}^{-1}(x^*)\|$ comes from strong g-convexity:

$$\begin{aligned}\|\mathrm{Exp}_{x_1}^{-1}(x^*)\|^2 &\leq \frac{2}{\mu}(f(x_1) - f(x^*)) \leq \frac{2}{\mu}(f(x_0) - f(x^*)) + \frac{\gamma}{\mu}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2 \\ &\leq \frac{L+\gamma}{\mu}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2,\end{aligned}$$

whereas bound on $\|\mathrm{Exp}_{v_1}^{-1}(x^*)\|$ utilizes the tangent space distance comparison theorem. First, from the definition of $\overline{\Phi}_1$ we have

$$\|\mathrm{Exp}_{y_0}^{-1}(x^*) - \mathrm{Exp}_{y_0}^{-1}(v_1)\|^2 = \frac{2}{\gamma}(\overline{\Phi}_1(x^*) - \Phi_1^*) \leq \frac{2}{\gamma}(\Phi_0(x^*) - f(x^*)) \leq \frac{L+\gamma}{\gamma}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2$$

Then note that (4.33) implies that the assumption in Theorem 4.2 is satisfied when $k = 0$, thus we have

$$\|\mathrm{Exp}_{v_1}^{-1}(x^*)\|^2 \leq (1+\beta)\|\mathrm{Exp}_{y_0}^{-1}(x^*) - \mathrm{Exp}_{y_0}^{-1}(v_1)\|^2 \leq \frac{2(L+\gamma)}{\gamma}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2.$$

Together we have

$$\begin{aligned}
\|\mathrm{Exp}_{y_1}^{-1}(x^*)\| &\leq \|\mathrm{Exp}_{x_1}^{-1}(x^*)\| + \frac{\alpha\gamma}{\gamma+\alpha\mu}\|\mathrm{Exp}_{x_1}^{-1}(v_1)\| \\
&\leq \|\mathrm{Exp}_{x_1}^{-1}(x^*)\| + \frac{\alpha\gamma}{\gamma+\alpha\mu}\left(\|\mathrm{Exp}_{x_1}^{-1}(x^*)\| + \|\mathrm{Exp}_{v_1}^{-1}(x^*)\|\right) \\
&\leq \sqrt{\frac{L+\gamma}{\mu}}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\| + \frac{\alpha\gamma}{\gamma+\alpha\mu}\left(\sqrt{\frac{L+\gamma}{\mu}} + \sqrt{\frac{2(L+\gamma)}{\mu}}\right)\|\mathrm{Exp}_{x_0}^{-1}(x^*)\| \\
&\leq \left(1+\frac{1+\sqrt{2}}{2}\right)\sqrt{\frac{L+\gamma}{\mu}}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\| \\
&\leq \frac{1}{10\sqrt{K}}\left(\frac{\mu}{L}\right)^{\frac{1}{4}} \leq \frac{1}{4\sqrt{K}} \qquad (4.34)
\end{aligned}$$

which also implies

$$5K\|\mathrm{Exp}_{y_1}^{-1}(x^*)\|^2 \leq \frac{1}{20}\sqrt{\frac{\mu}{L}} \leq \beta \qquad (4.35)$$

By (4.34), (4.35) and Theorem 4.2 it is hence guaranteed that

$$\gamma\|\mathrm{Exp}_{y_1}^{-1}(x^*) - \mathrm{Exp}_{y_1}^{-1}(v_1)\|^2 \leq \overline{\gamma}\|\mathrm{Exp}_{y_0}^{-1}(x^*) - \mathrm{Exp}_{y_0}^{-1}(v_1)\|^2.$$

The inductive step. Assume that for $i = 0, \ldots, k-1$, (4.8) hold simultaneously, i.e.:

$$\gamma\|\mathrm{Exp}_{y_{i+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{i+1}}^{-1}(v_{i+1})\|^2 \leq \overline{\gamma}\|\mathrm{Exp}_{y_i}^{-1}(x^*) - \mathrm{Exp}_{y_i}^{-1}(v_{i+1})\|^2, \forall i = 0, \ldots, k-1$$

and also that $\|\mathrm{Exp}_{y_k}^{-1}(x^*)\| \leq \frac{1}{10\sqrt{K}}\left(\frac{\mu}{L}\right)^{\frac{1}{4}}$. To bound $\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*)\|$, observe that y_{k+1} is on the geodesic between x_{k+1} and v_{k+1}. So first we bound $\|\mathrm{Exp}_{x_{k+1}}^{-1}(x^*)\|$ and $\|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|$. Note that due to the sequential nature of the algorithm, statements about any step only depend on its previous steps, but not any step afterwards. Since (4.8) hold for steps $i = 0, \ldots, k-1$, the analysis in the previous section already applies for steps $i = 0, \ldots, k-1$. Therefore by Theorem 4.1 and the proof of Lemma 4.5 we know

$$\begin{aligned}
f(x^*) \leq f(x_{k+1}) \leq \Phi_{k+1}^* \leq \Phi_{k+1}(x^*) &\leq f(x^*) + (1-\alpha)^{k+1}(\Phi_0(x^*) - f(x^*)) \\
&\leq \Phi_0(x^*) = f(x_0) + \frac{\gamma}{2}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2
\end{aligned}$$

Hence we get $f(x_{k+1}) - f(x^*) \leq \Phi_0(x^*) - f(x^*)$ and $\frac{\gamma}{2}\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2 \equiv \overline{\Phi}_{k+1}(x^*) - \Phi_{k+1}^* \leq \Phi_0(x^*) - f(x^*)$. Bound on $\|\mathrm{Exp}_{x_{k+1}}^{-1}(x^*)\|$ comes from strong

g-convexity:

$$\begin{aligned}\|\mathrm{Exp}_{x_{k+1}}^{-1}(x^*)\|^2 &\leq \frac{2}{\mu}(f(x_{k+1}) - f(x^*)) \leq \frac{2}{\mu}(f(x_0) - f(x^*)) + \frac{\gamma}{\mu}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2 \\ &\leq \frac{L+\gamma}{\mu}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2,\end{aligned}$$

whereas bound on $\|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|$ utilizes the tangent space distance comparison theorem. First, from the definition of $\overline{\Phi}_{k+1}$ we have

$$\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2 = \frac{2}{\gamma}(\overline{\Phi}_{k+1}(x^*) - \Phi_{k+1}^*) \leq \frac{2}{\gamma}(\Phi_0(x^*) - f(x^*)) \leq \frac{L+\gamma}{\gamma}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2$$

Then note that the inductive hypothesis implies that

$$\|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|^2 \leq (1+\beta)\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2 \leq \frac{2(L+\gamma)}{\gamma}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\|^2$$

Together we have

$$\begin{aligned}\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*)\| \leq\ & \|\mathrm{Exp}_{x_{k+1}}^{-1}(x^*)\| + \frac{\alpha\gamma}{\gamma+\alpha\mu}\|\mathrm{Exp}_{x_{k+1}}^{-1}(v_{k+1})\| \\ \leq\ & \|\mathrm{Exp}_{x_{k+1}}^{-1}(x^*)\| + \frac{\alpha\gamma}{\gamma+\alpha\mu}\left(\|\mathrm{Exp}_{x_{k+1}}^{-1}(x^*)\| + \|\mathrm{Exp}_{v_{k+1}}^{-1}(x^*)\|\right) \\ \leq\ & \sqrt{\frac{L+\gamma}{\mu}}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\| + \frac{\alpha\gamma}{\gamma+\alpha\mu}\left(\sqrt{\frac{L+\gamma}{\mu}} + \sqrt{\frac{2(L+\gamma)}{\mu}}\right)\|\mathrm{Exp}_{x_0}^{-1}(x^*)\| \\ \leq\ & \left(1 + \frac{1+\sqrt{2}}{2}\right)\sqrt{\frac{L+\gamma}{\mu}}\|\mathrm{Exp}_{x_0}^{-1}(x^*)\| \\ \leq\ & \frac{1}{10\sqrt{K}}\left(\frac{\mu}{L}\right)^{\frac{1}{4}} \leq \frac{1}{4\sqrt{K}}\end{aligned}$$

which also implies that

$$5K\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*)\|^2 \leq \frac{1}{20}\sqrt{\frac{\mu}{L}} \leq \beta$$

By the two lines of equations above and Theorem 4.2 it is guaranteed that $\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*)\| \leq \frac{1}{10\sqrt{K}}\left(\frac{\mu}{L}\right)^{\frac{1}{4}}$ and also

$$\gamma\|\mathrm{Exp}_{y_{k+1}}^{-1}(x^*) - \mathrm{Exp}_{y_{k+1}}^{-1}(v_{k+1})\|^2 \leq \overline{\gamma}\|\mathrm{Exp}_{y_k}^{-1}(x^*) - \mathrm{Exp}_{y_k}^{-1}(v_{k+1})\|^2.$$

i.e. (4.8) hold for $i = 0, \ldots, k$. This concludes the inductive step.

By induction, (4.8) hold for all $k \geq 0$, hence by Theorem 4.1, Algorithm 4 converges,

with

$$\alpha_i \equiv \alpha = \frac{\sqrt{\beta^2 + 4(1+\beta)\mu h} - \beta}{2} = \frac{\sqrt{\mu h}}{2}\left(\sqrt{\frac{1}{25} + 4\left(1 + \frac{\sqrt{\mu h}}{5}\right)} - \frac{1}{5}\right) \geq \frac{9}{10}\sqrt{\frac{\mu}{L}}.$$

□

5

mixup: Beyond Empirical Risk Minimization

Large deep neural networks are powerful, but exhibit undesirable behaviors such as memorization and sensitivity to adversarial examples. In this work, we propose Mixup, a simple learning principle to alleviate these issues. In essence, Mixup trains a neural network on convex combinations of pairs of examples and their labels. By doing so, Mixup regularizes the neural network to favor simple linear behavior in-between training examples. Our experiments on the ImageNet-2012, CIFAR-10, CIFAR-100, Google commands and UCI datasets show that Mixup improves the generalization of state-of-the-art neural network architectures. We also find that Mixup reduces the memorization of corrupt labels, increases the robustness to adversarial examples, and stabilizes the training of generative adversarial

networks.

5.1 Introduction

Large deep neural networks have enabled breakthroughs in fields such as computer vision [Krizhevsky et al., 2012], speech recognition [Hinton et al., 2012], and reinforcement learning [Silver et al., 2016]. In most successful applications, these neural networks share two commonalities. First, they are trained as to minimize their average error over the training data, a learning rule also known as the Empirical Risk Minimization (ERM) principle [Vapnik, 1998]. Second, the size of these state-of-the-art neural networks scales linearly with the number of training examples. For instance, the network of Springenberg et al. [2015] used 10^6 parameters to model the $5 \cdot 10^4$ images in the CIFAR-10 dataset, the network of [Simonyan and Zisserman, 2015] used 10^8 parameters to model the 10^6 images in the ImageNet-2012 dataset, and the network of Chelba et al. [2013] used $2 \cdot 10^{10}$ parameters to model the 10^9 words in the One Billion Word dataset.

Strikingly, a classical result in learning theory [Vapnik and Chervonenkis, 1971] tells us that the convergence of ERM is guaranteed as long as the size of the learning machine (e.g., the neural network) does not increase with the number of training data. Here, the size of a learning machine is measured in terms of its number of parameters or, relatedly, its VC-complexity [Harvey et al., 2017].

This contradiction challenges the suitability of ERM to train our current neural network models, as highlighted in recent research. On the one hand, ERM allows large neural networks to *memorize* (instead of *generalize* from) the training data even in the presence of strong regularization, or in classification problems where the labels are assigned at random [Zhang et al., 2017]. On the other hand, neural networks trained with ERM change their predictions drastically when evaluated on examples just outside the training distribution [Szegedy et al., 2014], also known as *adversarial examples*. This evidence suggests that ERM is unable to explain or provide generalization on testing distributions that differ *only slightly* from

the training data. However, what is the alternative to ERM?

The method of choice to train on similar but different examples to the training data is known as *data augmentation* [Simard et al., 1998], formalized by the Vicinal Risk Minimization (VRM) principle [Chapelle et al., 2000]. In VRM, human knowledge is required to describe a *vicinity* or neighborhood around each example in the training data. Then, additional *virtual* examples can be drawn from the vicinity distribution of the training examples to enlarge the support of the training distribution. For instance, when performing image classification, it is common to define the vicinity of one image as the set of its horizontal reflections, slight rotations, and mild scalings. While data augmentation consistently leads to improved generalization [Simard et al., 1998], the procedure is dataset-dependent, and thus requires the use of expert knowledge. Furthermore, data augmentation assumes that the examples in the vicinity share the same class, and does not model the vicinity relation across examples of different classes.

Contribution Motivated by these issues, we introduce a simple and data-agnostic data augmentation routine, termed Mixup (Section 5.2). In a nutshell, Mixup constructs virtual training examples

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda) y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$. Therefore, Mixup extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. Mixup can be implemented in a few lines of code, and introduces minimal computation overhead.

Despite its simplicity, Mixup allows a new state-of-the-art performance in the CIFAR-10, CIFAR-100, and ImageNet-2012 image classification datasets (Sections 5.3.1 and 5.3.2). Furthermore, Mixup increases the robustness of neural networks when learning from corrupt labels (Section 5.3.4), or facing adversarial examples (Sec-

tion 5.3.5). Finally, Mixup improves generalization on speech (Sections 5.3.3) and tabular (Section 5.3.6) data, and can be used to stabilize the training of GANs (Section 5.3.7). The source-code necessary to replicate our CIFAR-10 experiments is available at:

https://github.com/facebookresearch/mixup-cifar10.

To understand the effects of various design choices in Mixup, we conduct a thorough set of ablation study experiments (Section 5.3.8). The results suggest that Mixup performs significantly better than related methods in previous work, and each of the design choices contributes to the final performance. We conclude by exploring the connections to prior work (Section 5.4), as well as offering some points for discussion (Section 5.5).

5.2 From Empirical Risk Minimization to Mixup

In supervised learning, we are interested in finding a function $f \in \mathcal{F}$ that describes the relationship between a random feature vector X and a random target vector Y, which follow the joint distribution $P(X, Y)$. To this end, we first define a loss function ℓ that penalizes the differences between predictions $f(x)$ and actual targets y, for examples $(x, y) \sim P$. Then, we minimize the average of the loss function ℓ over the data distribution P, also known as the *expected risk*:

$$R(f) = \int \ell(f(x), y) \mathrm{d}P(x, y).$$

Unfortunately, the distribution P is unknown in most practical situations. Instead, we usually have access to a set of training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $(x_i, y_i) \sim P$ for all $i = 1, \ldots, n$. Using the training data $\mathcal{D}$, we may approximate P by the *empirical distribution*

$$P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i),$$

where $\delta(x = x_i, y = y_i)$ is a Dirac mass centered at (x_i, y_i). Using the empirical distribution P_δ, we can now approximate the expected risk by the *empirical risk*:

$$R_\delta(f) = \int \ell(f(x), y) \mathrm{d}P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i). \tag{5.1}$$

Learning the function f by minimizing equation 5.1 is known as the Empirical Risk Minimization (ERM) principle [Vapnik, 1998]. While efficient to compute, the empirical risk equation 5.1 monitors the behaviour of f only at a finite set of n examples. When considering functions with a number parameters comparable to n (such as large neural networks), one trivial way to minimize equation 5.1 is to memorize the training data [Zhang et al., 2017]. Memorization, in turn, leads to the undesirable behaviour of f outside the training data [Szegedy et al., 2014].

However, the naïve estimate P_δ is one out of many possible choices to approximate the true distribution P. For instance, in the *Vicinal Risk Minimization* (VRM) principle [Chapelle et al., 2000], the distribution P is approximated by

$$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^{n} \nu(\tilde{x}, \tilde{y} | x_i, y_i),$$

where ν is a *vicinity distribution* that measures the probability of finding the *virtual* feature-target pair $(\tilde{x}, \tilde{y})$ in the *vicinity* of the training feature-target pair (x_i, y_i). In particular, Chapelle et al. [2000] considered Gaussian vicinities $\nu(\tilde{x}, \tilde{y} | x_i, y_i) = \mathcal{N}(\tilde{x} - x_i, \sigma^2)\delta(\tilde{y} = y_i)$, which is equivalent to augmenting the training data with additive Gaussian noise. To learn using VRM, we sample the vicinal distribution to construct a dataset $\mathcal{D}_\nu := \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{m}$, and minimize the *empirical vicinal risk*:

$$R_\nu(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f(\tilde{x}_i), \tilde{y}_i).$$

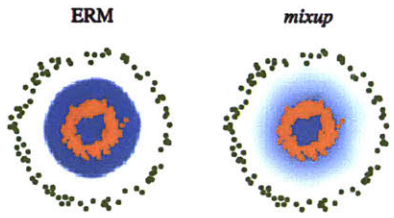
The contribution of this chapter is to propose a generic vicinal distribution, called Mixup:

$$\mu(\tilde{x}, \tilde{y} | x_i, y_i) = \frac{1}{n} \sum_{j}^{n} \mathbb{E}_\lambda \left[\delta(\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j) \right],$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$. In a nutshell, sampling from the Mixup

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of Mixup training in PyTorch.



(b) Effect of Mixup ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Figure 5-1: Illustration of Mixup, which converges to ERM as $\alpha \to 0$.

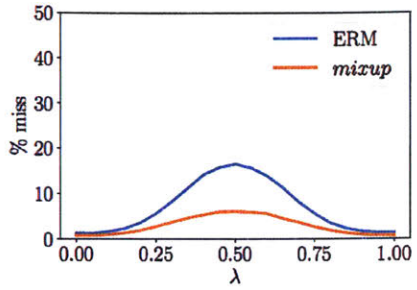
vicinal distribution produces virtual feature-target vectors

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j,$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j,$$

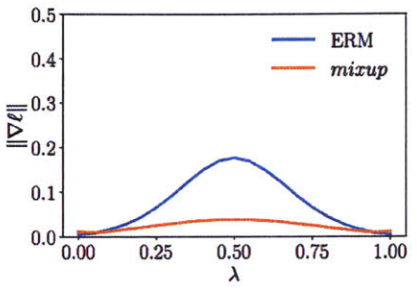
where (x_i, y_i) and (x_j, y_j) are two feature-target vectors drawn at random from the training data, and $\lambda \in [0, 1]$. The Mixup hyper-parameter α controls the strength of interpolation between feature-target pairs, recovering the ERM principle as $\alpha \to 0$.

The implementation of Mixup training is straightforward, and introduces a minimal computation overhead. Figure 5-1a shows the few lines of code necessary to implement Mixup training in PyTorch. Finally, we mention alternative design choices. First, in preliminary experiments we find that convex combinations of three or more examples with weights sampled from a Dirichlet distribution does not provide further gain, but increases the computation cost of Mixup. Second, our current implementation uses a single data loader to obtain one minibatch, and then Mixup is applied to the same minibatch after random shuffling. We found this strategy works equally well, while reducing I/O requirements. Third, interpolating only between inputs with equal label did not lead to the performance gains of Mixup discussed in the sequel. More empirical comparison can be found in Section 5.3.8.

What is Mixup doing? The Mixup vicinal distribution can be understood as a form of data augmentation that encourages the model f to behave linearly in-



(a) Prediction errors in-between training data. Evaluated at $x = \lambda x_i + (1 - \lambda)x_j$, a prediction is counted as a "miss" if it does not belong to $\{y_i, y_j\}$. The model trained with Mixup has fewer misses.



(b) Norm of the gradients of the model w.r.t. input in-between training data, evaluated at $x = \lambda x_i + (1 - \lambda)x_j$. The model trained with Mixup has smaller gradient norms.

Figure 5-2: Mixup leads to more robust model behaviors in-between the training data.

between training examples. We argue that this linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples. Also, linearity is a good inductive bias from the perspective of Occam's razor, since it is one of the simplest possible behaviors. Figure 5-1b shows that Mixup leads to decision boundaries that transition linearly from class to class, providing a smoother estimate of uncertainty. Figure 5-2 illustrate the average behaviors of two neural network models trained on the CIFAR-10 dataset using ERM and Mixup. Both models have the same architecture, are trained with the same procedure, and are evaluated at the same points in-between randomly sampled training data. The model trained with Mixup is more stable in terms of model predictions and gradient norms in-between training samples.

5.3 Experiments

5.3.1 ImageNet classification

We evaluate Mixup on the ImageNet-2012 classification dataset [Russakovsky et al., 2015]. This dataset contains 1.3 million training images and 50,000 validation images, from a total of 1,000 classes. For training, we follow standard data augmen-

Model	Method	Epochs	Top-1 Error	Top-5 Error
ResNet-50	ERM [Goyal et al., 2017]	90	23.5	-
	Mixup $\alpha = 0.2$	90	**23.3**	**6.6**
ResNet-101	ERM [Goyal et al., 2017]	90	22.1	-
	Mixup $\alpha = 0.2$	90	**21.5**	**5.6**
ResNeXt-101 32*4d	ERM [Xie et al., 2016]	100	21.2	-
	ERM	90	21.2	5.6
	Mixup $\alpha = 0.4$	90	**20.7**	**5.3**
ResNeXt-101 64*4d	ERM [Xie et al., 2016]	100	20.4	5.3
	Mixup $\alpha = 0.4$	90	**19.8**	**4.9**
ResNet-50	ERM	200	23.6	7.0
	Mixup $\alpha = 0.2$	200	**22.1**	**6.1**
ResNet-101	ERM	200	22.0	6.1
	Mixup $\alpha = 0.2$	200	**20.8**	**5.4**
ResNeXt-101 32*4d	ERM	200	21.3	5.9
	Mixup $\alpha = 0.4$	200	**20.1**	**5.0**

Table 5.1: Validation errors for ERM and Mixup on the development set of ImageNet-2012.

tation practices: scale and aspect ratio distortions, random crops, and horizontal flips [Goyal et al., 2017]. During evaluation, only the 224×224 central crop of each image is tested. We use Mixup and ERM to train several state-of-the-art ImageNet-2012 classification models, and report both top-1 and top-5 error rates in Table 5.1.

For all the experiments in this section, we use data-parallel distributed training in Caffe2[1] with a minibatch size of 1,024. We use the learning rate schedule described in [Goyal et al., 2017]. Specifically, the learning rate is increased linearly from 0.1 to 0.4 during the first 5 epochs, and it is then divided by 10 after 30, 60 and 80 epochs when training for 90 epochs; or after 60, 120 and 180 epochs when training for 200 epochs.

For Mixup, we find that $\alpha \in [0.1, 0.4]$ leads to improved performance over ERM, whereas for large α, Mixup leads to underfitting. We also find that models with higher capacities and/or longer training runs are the ones to benefit the most from Mixup. For example, when trained for 90 epochs, the Mixup variants of ResNet-101 and ResNeXt-101 obtain a greater improvement (0.5% to 0.6%) over their ERM analogues than the gain of smaller models such as ResNet-50 (0.2%). When trained

[1] `https://caffe2.ai`

for 200 epochs, the top-1 error of the Mixup variant of ResNet-50 is further reduced by 1.2% compared to the 90 epoch run, whereas its ERM analogue stays the same.

5.3.2 CIFAR-10 and CIFAR-100

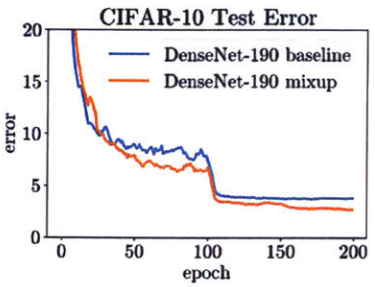
We conduct additional image classification experiments on the CIFAR-10 and CIFAR-100 datasets to further evaluate the generalization performance of Mixup. In particular, we compare ERM and Mixup training for: PreAct ResNet-18 [He et al., 2016b] as implemented in [Liu, 2017], WideResNet-28-10 [Zagoruyko and Komodakis, 2016b] as implemented in [Zagoruyko and Komodakis, 2016a], and DenseNet [Huang et al., 2017] as implemented in [Veit, 2017]. For DenseNet, we change the growth rate to 40 to follow the DenseNet-BC-190 specification from [Huang et al., 2017]. For Mixup, we fix $\alpha = 1$, which results in interpolations λ uniformly distributed between zero and one. All models are trained on a single Nvidia Tesla P100 GPU using PyTorch[2] for 200 epochs on the training set with 128 examples per minibatch, and evaluated on the test set. Learning rates start at 0.1 and are divided by 10 after 100 and 150 epochs for all models except WideResNet. For WideResNet, we follow [Zagoruyko and Komodakis, 2016b] and divide the learning rate by 10 after 60, 120 and 180 epochs. Weight decay is set to 10^{-4}. We do not use dropout in these experiments.

We summarize our results in Figure 5-3a. In both CIFAR-10 and CIFAR-100 classification problems, the models trained using Mixup significantly outperform their analogues trained with ERM. As seen in Figure 5-3b, Mixup and ERM converge at a similar speed to their best test errors. Note that the DenseNet models in [Huang et al., 2017] were trained for 300 epochs with further learning rate decays scheduled at the 150 and 225 epochs, which may explain the discrepancy the performance of DenseNet reported in Figure 5-3a and the original result of Huang et al. [2017].

[2] `http://pytorch.org`

Dataset	Model	ERM	Mixup
CIFAR-10	PreAct ResNet-18	5.6	**4.2**
	WideResNet-28-10	3.8	**2.7**
	DenseNet-BC-190	3.7	**2.7**
CIFAR-100	PreAct ResNet-18	25.6	**21.1**
	WideResNet-28-10	19.4	**17.5**
	DenseNet-BC-190	19.0	**16.8**

(a) Test errors for the CIFAR experiments.



(b) Test error evolution for the best ERM and Mixup models.

Figure 5-3: Test errors for ERM and Mixup on the CIFAR experiments.

5.3.3 Speech data

Next, we perform speech recognition experiments using the Google commands dataset [Warden, 2017]. The dataset contains 65,000 utterances, where each utterance is about one-second long and belongs to one out of 30 classes. The classes correspond to voice commands such as *yes, no, down, left*, as pronounced by a few thousand different speakers. To preprocess the utterances, we first extract normalized spectrograms from the original waveforms at a sampling rate of 16 kHz. Next, we zero-pad the spectrograms to equalize their sizes at 160×101. For speech data, it is reasonable to apply Mixup both at the waveform and spectrogram levels. Here, we apply Mixup at the spectrogram level just before feeding the data to the network.

For this experiment, we compare a LeNet [Lecun et al., 2001] and a VGG-11 [Simonyan and Zisserman, 2015] architecture, each of them composed by two convolutional and two fully-connected layers. We train each model for 30 epochs with minibatches of 100 examples, using Adam as the optimizer [Kingma and Ba, 2014]. Training starts with a learning rate equal to 3×10^{-3} and is divided by 10 every 10 epochs. For Mixup, we use a warm-up period of five epochs where we train the network on original training examples, since we find it speeds up initial convergence. Table 5-4 shows that Mixup outperforms ERM on this task, specially when using VGG-11, the model with larger capacity.

Model	Method	Validation set	Test set
LeNet	ERM	**9.8**	**10.3**
	Mixup ($\alpha = 0.1$)	10.1	10.8
	Mixup ($\alpha = 0.2$)	10.2	11.3
VGG-11	ERM	5.0	4.6
	Mixup ($\alpha = 0.1$)	4.0	3.8
	Mixup ($\alpha = 0.2$)	**3.9**	**3.4**

Figure 5-4: Classification errors of ERM and Mixup on the Google commands dataset.

5.3.4 Memorization of corrupted labels

Following Zhang et al. [2017], we evaluate the robustness of ERM and Mixup models against randomly corrupted labels. We hypothesize that increasing the strength of Mixup interpolation α should generate virtual examples further from the training examples, making memorization more difficult to achieve. In particular, it should be easier to learn interpolations between real examples compared to memorizing interpolations involving random labels. We adapt an open-source implementation [Zhang, 2017] to generate three CIFAR-10 training sets, where 20%, 50%, or 80% of the labels are replaced by random noise, respectively. All the test labels are kept intact for evaluation. Dropout [Srivastava et al., 2014] is considered the state-of-the-art method for learning with corrupted labels [Arpit et al., 2017]. Thus, we compare in these experiments Mixup, dropout, Mixup + dropout, and ERM. For Mixup, we choose $\alpha \in \{1, 2, 8, 32\}$; for dropout, we add one dropout layer in each PreAct block after the ReLU activation layer between two convolution layers, as suggested in [Zagoruyko and Komodakis, 2016b]. We choose the dropout probability $p \in \{0.5, 0.7, 0.8, 0.9\}$. For the combination of Mixup and dropout, we choose $\alpha \in \{1, 2, 4, 8\}$ and $p \in \{0.3, 0.5, 0.7\}$. These experiments use the PreAct ResNet-18 [He et al., 2016b] model implemented in [Liu, 2017]. All the other settings are the same as in Section 5.3.2.

We summarize our results in Table 5.2, where we note the best test error achieved during the training session, as well as the final test error after 200 epochs. To quantify the amount of memorization, we also evaluate the training errors at the last

Label corruption	Method	Test error		Training error	
		Best	Last	Real	Corrupted
20%	ERM	12.7	16.6	0.05	0.28
	ERM + dropout ($p = 0.7$)	8.8	10.4	5.26	83.55
	Mixup ($\alpha = 8$)	**5.9**	6.4	2.27	86.32
	Mixup + dropout ($\alpha = 4, p = 0.1$)	6.2	**6.2**	1.92	85.02
50%	ERM	18.8	44.6	0.26	0.64
	ERM + dropout ($p = 0.8$)	14.1	15.5	12.71	86.98
	Mixup ($\alpha = 32$)	11.3	12.7	5.84	85.71
	Mixup + dropout ($\alpha = 8, p = 0.3$)	**10.9**	**10.9**	7.56	87.90
80%	ERM	36.5	73.9	0.62	0.83
	ERM + dropout ($p = 0.8$)	30.9	35.1	29.84	86.37
	Mixup ($\alpha = 32$)	25.3	30.9	18.92	85.44
	Mixup + dropout ($\alpha = 8, p = 0.3$)	**24.0**	**24.8**	19.70	87.67

Table 5.2: Results on the corrupted label experiments for the best models.

epoch on real labels and corrupted labels. As the training progresses with a smaller learning rate (e.g. less than 0.01), the ERM model starts to overfit the corrupted labels. When using a large probability (e.g. 0.7 or 0.8), dropout can effectively reduce overfitting. Mixup with a large α (e.g. 8 or 32) outperforms dropout on both the best and last epoch test errors, and achieves lower training error on real labels while remaining resistant to noisy labels. Interestingly, Mixup + dropout performs the best of all, showing that the two methods are compatible.

5.3.5 Robustness to adversarial examples

One undesirable consequence of models trained using ERM is their fragility to adversarial examples [Szegedy et al., 2014]. Adversarial examples are obtained by adding tiny (visually imperceptible) perturbations to legitimate examples in order to deteriorate the performance of the model. The adversarial noise is generated by ascending the gradient of the loss surface with respect to the legitimate example. Improving the robustness to adversarial examples is a topic of active research.

Among the several methods aiming to solve this problem, some have proposed to penalize the norm of the Jacobian of the model to control its Lipschitz constant [Bartlett et al., 2017, Cisse et al., 2017, Drucker and LeCun, 1992, Hein and

Andriushchenko, 2017]. Other approaches perform data augmentation by producing and training on adversarial examples [Goodfellow et al., 2015]. Unfortunately, all of these methods add significant computational overhead to ERM. Here, we show that Mixup can significantly improve the robustness of neural networks without hindering the speed of ERM by penalizing the norm of the gradient of the loss w.r.t a given input along the most plausible directions (e.g. the directions to other training points). Indeed, Figure 5-2 shows that Mixup results in models having a smaller loss and gradient norm between examples compared to vanilla ERM.

To assess the robustness of Mixup models to adversarial examples, we use three ResNet-101 models: two of them trained using ERM on ImageNet-2012, and the third trained using Mixup. In the first set of experiments, we study the robustness of one ERM model and the Mixup model against white box attacks. That is, for each of the two models, we use the model itself to generate adversarial examples, either using the Fast Gradient Sign Method (FGSM) or the Iterative FGSM (I-FGSM) methods [Goodfellow et al., 2015], allowing a maximum perturbation of $\epsilon = 4$ for every pixel. For I-FGSM, we use 10 iterations with equal step size. In the second set of experiments, we evaluate robustness against black box attacks. That is, we use the first ERM model to produce adversarial examples using FGSM and I-FGSM. Then, we test the robustness of the second ERM model and the Mixup model to these examples. The results of both settings are summarized in Table 5.3.

For the FGSM white box attack, the Mixup model is 2.7 times more robust than the ERM model in terms of Top-1 error. For the FGSM black box attack, the Mixup model is 1.25 times more robust than the ERM model in terms of Top-1 error. Also, while both Mixup and ERM are not robust to white box I-FGSM attacks, Mixup is about 40% more robust than ERM in the black box I-FGSM setting. Overall, Mixup produces neural networks that are significantly more robust than ERM against adversarial examples in white box and black settings without additional overhead compared to ERM.

Metric	Method	FGSM	I-FGSM
Top-1	ERM	90.7	99.9
	Mixup	**75.2**	99.6
Top-5	ERM	63.1	93.4
	Mixup	**49.1**	95.8

(a) White box attacks.

Metric	Method	FGSM	I-FGSM
Top-1	ERM	57.0	57.3
	Mixup	**46.0**	**40.9**
Top-5	ERM	24.8	18.1
	Mixup	**17.4**	**11.8**

(b) Black box attacks.

Table 5.3: Classification errors of ERM and Mixup models when tested on adversarial examples.

Dataset	ERM	Mixup
Abalone	74.0	73.6
Arcene	57.6	**48.0**
Arrhythmia	56.6	**46.3**

Dataset	ERM	Mixup
Htru2	2.0	2.0
Iris	21.3	**17.3**
Phishing	16.3	15.2

Table 5.4: ERM and Mixup classification errors on the UCI datasets.

5.3.6 Tabular data

To further explore the performance of Mixup on non-image data, we performed a series of experiments on six arbitrary classification problems drawn from the UCI dataset [Dheeru and Karra Taniskidou, 2017]. The neural networks in this section are fully-connected, and have two hidden layers of 128 ReLU units. The parameters of these neural networks are learned using Adam [Kingma and Ba, 2014] with default hyper-parameters, over 10 epochs of mini-batches of size 16. Table 5.4 shows that Mixup improves the average test error on four out of the six considered datasets, and never underperforms ERM.

5.3.7 Stabilization of Generative Adversarial Networks (GANs)

Generative Adversarial Networks, also known as GANs [Goodfellow et al., 2014], are a powerful family of implicit generative models. In GANs, a generator and a discriminator compete against each other to model a distribution P. On the one

hand, the generator g competes to transform noise vectors $z \sim Q$ into fake samples $g(z)$ that resemble real samples $x \sim P$. On the other hand, the discriminator competes to distinguish between real samples x and fake samples $g(z)$. Mathematically, training a GAN is equivalent to solving the optimization problem

$$\max_g \min_d \mathbb{E}_{x,z}\, \ell(d(x), 1) + \ell(d(g(z)), 0),$$

where ℓ is the binary cross entropy loss. Unfortunately, solving the previous min-max equation is a notoriously difficult optimization problem [Goodfellow, 2016], since the discriminator often provides the generator with vanishing gradients. We argue that Mixup should stabilize GAN training because it acts as a regularizer on the gradients of the discriminator, akin to the binary classifier in Figure 5-1b. Then, the smoothness of the discriminator guarantees a stable source of gradient information to the generator. The Mixup formulation of GANs is:

$$\max_g \min_d \mathbb{E}_{x,z,\lambda}\, \ell(d(\lambda x + (1-\lambda)g(z)), \lambda).$$

Figure 5-5 illustrates the stabilizing effect of Mixup the training of GAN (orange samples) when modeling two toy datasets (blue samples). The neural networks in these experiments are fully-connected and have three hidden layers of 512 ReLU units. The generator network accepts two-dimensional Gaussian noise vectors. The networks are trained for 20,000 mini-batches of size 128 using the Adam optimizer with default parameters, where the discriminator is trained for five iterations before every generator iteration. The training of Mixup GANs seems promisingly robust to hyper-parameter and architectural choices.

5.3.8 Ablation Studies

5.3.8.1 Comparison with alternative designs

Mixup is a data augmentation method that consists of only two parts: random convex combination of raw inputs, and correspondingly, convex combination of one-hot label encodings. However, there are several design choices to make. For example, on how to augment the inputs, we could have chosen to interpolate the

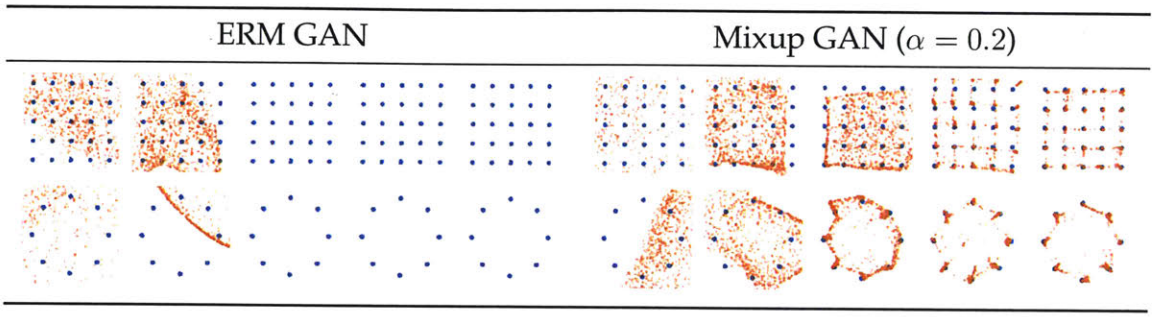


Figure 5-5: Effect of Mixup on stabilizing GAN training at iterations 10, 100, 1000, 10000, and 20000.

latent representations (i.e. feature maps) of a neural network, and we could have chosen to interpolate only between the nearest neighbors, or only between inputs of the same class. When the inputs to interpolate come from two different classes, we could have chosen to assign a single label to the synthetic input, for example using the label of the input that weights more in the convex combination. To compare Mixup with these alternative possibilities, we run a set of ablation study experiments using the PreAct ResNet-18 architecture on the CIFAR-10 dataset.

Specifically, for each of the data augmentation methods, we test two weight decay settings (10^{-4} which works well for Mixup, and 5×10^{-4} which works well for ERM). All the other settings and hyperparameters are the same as reported in Section 5.3.2.

To compare interpolating raw inputs with interpolating latent representations, we test on random convex combination of the learned representations before each residual block (denoted Layer 1-4) or before the uppermost "average pooling + fully connected" layer (denoted Layer 5). To compare mixing random pairs of inputs (RP) with mixing nearest neighbors (KNN), we first compute the 200 nearest neighbors for each training sample, either from the same class (SC) or from all the classes (AC). Then during training, for each sample in a minibatch, we replace the sample with a synthetic sample by convex combination with a random draw from its nearest neighbors. To compare mixing all the classes (AC) with mixing within the same class (SC), we convex combine a minibatch with a random permutation of its sample index, where the permutation is done in a per-batch basis (AC) or a

Method	Specification	Modified		Weight decay	
		Input	Target	10^{-4}	5×10^{-4}
ERM		✗	✗	5.53	5.18
Mixup	AC + RP	✓	✓	**4.24**	4.68
	AC + KNN	✓	✓	4.98	5.26
mix labels and latent representations (AC + RP)	Layer 1	✓	✓	4.44	**4.51**
	Layer 2	✓	✓	4.56	4.61
	Layer 3	✓	✓	5.39	5.55
	Layer 4	✓	✓	5.95	5.43
	Layer 5	✓	✓	5.39	5.15
mix inputs only	SC + KNN [Chawla et al., 2002]	✓	✗	5.45	5.52
	AC + KNN	✓	✗	5.43	5.48
	SC + RP	✓	✗	5.23	5.55
	AC + RP	✓	✗	5.17	5.72
label smoothing [Szegedy et al., 2016]	$\epsilon = 0.05$	✗	✓	5.25	5.02
	$\epsilon = 0.1$	✗	✓	5.33	5.17
	$\epsilon = 0.2$	✗	✓	5.34	5.06
mix inputs + label smoothing (AC + RP)	$\epsilon = 0.05$	✓	✓	5.02	5.40
	$\epsilon = 0.1$	✓	✓	5.08	5.09
	$\epsilon = 0.2$	✓	✓	4.98	5.06
	$\epsilon = 0.4$	✓	✓	5.25	5.39
add Gaussian noise to inputs	$\sigma = 0.05$	✓	✗	5.53	5.04
	$\sigma = 0.1$	✓	✗	6.41	5.86
	$\sigma = 0.2$	✓	✗	7.16	7.24

Table 5.5: Results of the ablation studies on the CIFAR-10 dataset. Reported are the median test errors of the last 10 epochs. A tick (✓) means the component is different from standard ERM training, whereas a cross (✗) means it follows the standard training practice. AC: mix between all classes. SC: mix within the same class. RP: mix between random pairs. KNN: mix between k-nearest neighbors (k=200). Please refer to the text for details about the experiments and interpretations.

per-class basis (SC). To compare mixing inputs and labels with mixing inputs only, we either use a convex combination of the two one-hot encodings as the target, or select the one-hot encoding of the closer training sample as the target. For label smoothing, we follow Szegedy et al. [2016] and use $\frac{\epsilon}{10}$ as the target for incorrect classes, and $1 - \frac{9\epsilon}{10}$ as the target for the correct class. Adding Gaussian noise to inputs is used as another baseline. We report the median test errors of the last 10 epochs. Results are shown in Table 5.5.

From the ablation study experiments, we have the following observations. First, Mixup is the best data augmentation method we test, and is significantly better than the second best method (mix input + label smoothing). Second, the effect of regularization can be seen by comparing the test error with a small weight decay (10^{-4}) with a large one (5×10^{-4}). For example, for ERM a large weight decay works better, whereas for Mixup a small weight decay is preferred, confirming its regularization effects. We also see an increasing advantage of large weight decay when interpolating in higher layers of latent representations, indicating decreasing strength of regularization. Among all the input interpolation methods, mixing random pairs from all classes (AC + RP) has the strongest regularization effect. Label smoothing and adding Gaussian noise have a relatively small regularization effect. Finally, we note that the SMOTE algorithm [Chawla et al., 2002] does not lead to a noticeable gain in performance.

5.3.8.2 Effects of dataset size

5.4 Related Work

Data augmentation lies at the heart of all successful applications of deep learning, ranging from image classification [Krizhevsky et al., 2012] to speech recognition [Amodei et al., 2016, Graves et al., 2013]. In all cases, substantial domain knowledge is leveraged to design suitable data transformations leading to improved generalization. In image classification, for example, one routinely uses rotation, translation, cropping, resizing, flipping [Lecun et al., 2001, Simonyan and

Zisserman, 2015], and random erasing [Zhong et al., 2017] to enforce visually plausible invariances in the model through the training data. Similarly, in speech recognition, noise injection is a prevalent practice to improve the robustness and accuracy of the trained models [Amodei et al., 2016].

More related to Mixup, Chawla et al. [2002] propose to augment the rare class in an imbalanced dataset by interpolating the nearest neighbors; DeVries and Taylor [2017b] show that interpolation and extrapolation the nearest neighbors of the same class in feature space can improve generalization. However, their proposals only operate among the nearest neighbors within a certain class at the input / feature level, and hence does not account for changes in the corresponding labels. Recent approaches have also proposed to regularize the output distribution of a neural network by label smoothing [Szegedy et al., 2016], or penalizing high-confidence softmax distributions [Pereyra et al., 2017]. These methods bear similarities with Mixup in the sense that supervision depends on multiple smooth labels, rather than on single hard labels as in traditional ERM. However, the label smoothing in these works is applied or regularized independently from the associated feature values.

Mixup enjoys several desirable aspects of previous data augmentation and regularization schemes without suffering from their drawbacks. Like the method of DeVries and Taylor [2017b], it does not require significant domain knowledge. Like label smoothing, the supervision of every example is not overly dominated by the ground-truth label. Unlike both of these approaches, the Mixup transformation establishes a linear relationship between data augmentation and the supervision signal. We believe that this leads to a strong regularizer that improves generalization as demonstrated by our experiments. The linearity constraint, through its effect on the derivatives of the function approximated, also relates Mixup to other methods such as Sobolev training of neural networks [Czarnecki et al., 2017] or WGAN-GP [Gulrajani et al., 2017].

5.5 Discussion

We have proposed Mixup, a data-agnostic and straightforward data augmentation principle. We have shown that Mixup is a form of vicinal risk minimization, which trains on virtual examples constructed as the linear interpolation of two random examples from the training set and their labels. Incorporating Mixup into existing training pipelines reduces to a few lines of code, and introduces little or no computational overhead. Throughout an extensive evaluation, we have shown that Mixup improves the generalization error of state-of-the-art models on ImageNet, CIFAR, speech, and tabular datasets. Furthermore, Mixup helps to combat memorization of corrupt labels, sensitivity to adversarial examples, and instability in adversarial training.

In our experiments, the following trend is consistent: with increasingly large α, the training error on real data increases, while the generalization gap decreases. This sustains our hypothesis that Mixup implicitly controls model complexity. However, we do not yet have a good theory for understanding the 'sweet spot' of this bias-variance trade-off. For example, in CIFAR-10 classification we can get very low training error on real data even when $\alpha \to \infty$ (i.e., training *only* on averages of pairs of real examples), whereas in ImageNet classification, the training error on real data increases significantly with $\alpha \to \infty$. Based on our ImageNet and Google commands experiments with different model architectures, we conjecture that increasing the model capacity would make training error less sensitive to large α, hence giving Mixup a more significant advantage.

Mixup also opens up several possibilities for further exploration. First, is it possible to make similar ideas work on other types of supervised learning problems, such as regression and structured prediction? While generalizing Mixup to regression problems is straightforward, its application to structured prediction problems such as image segmentation remains less obvious. Second, can similar methods prove helpful beyond supervised learning? The interpolation principle seems like a reasonable inductive bias which might also help in unsupervised,

semi-supervised, and reinforcement learning. Can we extend Mixup to feature-label extrapolation to guarantee a robust model behavior far away from the training data? Although our discussion of these directions is still speculative, we are excited about the possibilities Mixup opens up, and hope that our observations will prove useful for future development.

6

ZeroInit: Training Deep Residual Networks without Normalization

Normalization layers are a staple in state-of-the-art deep neural network architectures. They are widely believed to stabilize training, enable higher learning rate, accelerate convergence and improve generalization, though the reason for their effectiveness is still an active research topic. In this work, we challenge the commonly-held beliefs by showing that *none* of the perceived benefits is unique to normalization. Specifically, we propose ZeroInit, an initialization motivated by solving the exploding and vanishing gradient problem at the beginning of training by properly rescaling a standard initialization. We find training residual networks with ZeroInit to be as stable as training with normalization - even for networks with 10,000 layers. Furthermore, with proper regularization, ZeroInit *with-*

out normalization matches or exceeds the performance of state-of-the-art residual networks in image classification and machine translation.

6.1 Introduction

Artificial intelligence applications have witnessed major advances in recent years [Hinton et al., 2012, Krizhevsky et al., 2012, Sutskever et al., 2014]. At the core of this revolution is the development of novel neural network models and their training techniques. For example, since the landmark work of He et al. [2016a], most of the state-of-the-art image recognition systems are built upon a deep stack of network blocks consisting of convolutional layers and additive skip connections, with some normalization mechanism (e.g. batch normalization [Ioffe and Szegedy, 2015]) to facilitate training and generalization. Besides image classification, various normalization techniques [Ba et al., 2016, Salimans and Kingma, 2016, Ulyanov et al., 2016, Wu and He, 2018] have been found essential to achieving good performance on other tasks, such as machine translation [Vaswani et al., 2017] and generative modeling [Zhu et al., 2017]. They are widely believed to have multiple benefits for training very deep neural networks, including stabilizing learning, enabling higher learning rate, accelerating convergence, and improving generalization.

Despite the enormous empirical success of training deep networks with skip connections with normalization, there is currently no general consensus on why these normalization techniques help the training process [Santurkar et al., 2018]. Intrigued by this topic, in this work we study

(i) *without* normalization, can a deep residual network be trained reliably? (And if so,)

(ii) *without* normalization, can a deep residual network be trained with the same learning rate, converge at the same speed, and generalize equally well (or even better)?

Perhaps surprisingly, we find the answers to both questions are *Yes*. In particu-

lar, we show:

- **Why normalization helps training.** We derive a lower bound for the gradient norm of a residual network at initialization, which explains why with standard initializations, normalization techniques are *essential* for training deep residual networks at maximal learning rate. (Section 6.2)
- **Training without normalization.** We propose ZeroInit, a method that rescales the standard initialization of residual branches by adjusting for the network architecture. ZeroInit enables training very deep residual networks stably at maximal learning rate without normalization. (Section 6.3)
- **Image classification.** We apply ZeroInit to replace batch normalization on image classification benchmarks CIFAR-10 (with Wide-ResNet) and ImageNet (with ResNet), and find ZeroInit with proper regularization matches the well-tuned baseline trained with normalization. (Section 6.4.2)
- **Machine translation.** We apply ZeroInit to replace layer normalization on machine translation benchmarks IWSLT and WMT using the Transformer model, and find it outperforms the baseline and achieves new state-of-the-art results. (Section 6.4.3)

In the remaining of this chapter, we first analyze the exploding gradient problem of residual networks at initialization in Section 6.2. To solve this problem, we develop ZeroInit in Section 6.3. In Section 6.4 we quantify the properties of ZeroInit and compare it against state-of-the-art normalization methods on real world benchmarks. A comparison with related work is presented in Section 6.5.

6.2 Problem: ResNet with Standard Initializations Lead to Exploding Gradients

Standard initialization methods [Glorot and Bengio, 2010, He et al., 2015, Xiao et al., 2018] attempt to set the initial parameters of the network such that the activa-

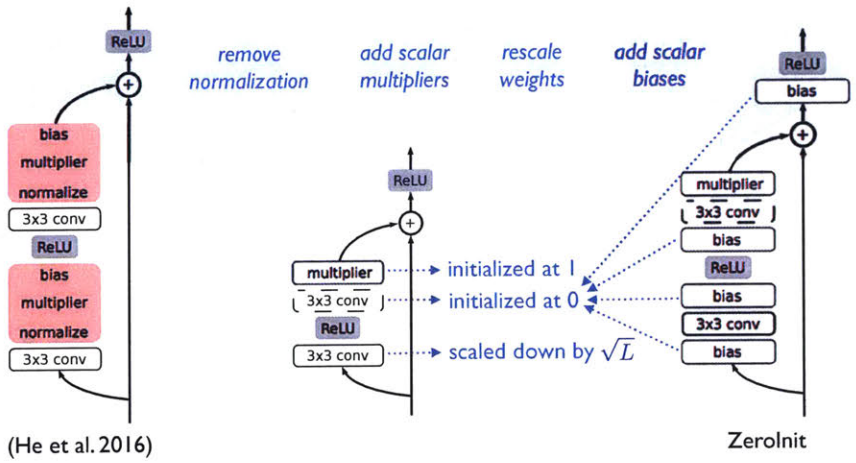


Figure 6-1: Left: ResNet basic block. Batch normalization [Ioffe and Szegedy, 2015] layers are marked in red. **Middle:** A simple network block that trains stably when stacked together. **Right:** ZeroInit further improves by adding bias parameters. (See Section 6.3 for details.)

tions neither vanish nor explode. Unfortunately, it has been observed that without normalization techniques such as BatchNorm they do not account properly for the effect of residual connections and this causes exploding gradients. Balduzzi et al. [2017] characterizes this problem for ReLU networks, and we will generalize this to residual networks with positively homogenous activation functions. A plain (i.e. without normalization layers) ResNet with residual blocks $\{F_1, \ldots, F_L\}$ and input $\mathbf{x}_0$ computes the activations as

$$\mathbf{x}_l = \mathbf{x}_0 + \sum_{i=0}^{l-1} F_i(\mathbf{x}_i). \tag{6.1}$$

ResNet output variance grows exponentially with depth. Here we only consider the initialization, view the input $\mathbf{x}_0$ as fixed, and consider the randomness of the weight initialization. We analyze the variance of each layer $\mathbf{x}_l$, denoted by $\mathrm{Var}[\mathbf{x}_l]$ (which is technically defined as the sum of the variance of all the coordinates of $\mathbf{x}_l$.) For simplicity we assume the blocks are initialized to be zero mean, i.e., $\mathbb{E}[F_l(\mathbf{x}_l) \mid \mathbf{x}_l] = 0$. By $\mathbf{x}_{l+1} = \mathbf{x}_l + F_l(\mathbf{x}_l)$, and the law of total variance, we have $\mathrm{Var}[\mathbf{x}_{l+1}] = \mathbb{E}[\mathrm{Var}[F(\mathbf{x}_l)|\mathbf{x}_l]] + \mathrm{Var}(\mathbf{x}_l)$. Resnet structure prevents $\mathbf{x}_l$ from vanishing by forcing the variance to grow with depth, i.e. $\mathrm{Var}[\mathbf{x}_l] < \mathrm{Var}[\mathbf{x}_{l+1}]$ if $\mathbb{E}[\mathrm{Var}[F(\mathbf{x}_l)|\mathbf{x}_l]] > 0$. Yet, combined with initialization methods such as [He et al.,

2015], the output variance of each residual branch $\mathrm{Var}[F_l(\mathbf{x}_l)|\mathbf{x}_l]$ will be about the same as its input variance $\mathrm{Var}[\mathbf{x}_l]$, and thus $\mathrm{Var}[\mathbf{x}_{l+1}] \approx 2\mathrm{Var}[\mathbf{x}_l]$. This causes the output variance to explode exponentially with depth without normalization

$$\mathrm{Var}[\mathbf{x}_l] = \mathrm{Var}[\mathbf{x}_0] + \sum_{i=0}^{l-1} \mathrm{Var}[\mathbf{x}_i] \mathbb{E}\left[\mathrm{Var}\left[F_i\left(\frac{\mathbf{x}_i}{\sqrt{\mathrm{Var}[\mathbf{x}_i]}}\right)\middle|\mathbf{x}_i\right]\right] = \Omega(2^l) \tag{6.2}$$

for positively homogeneous blocks (see Definition 6.1). This is detrimental to learning because it can in turn cause gradient explosion.

As we will show, at initialization, the gradient norm of certain activations and weight tensors is *lower bounded* by the cross-entropy loss up to some constant. Intuitively, this implies that blowup in the logits will cause gradient explosion. Our result applies to convolutional and linear weights in a neural network with ReLU nonlinearity (e.g. feed-forward network, CNN), possibly with skip connections (e.g. ResNet, DenseNet), but without any normalization.

Our analysis utilizes properties of positively homogeneous functions, which we now introduce.

Definition 6.1 (positively homogeneous function of first degree)**.** A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is called *positively homogeneous (of first degree)* (p.h.) if for any input $\mathbf{x} \in \mathbb{R}^m$ and $\alpha > 0$, $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$.

Definition 6.2 (positively homogeneous set of first degree)**.** Let $\theta = \{\theta_i\}_{i\in S}$ be the set of parameters of $f(\mathbf{x})$ and $\theta_{ph} = \{\theta_i\}_{i\in S_{ph}\subset S}$. We call θ_{ph} a *positively homogeneous set (of first degree)* (p.h. set) if for any $\alpha > 0$, $f(\mathbf{x}; \theta \setminus \theta_{ph}, \alpha\theta_{ph}) = \alpha f(\mathbf{x}; \theta \setminus \theta_{ph}, \theta_{ph})$, where $\alpha\theta_{ph}$ denotes $\{\alpha\theta_i\}_{i\in S_{ph}}$.

Intuitively, a p.h. set is a set of parameters θ_{ph} in function f such that for any fixed input $\mathbf{x}$ and fixed parameters $\theta\setminus\theta_{ph}$, $\bar{f}(\theta_{ph}) \triangleq f(\mathbf{x}; \theta\setminus\theta_{ph}, \theta_{ph})$ is a p.h. function.

Examples of p.h. functions are ubiquitous in neural networks, including various kinds of linear operations without bias (fully-connected (FC) and convolution layers, pooling, addition, concatenation and dropout etc.) as well as ReLU nonlinearity. Moreover, we have the following claim:

Proposition 6.1. *A function that is the composition of p.h. functions is itself p.h.*

We study classification problems with c classes and the cross-entropy loss. We use f to denote a neural network function except for the softmax layer. Cross-entropy loss is defined as $\ell(\mathbf{z}, \mathbf{y}) \triangleq -\mathbf{y}^T(\mathbf{z} - \texttt{logsumexp}(\mathbf{z}))$ where $\mathbf{y}$ is the one-hot label vector, $\mathbf{z} \triangleq f(\mathbf{x}) \in \mathbb{R}^c$ is the logits where z_i denotes its i-th element, and $\texttt{logsumexp}(\mathbf{z}) \triangleq \log\left(\sum_{i\in[c]} \exp(z_i)\right)$. Consider a minibatch of training examples $\mathcal{D}_M = \{(\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}_{m=1}^M$ and the average cross-entropy loss $\ell_{\text{avg}}(\mathcal{D}_M) \triangleq \frac{1}{M}\sum_{m=1}^M \ell(f(\mathbf{x}^{(m)}), \mathbf{y}^{(m)})$, where we use $^{(m)}$ to index quantities referring to the m-th example. $\|\cdot\|$ denotes any valid norm. We only make the following assumptions about the network f:

1. f is a sequential composition of network blocks $\{f_i\}_{i=1}^L$, i.e. $f(\mathbf{x}_0) = f_L(f_{L-1}(\dots f_1(\mathbf{x}_0)))$, each of which is composed of p.h. functions.
2. Weight elements in the FC layer are i.i.d. sampled from a zero-mean symmetric distribution.

These assumptions hold at initialization if we remove all the normalization layers in a residual network with ReLU nonlinearity, assuming all the biases are initialized at 0.

Our results are summarized in the following two theorems, whose proofs are listed in the appendix:

Theorem 6.1. *Denote the input to the i-th block by $\mathbf{x}_{i-1}$. With Assumption 1, we have*

$$\left\|\frac{\partial \ell}{\partial \mathbf{x}_{i-1}}\right\| \geq \frac{\ell(\mathbf{z}, \mathbf{y}) - H(\mathbf{p})}{\|\mathbf{x}_{i-1}\|}, \tag{6.3}$$

where $\mathbf{p}$ is the softmax probabilities and H denotes the Shannon entropy.

Since $H(\mathbf{p})$ is upper bounded by $\log(c)$ and $\|\mathbf{x}_{i-1}\|$ is small in the lower blocks, blowup in the loss will cause large gradient norm with respect to the lower block input. Our second theorem proves a lower bound on the gradient norm of a p.h. set in a network.

Theorem 6.2. *With Assumption 1, we have*

$$\left\|\frac{\partial \ell_{\text{avg}}}{\partial \theta_{ph}}\right\| \geq \frac{1}{M\|\theta_{ph}\|}\sum_{m=1}^M \ell(\mathbf{z}^{(m)}, \mathbf{y}^{(m)}) - H(\mathbf{p}^{(m)}) \triangleq G(\theta_{ph}). \tag{6.4}$$

Furthermore, with Assumptions 1 and 2, we have

$$\mathbb{E}G(\theta_{ph}) \geq \frac{\mathbb{E}[\max_{i\in[c]} z_i] - \log(c)}{\|\theta_{ph}\|}. \tag{6.5}$$

It remains to identify such p.h. sets in a neural network. In Figure 6-2 we provide three examples of p.h. sets in a ResNet without normalization. Theorem 6.2 suggests that these layers would suffer from the exploding gradient problem, if the logits $\mathbf{z}$ blow up at initialization, which unfortunately would occur in a ResNet without normalization if initialized in a traditional way. This motivates us to introduce a new initialization in the next section.

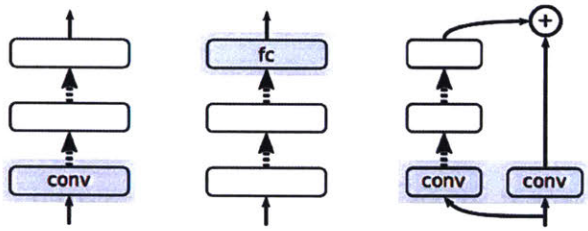


Figure 6-2: Examples of p.h. sets in a ResNet without normalization: (1) the first convolution layer before max pooling; (2) the fully connected layer before softmax; (3) the union of a spatial downsampling layer in the backbone and a convolution layer in its corresponding residual branch.

6.3 ZeroInit: Update a Residual Network $\Theta(\eta)$ per SGD Step

Our analysis in the previous section points out the failure mode of standard initializations for training deep residual network: the gradient norm of certain layers is in expectation lower bounded by a quantity that increases indefinitely with the network depth. However, escaping this failure mode does not necessarily lead us to successful training – after all, it is *the whole network as a function* that we care about, rather than a layer or a network block. In this section, we propose a top-down design of a new initialization that ensures proper update scale to the net-

work function, by simply rescaling a standard initialization. To start, we denote the learning rate by η and set our goal:

$f(\mathbf{x};\theta)$ is updated by $\Theta(\eta)$ per SGD step after initialization as $\eta \to 0$.
That is, $\|\Delta f(\mathbf{x})\| = \Theta(\eta)$ where $\Delta f(\mathbf{x}) \triangleq f(\mathbf{x};\theta - \eta\frac{\partial}{\partial\theta}\ell(f(\mathbf{x}),\mathbf{y})) - f(\mathbf{x};\theta)$.

Put another way, our goal is to design an initialization such that SGD updates to the network function are in the right scale and independent of the depth.

We define the *Shortcut* as the shortest path from input to output in a residual network. The Shortcut is typically a shallow network with a few trainable layers.[1] We assume the Shortcut is initialized using a standard method, and focus on the initialization of the residual branches.

Residual branches update the network in sync. To start, we first make an important observation that the SGD update to each residual branch changes the network function in highly correlated directions. This implies that if a residual network has L residual branches, then an SGD step to each residual branch should change the network function by $\Theta(\eta/L)$ on average to achieve an overall $\Theta(\eta)$ update. We defer the formal statement and its proof until Section 6.7.2.1.

Study of a scalar branch. Next we study how to initialize a residual branch with m layers so that its SGD update changes the network function by $\Theta(\eta/L)$. We assume m is a small positive integer (e.g. 2 or 3). As we are only concerned about the scale of the update, it is sufficiently instructive to study the scalar case, i.e. $F(x) = (\prod_{i=1}^{m} a_i)\, x$ where $a_1, \ldots, a_m, x \in \mathbb{R}^+$. For example, the standard initialization methods typically initialize each layer so that the output (after nonlinear activation) preserves the input variance, which can be modeled as setting $\forall i \in [m], a_i = 1$. In turn, setting a_i to a positive number other than 1 corresponds to rescaling the i-th layer by a_i.

[1] For example, in the ResNet architecture (e.g. ResNet-50, ResNet-101 or ResNet-152) for ImageNet classification, the Shortcut is always a 6-layer network with five convolution layers and one fully-connected layer, irrespective of the total depth of the whole network.

Through deriving the constraints for $F(x)$ to make $\Theta(\eta/L)$ updates, we will also discover how to rescale the weight layers of a standard initialization as desired. In particular, we show the SGD update to $F(x)$ is $\Theta(\eta/L)$ *if and only if* the initialization satisfies the following constraint:

$$\left(\prod_{i\in[m]\setminus\{j\}} a_i\right) x = \Theta\left(\frac{1}{\sqrt{L}}\right), \quad \text{where} \quad j \in \arg\min_k a_k \tag{6.6}$$

We defer the derivation until Section 6.7.2.2.

Equation (6.6) suggests new methods to initialize a residual branch through *rescaling the standard initialization of i-th layer in a residual branch by its corresponding scalar* a_i. For example, we could set $\forall i \in [m], a_i = L^{-\frac{1}{2m-2}}$. Alternatively, we could start the residual branch as a zero function by setting $a_m = 0$ and $\forall i \in [m-1], a_i = L^{-\frac{1}{2m-2}}$. In the second option, the residual branch does not need to "unlearn" its potentially bad random initial state, which can be beneficial for learning. Therefore, we use the latter option in our experiments, unless otherwise specified.

The effects of biases and multipliers. With proper rescaling of the weights in all the residual branches, a residual network is supposed to be updated by $\Theta(\eta)$ per SGD step – our goal is achieved. However, in order to match the training performance of a corresponding network with normalization, there are two more things to consider: biases and multipliers.

Using biases in the linear and convolution layers is a common practice. In normalization methods, bias and scale parameters are typically used to restore the representation power after normalization.[2] Intuitively, because the preferred input/output mean of a weight layer may be different from the preferred output/input mean of an activation layer, it also helps to insert bias terms in a residual network without normalization. Empirically, we find that inserting just one scalar bias before each weight layer and nonlinear activation layer significantly improves the

[2]For example, in batch normalization gamma and beta parameters are used to affine-transform the normalized activations per each channel.

training performance.

Multipliers scale the output of a residual branch, similar to the scale parameters in batch normalization. They have an interesting effect on the learning dynamics of weight layers in the same branch. Specifically, as the stochastic gradient of a layer is typically almost orthogonal to its weight, learning rate decay tends to cause the weight norm equilibrium to shrink when combined with L2 weight decay [van Laarhoven, 2017]. In a branch with multipliers, this in turn causes the growth of the multipliers, increasing the effective learning rate of other layers. In particular, we observe that inserting just one scalar multiplier per residual branch mimics the weight norm dynamics of a network with normalization, and spares us the search of a new learning rate schedule.

Put together, we propose the following method to train residual networks without normalization:

ZeroInit (or: How to train a deep residual network without normalization)

1. Initialize the classification layer and the last layer of each residual branch to 0.

2. Initialize every other layer using a standard method, e.g. He et al. [2015], and scale only the weight layers inside residual branches by $L^{-\frac{1}{2m-2}}$.

3. Add a scalar multiplier (initialized at 1) in every branch and a scalar bias (initialized at 0) before each convolution, linear, and element-wise activation layer.

It is important to note that Rule 2 of ZeroInit is the essential part as predicted by Equation (6.6). Indeed, we observe that using Rule 2 alone is sufficient and necessary for training extremely deep residual networks. On the other hand, Rule 1 and Rule 3 make further improvements for training so as to match the performance of a residual network with normalization layers, as we explain in the above text.[3] We

[3]It is worth noting that the design of ZeroInit is a simplification of the common practice, in that we only introduce $O(K)$ parameters beyond convolution and linear weights (since we remove bias terms from convolution and linear layers), whereas the common practice includes $O(KC)$

find ablation experiments confirm our claims (see Section 5.3.8).

6.4 Experiments

6.4.1 Training at increasing depth

One of the key advatanges of BatchNorm is that it leads to fast training even for very deep models [Ioffe and Szegedy, 2015]. Here we will determine if we can match this desirable property by relying only on proper initialization. We propose to evaluate how each method affects training very deep nets by *measuring the test accuracy after the first epoch as we increase depth.* In particular, we use the wide residual network (WRN) architecture with width 1 and the default weight decay $5e{-}4$ [Zagoruyko and Komodakis, 2016b]. We specifically use the default learning rate of 0.1 because the ability to use high learning rates is considered to be important to the success of BatchNorm. We compare ZeroInit against three baseline methods – (1) rescale the output of each residual block by $\frac{1}{\sqrt{2}}$ [Balduzzi et al., 2017], (2) post-process an orthogonal initialization such that the output variance of each residual block is close to 1 (Layer-sequential unit-variance orthogonal initialization, or LSUV) [Mishkin and Matas, 2015], (3) batch normalization [Ioffe and Szegedy, 2015]. We use the default batch size of 128 up to 1000 layers, with a batch size of 64 for 10,000 layers. We limit our budget of epochs to 1 due to the computational strain of evaluating models with up to 10,000 layers.

Figure 6-3 shows the test accuracy at the first epoch as depth increases. Observe that ZeroInit matches or exceeds the performance of BatchNorm at the first epoch, even with 10,000 layers. LSUV and $\sqrt{1/2}$-scaling are not able to train with the same learning rate as BatchNorm past 100 layers.

[Ioffe and Szegedy, 2015, Salimans and Kingma, 2016] or $O(KCWH)$ [Ba et al., 2016] additional parameters, where K is the number of layers, C is the max number of channels per layer and W, H are the spatial dimension of the largest feature maps.

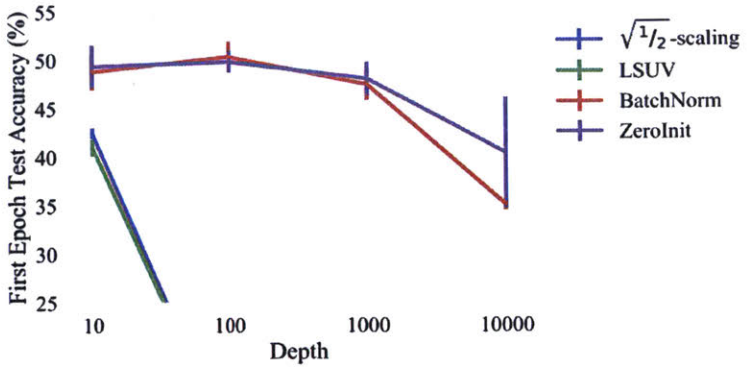


Figure 6-3: Depth of residual networks versus test accuracy at the first epoch for various methods on CIFAR-10 with the default BatchNorm learning rate. We observe that ZeroInit is able to train very deep networks with the same learning rate as batch normalization. (Higher is better.)

6.4.2 Image classification

In this section, we evaluate the ability of ZeroInit to replace batch normalization in image classification applications. On the CIFAR-10 dataset, we first test on ResNet-110 [He et al., 2016a] with default hyper-parameters; results are shown in Table 6.1. ZeroInit obtains 7% relative improvement in test error compared with standard initialization; however, we note a substantial difference in the difficulty of training. While network with ZeroInit is trained with the same learning rate and converge as fast as network with batch normalization, we fail to train a Xavier initialized ResNet-110 with 0.1x maximal learning rate.[4] The test error gap in Table 6.1 is likely due to the regularization effect of BatchNorm rather than difficulty in optimization; when we train ZeroInit networks with better regularization, the test error gap disappears and we obtain state-of-the-art results on CIFAR-10 and SVHN without normalization layers (see Section 6.7.3.2).

On the ImageNet dataset, we benchmark ZeroInit with the ResNet-50 and ResNet-101 architectures [He et al., 2016a], trained for 100 epochs and 200 epochs respectively. Similar to our finding on the CIFAR-10 dataset, we observe that (1) training with ZeroInit is fast and stable with the default hyperparameters, (2) ZeroInit alone significantly improves the test error of standard initialization, and (3) there

[4]Personal communication with the authors of [Shang et al., 2017] confirms our observation, and reveals that the Xavier initialized network need more epochs to converge.

Dataset	ResNet-110	Normalization	Large η	Test Error (%)
CIFAR-10	w/ BatchNorm [He et al., 2016a]	✓	✓	6.61
	w/ Xavier Init [Shang et al., 2017]	✗	✗	7.78
	w/ ZeroInit	✗	✓	7.24

Table 6.1: Results on CIFAR-10 with ResNet-110 (mean/median of 5 runs; lower is better).

is a large test error gap between ZeroInit and BatchNorm. Further inspection reveals that ZeroInit models obtain significantly lower training error compared with BatchNorm models (see Section 6.7.3.3), i.e. ZeroInit suffers from overfitting. We therefore apply stronger regularization to the ZeroInit models using Mixup [Zhang et al., 2017]. We find it is beneficial to reduce the learning rate of the scalar multiplier and bias by 10x when additional large regularization is used. Best Mixup coefficients are found through cross-validation: they are 0.2, 0.1 and 0.7 for BatchNorm, GroupNorm [Wu and He, 2018] and ZeroInit respectively. We present the results in Table 6.2, noting that with better regularization, the performance of ZeroInit is on par with GroupNorm.

Model	Method	Normalization	Test Error (%)
ResNet-50	BatchNorm [Goyal et al., 2017]	✓	23.6
	BatchNorm + Mixup [Zhang et al., 2017]		**23.3**
	GroupNorm + Mixup		23.9
	Xavier Init [Shang et al., 2017]	✗	31.5
	ZeroInit		27.6
	ZeroInit + Mixup		**24.0**
ResNet-101	BatchNorm [Zhang et al., 2017]	✓	22.0
	BatchNorm + Mixup [Zhang et al., 2017]		**20.8**
	GroupNorm + Mixup		21.4
	ZeroInit + Mixup	✗	21.4

Table 6.2: ImageNet test results using the ResNet architecture. (Lower is better.)

6.4.3 Machine translation

To demonstrate the generality of ZeroInit, we also apply it to replace layer normalization [Ba et al., 2016] in Transformer [Vaswani et al., 2017], a state-of-the-art

neural network for machine translation. Specifically, we use the fairseq library [Gehring et al., 2017] and follow the ZeroInit template in Section 6.3 to modify the baseline model. We evaluate on two standard machine translation datasets, IWSLT German-English (de-en) and WMT English-German (en-de) following the setup of Ott et al. [2018]. For the IWSLT de-en dataset, we cross-validate the dropout probability from $\{0.3, 0.4, 0.5, 0.6\}$ and find 0.5 to be optimal for both ZeroInit and the LayerNorm baseline. For the WMT'16 en-de dataset, we use dropout probability 0.4. All models are trained for 200k updates.

It was reported [Chen et al., 2018] that "Layer normalization is most critical to stabilize the training process... removing layer normalization results in unstable training runs". However we find training with ZeroInit to be very stable and as fast as the baseline model. Results are shown in Table 6.3. Surprisingly, we find the models do not suffer from overfitting when LayerNorm is replaced by ZeroInit, thanks to the strong regularization effect of dropout. Instead, ZeroInit matches or supersedes the state-of-the-art results using Transformer model on both datasets.

Dataset	Model	Normalization	BLEU
IWSLT DE-EN	[Deng et al., 2018]	✓	33.1
	LayerNorm		34.2
	ZeroInit	✗	**34.5**
WMT EN-DE	[Vaswani et al., 2017]	✓	28.4
	LayerNorm [Ott et al., 2018]		**29.3**
	ZeroInit	✗	**29.3**

Table 6.3: Comparing ZeroInit vs. LayerNorm for machine translation tasks. (Higher is better.)

6.5 Related Work

Normalization methods. Normalization methods have enabled training very deep residual networks, and are currently an essential building block of the most successful deep learning architectures. All normalization methods for training neu-

ral networks explicitly normalize (i.e. standardize) some component (activations or weights) through dividing activations or weights by some real number computed from its statistics and/or subtracting some real number activation statistics (typically the mean) from the activations.[5] In contrast, ZeroInit does not compute statistics (mean, variance or norm) at initialization or during any phase of training, hence is not a normalization method.

Theoretical analysis of residual networks. Training very deep neural networks is an important theoretical problem. Early works study the propagation of variance in the forward and backward pass for different activation functions [Glorot and Bengio, 2010, He et al., 2015]. Recently, the study of *dynamical isometry* [Saxe et al., 2013] provides a more detailed characterization of the forward and backward signal propogation at initialization [Hanin, 2018, Pennington et al., 2017], enabling training 10,000-layer CNNs from scratch [Xiao et al., 2018]. For residual networks, activation scale [Hanin and Rolnick, 2018], gradient variance [Balduzzi et al., 2017] and dynamical isometry property [Yang and Schoenholz, 2017] have been studied. Our analysis in Section 6.2 leads to the similar conclusion as previous work that the standard initialization for residual networks is problematic. However, our use of positive homogeneity for lower bounding the gradient norm of a neural network is novel, and applies to a broad class of neural network architectures (e.g. ResNet, DenseNet) and initialization methods (e.g. Xavier, LSUV) with a minimal set of assumptions and a simple proof.

Understanding batch normalization. Despite its popularity in practice, batch normalization has not been well understood. Ioffe and Szegedy [2015] attributed its success to "reducing internal covariate shift", whereas Santurkar et al. [2018] argued that its effect may be "smoothing loss surface". Our analysis in Section 6.2 corroborates the latter idea by showing that standard initialization leads to very steep loss surface at initialization; however, Section 6.3 and various experiments

[5]For reference, we include a brief history of normalization methods in Section 6.7.4.

suggests that steep loss surface may be an artifact of improper initialization, at least for residual networks. Hoffer et al. [2018], van Laarhoven [2017] studied the effect of (batch) normalization and weight decay on the effective learning rate. Their results inspire us to include a multiplier in each residual branch.

ResNet initialization in practice. Balduzzi et al. [2017], Gehring et al. [2017] proposed to address the initialization problem of residual nets by using the recurrence $\mathbf{x}_l = \sqrt{1/2}(\mathbf{x}_{l-1} + F_l(\mathbf{x}_{l-1}))$. Mishkin and Matas [2015] proposed a data-dependent initialization to mimic the effect of batch normalization in the first forward pass. While both methods limit the scale of activation and gradient, they would fail to train stably at the maximal learning rate for very deep residual networks, since they fail to consider the accumulation of highly correlated updates contributed by different residual branches to the network function (Section 6.7.2.1). Goyal et al. [2017], Hardt and Ma [2016], Kingma and Dhariwal [2018], Srivastava et al. [2015] found that initializing the residual branches at (or close to) zero helped optimization. Our results support their observation in general, but Equation (6.6) suggests additional subtleties when choosing a good initialization scheme.

6.6 Conclusion

In this work, we study how to train a deep residual network reliably without normalization. Our theory in Section 6.2 suggests that the exploding gradient problem at initialization in a positively homogeneous network such as ResNet is directly linked to the blowup of logits. In Section 6.3 we develop ZeroInit to ensure the whole network as well as each residual branch gets updates of proper scale, based on a top-down analysis. Extensive experiments on real world datasets demonstrate that ZeroInit matches normalization techniques in training deep residual networks, and achieves state-of-the-art test performance with proper regularization.

Our work opens up new possibilities for both theory and applications. On the

theory side, removing the normalization layers is supposed to simplify the analysis of these residual networks. Our empirical results suggest that some previous hypotheses [e.g. Ioffe and Szegedy, 2015, Santurkar et al., 2018] about the effects of (batch) normalization may need to be revised. It would also be interesting to understand the regularization benefits of various normalization methods. On the application side, it may be possible to develop better regularization methods, which, when combined with ZeroInit, yield further improvements.

6.7 Appendix

6.7.1 Proofs for Section 6.2

6.7.1.1 Gradient norm lower bound for the input to a network block

Proof of Theorem 6.1. We use $f_{i\to j}$ to denote the composition $f_j \circ f_{j-1} \circ \cdots \circ f_i$, so that $\mathbf{z} = f_{i\to L}(\mathbf{x}_{i-1})$ for all $i \in [L]$. Note that $\mathbf{z}$ is p.h. with respect to the input of each network block, i.e. $f_{i\to L}((1+\epsilon)\mathbf{x}_{i-1}) = (1+\epsilon)f_{i\to L}(\mathbf{x}_{i-1})$ for $\epsilon > -1$. This allows us to compute the gradient of the cross-entropy loss with respect to the scaling factor ϵ at $\epsilon = 0$ as

$$\frac{\partial}{\partial \epsilon}\ell(f_{i\to L}((1+\epsilon)\mathbf{x}_{i-1}), \mathbf{y})\bigg|_{\epsilon=0} = \frac{\partial \ell}{\partial \mathbf{z}}\frac{\partial f_{i\to L}}{\partial \epsilon} = -\mathbf{y}^T\mathbf{z} + \mathbf{p}^T\mathbf{z} = \ell(\mathbf{z}, \mathbf{y}) - H(\mathbf{p}) \quad (6.7)$$

Since the gradient L_2 norm $\|{}^{\partial \ell}/_{\partial \mathbf{x}_{i-1}}\|$ must be greater than the directional derivative $\frac{\partial}{\partial t}\ell(f_{i\to L}(\mathbf{x}_{i-1} + t\frac{\mathbf{x}_{i-1}}{\|\mathbf{x}_{i-1}\|}), \mathbf{y})$, defining $\epsilon = {}^{t}/_{\|\mathbf{x}_{i-1}\|}$ we have

$$\left\|\frac{\partial \ell}{\partial \mathbf{x}_{i-1}}\right\| \geq \frac{\partial}{\partial \epsilon}\ell(f_{i\to L}(\mathbf{x}_{i-1} + \epsilon\mathbf{x}_{i-1}), \mathbf{y})\frac{\partial \epsilon}{\partial t} = \frac{\ell(\mathbf{z}, \mathbf{y}) - H(\mathbf{p})}{\|\mathbf{x}_{i-1}\|}. \quad (6.8)$$

□

6.7.1.2 Gradient norm lower bound for positively homogeneous sets

Proof of Theorem 6.2. The proof idea is similar. Recall that if θ_{ph} is a p.h. set, then $\bar{f}^{(m)}(\theta_{ph}) \triangleq f(\mathbf{x}^{(m)}; \theta \setminus \theta_{ph}, \theta_{ph})$ is a p.h. function. We therefore have

$$\frac{\partial}{\partial \epsilon} \ell_{\text{avg}}(\mathcal{D}_M; (1+\epsilon)\theta_{ph})\Big|_{\epsilon=0} = \frac{1}{M}\sum_{m=1}^{M} \frac{\partial \ell}{\partial \mathbf{z}^{(m)}} \frac{\partial \bar{f}^{(m)}}{\partial \epsilon} = \frac{1}{M}\sum_{m=1}^{M} \ell(\mathbf{z}^{(m)}, \mathbf{y}^{(m)}) - H(\mathbf{p}^{(m)}) \tag{6.9}$$

hence we again invoke the directional derivative argument to show

$$\left\| \frac{\partial \ell_{\text{avg}}}{\partial \theta_{ph}} \right\| \geq \frac{1}{M\|\theta_{ph}\|} \sum_{m=1}^{M} \ell(\mathbf{z}^{(m)}, \mathbf{y}^{(m)}) - H(\mathbf{p}^{(m)}) \triangleq G(\theta_{ph}). \tag{6.10}$$

In order to estimate the scale of this lower bound, recall the FC layer weights are i.i.d. sampled from a symmetric, mean-zero distribution, therefore $\mathbf{z}$ has a symmetric probability density function with mean $\mathbf{0}$. We hence have

$$\mathbb{E}\ell(\mathbf{z}, \mathbf{y}) = \mathbb{E}[-\mathbf{y}^T(\mathbf{z} - \texttt{logsumexp}(\mathbf{z}))] \geq \mathbb{E}[\mathbf{y}^T(\max_{i\in[c]} z_i - \mathbf{z})] = \mathbb{E}[\max_{i\in[c]} z_i] \tag{6.11}$$

where the inequality uses the fact that $\texttt{logsumexp}(\mathbf{z}) \geq \max_{i\in[c]} z_i$; the last equality is due to $\mathbf{y}$ and $\mathbf{z}$ being independent at initialization and $\mathbb{E}\mathbf{z} = \mathbf{0}$. Using the trivial bound $\mathbb{E}H(\mathbf{p}) \leq \log(c)$, we get

$$\mathbb{E}G(\theta_{ph}) \geq \frac{\mathbb{E}[\max_{i\in[c]} z_i] - \log(c)}{\|\theta_{ph}\|} \tag{6.12}$$

which shows that the gradient norm of a p.h. set is of the order $\Omega(\mathbb{E}[\max_{i\in[c]} z_i])$ at initialization. □

6.7.2 Proofs for Section 6.3

6.7.2.1 Residual branches update the network in sync

A common theme in previous analysis of residual networks is the scale of activation and gradient [Balduzzi et al., 2017, Hanin and Rolnick, 2018, Yang and Schoenholz, 2017]. However, it is more important to consider the scale of actual change to the network function made by a (stochastic) gradient descent step. If the updates to different layers cancel out each other, the network would be stable as a whole

despite drastic changes in different layers; if, on the other hand, the updates to different layers align with each other, the whole network may incur a drastic change in one step, even if each layer only changes a tiny amount. We now provide analysis showing that the latter scenario more accurately describes what happens in reality at initialization.

For our result in this section, we make the following assumptions:

- f is a sequential composition of network blocks $\{f_i\}_{i=1}^{L}$, i.e. $f(\mathbf{x}_0) = f_L(f_{L-1}(\dots f_1(\mathbf{x}_0)))$, consisting of fully-connected weight layers, ReLU activation functions and residual branches.
- f_L is a fully-connected layer with weights i.i.d. sampled from a zero-mean distribution.
- There is no bias parameter in f.

For $l < L$, let $\mathbf{x}_{l-1}$ be the input to f_l and $F_l(\mathbf{x}_{l-1})$ be a branch in f_l with m_l layers. Without loss of generality, we study the following specific form of network architecture:

$$\begin{aligned} F_l(\mathbf{x}_{l-1}) &= (\overbrace{\mathrm{ReLU} \circ W_l^{(m_l)} \circ \cdots \circ \mathrm{ReLU} \circ W_l^{(1)}}^{m_l \text{ ReLU}})(\mathbf{x}_{l-1}), \\ f_l(\mathbf{x}_{l-1}) &= \mathbf{x}_{l-1} + F_l(\mathbf{x}_{l-1}). \end{aligned}$$

For the last block we denote $m_L = 1$ and $f_L(\mathbf{x}_{L-1}) = F_L(\mathbf{x}_{L-1}) = W_L^{(1)}\mathbf{x}_{L-1}$.

Furthermore, we always choose 0 as the gradient of ReLU when its input is 0. As such, with input $\mathbf{x}$, the output and gradient of $\mathrm{ReLU}(\mathbf{x})$ can be simply written as $D_{\mathbb{1}[\mathbf{x}>0]}\mathbf{x}$, where $D_{\mathbb{1}[\mathbf{x}>0]}$ is a diagonal matrix with diagonal entries corresponding to $\mathbb{1}[\mathbf{x} > 0]$. Denote the preactivation of the i-th layer (i.e. the input to the i-th ReLU) in the l-th block by $\mathbf{x}_l^{(i)}$. We define the following terms to simplify our presentation:

$$\begin{aligned} F_l^{(i-)} &\triangleq D_{\mathbb{1}[\mathbf{x}_l^{(i-1)}>0]} W_l^{(i-1)} \cdots D_{\mathbb{1}[\mathbf{x}_l^{(1)}>0]} W_l^{(1)} \mathbf{x}_{l-1}, \quad l < L, i \in [m_l] \\ F_l^{(i+)} &\triangleq D_{\mathbb{1}[\mathbf{x}_l^{(m_l)}>0]} W_l^{(m_l)} \cdots D_{\mathbb{1}[\mathbf{x}_l^{(i)}>0]}, \quad l < L, i \in [m_l] \\ F_L^{(1-)} &\triangleq \mathbf{x}_{L-1} \\ F_L^{(1+)} &\triangleq I \end{aligned}$$

We have the following result on the gradient update to f:

Theorem 6.3. *With the above assumptions, suppose we update the network parameters by* $\Delta\theta = -\eta \frac{\partial}{\partial \theta} \ell(f(\mathbf{x}_0;\theta), \mathbf{y})$, *then the update to network output* $\Delta f(\mathbf{x}_0) \triangleq f(\mathbf{x}_0; \theta + \Delta\theta) - f(\mathbf{x}_0;\theta)$ *is*

$$\Delta f(\mathbf{x}_0) = -\eta \sum_{l=1}^{L} \left[\sum_{i=1}^{m_l} \overbrace{\left\| F_l^{(i-)} \right\|^2 \left(\frac{\partial f}{\partial \mathbf{x}_l} \right)^T F_l^{(i+)} \left(F_l^{(i+)} \right)^T \left(\frac{\partial f}{\partial \mathbf{x}_l} \right)}^{\triangleq J_l^i} \right] \frac{\partial \ell}{\partial \mathbf{z}} + O(\eta^2), \quad (6.13)$$

where $\mathbf{z} \triangleq f(\mathbf{x}_0) \in \mathbb{R}^c$ *is the logits.*

Let us discuss the implecation of this result before delving into the proof. As each J_l^i is a $c \times c$ real symmetric positive semi-definite matrix, the trace norm of each J_l^i equals its trace. Similarly, the trace norm of $J \triangleq \sum_l \sum_i J_l^i$ equals the trace of the sum of all J_l^i as well, which scales linearly with the number of residual branches L. Since the output $\mathbf{z}$ has no (or little) correlation with the target $\mathbf{y}$ at the start of training, $\frac{\partial \ell}{\partial \mathbf{z}}$ is a vector of some random direction. It then follows that the expected update scale is proportional to the trace norm of J, which is proportional to L as well as the average trace of J_l^i. Simply put, to allow the whole network be updated by $\Theta(\eta)$ per step independent of depth, we need to ensure each residual branch contributes only a $\Theta(\eta/L)$ update on average.

Proof. The first insight to prove our result is to note that conditioning on a specific input $\mathbf{x}_0$, we can replace each ReLU activation layer by a diagonal matrix and does not change the forward and backward pass. (In fact, this is valid even after we apply a gradient descent update, as long as the learning rate $\eta > 0$ is sufficiently small so that all positive preactivation remains positive. This observation will be essential for our later analysis.) We thus have the gradient w.r.t. the i-th weight layer in the l-th block is

$$\frac{\partial \ell}{\partial \mathrm{vec}(W_l^{(i)})} = \frac{\partial \mathbf{x}_l}{\partial \mathrm{vec}(W_l^{(i)})} \cdot \frac{\partial f}{\partial \mathbf{x}_l} \cdot \frac{\partial \ell}{\partial \mathbf{z}} = \left(F_l^{(i-)} \otimes I_l^{(i)} \right) \left(F_l^{(i+)} \right)^T \frac{\partial f}{\partial \mathbf{x}_l} \cdot \frac{\partial \ell}{\partial \mathbf{z}}. \quad (6.14)$$

where $\otimes$ denotes the Kronecker product. The second insight is to note that with our assumptions, a network block and its gradient w.r.t. its input have the following

relation:

$$f_l(\mathbf{x}_{l-1}) = \frac{\partial f_l}{\partial \mathbf{x}_{l-1}} \cdot \mathbf{x}_{l-1}. \tag{6.15}$$

We then plug in Equation (6.14) to the gradient update $\Delta\theta = -\eta \frac{\partial}{\partial \theta} \ell(f(\mathbf{x}_0; \theta), \mathbf{y})$, and recalculate the forward pass $f(\mathbf{x}_0; \theta + \Delta\theta)$. The theorem follows by applying Equation (6.15) and a first-order Taylor series expansion in a small neighborhood of $\eta = 0$ where $f(\mathbf{x}_0; \theta + \Delta\theta)$ is smooth w.r.t. η. □

6.7.2.2 What scalar branch has $\Theta(\eta/L)$ updates?

For this section, we focus on the proper initialization of a scalar branch $F(x) = (\prod_{i=1}^{m} a_i)x$. We have the following result:

Theorem 6.4. *Assuming* $\forall i, a_i \geq 0, x = \Theta(1)$ *and* $\frac{\partial \ell}{\partial F(x)} = \Theta(1)$, *then* $\Delta F(x) \triangleq F(x; \theta - \eta \frac{\partial \ell}{\partial \theta}) - F(x; \theta)$ *is* $\Theta(\eta/L)$ if and only if

$$\left(\prod_{k \in [m] \setminus \{j\}} a_k \right) x = \Theta\left(\frac{1}{\sqrt{L}} \right), \quad \textit{where} \quad j \in \arg\min_k a_k \tag{6.16}$$

Proof. We start by calculating the gradient of each parameter:

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial \ell}{\partial F} \left(\prod_{k \in [m] \setminus \{i\}} a_k \right) x \tag{6.17}$$

and a first-order approximation of $\Delta F(x)$:

$$\Delta F(x) = -\eta \frac{\partial \ell}{\partial F(x)} (F(x))^2 \sum_{i=1}^{m} \frac{1}{a_i^2} \tag{6.18}$$

where we conveniently abuse some notations by defining

$$F(x)\frac{1}{a_i} \triangleq \left(\prod_{k \in [m] \setminus \{i\}} a_k \right) x, \quad \text{if } a_i = 0. \tag{6.19}$$

Denote $\sum_{i=1}^{m} \frac{1}{a_i^2}$ as M and $\min_k a_k$ as A, we have

$$(F(x))^2 \cdot \frac{1}{A^2} \leq (F(x))^2 M \leq (F(x))^2 \cdot \frac{m}{A^2} \tag{6.20}$$

and therefore by rearranging Equation (6.18) and letting $\Delta F(x) = \Theta(\eta/L)$ we get

$$(F(x))^2 \cdot \frac{1}{A^2} = \Theta\left(\frac{\Delta F(x)}{\eta \frac{\partial \ell}{\partial F(x)}} \right) = \Theta\left(\frac{1}{L} \right) \tag{6.21}$$

i.e. $F(x)/A = \Theta(1/\sqrt{L})$. Hence the "only if" part is proved. For the "if" part, we apply Equation (6.20) to Equation (6.18) and observe that by Equation (6.16)

$$\Delta F(x) = \Theta\left(\eta(F(x))^2 \cdot \frac{1}{A^2}\right) = \Theta\left(\frac{\eta}{L}\right) \tag{6.22}$$

□

The result of this theorem provides useful guidance on how to rescale the standard initialization to achieve the desired update scale for the network function.

6.7.3 Additional experiments

6.7.3.1 Ablation studies of ZeroInit

In this section we present the training curves of different architecture designs and initialization schemes. Specifically, we compare the training accuracy of batch normalization, ZeroInit, as well as a few ablated options: (1) removing the bias parameters in the network; (2) use 0.1x the suggested initialization scale and no bias parameters; (3) use 10x the suggested initialization scale and no bias parameters; and (4) remove all the residual branches. The results are shown in Figure 6-4. We see that initializing the residual branch layers at a smaller scale (or all zero) slows down learning, whereas training fails when initializing them at a larger scale; we also see the clear benefit of adding bias parameters in the network.

6.7.3.2 CIFAR and SVHN with better regularization

We perform additional experiments to validate our hypothesis that the gap in test error between ZeroInit and batch normalization is primarily due to overfitting. To combat overfitting, we use Mixup [Zhang et al., 2017] and Cutout [DeVries and Taylor, 2017a] with default hyperparameters as additional regularization. On the CIFAR-10 dataset, we perform experiments with WideResNet-40-10 and on SVHN we use WideResNet-16-12 [Zagoruyko and Komodakis, 2016b], all with the default hyperparameters. We observe in Table 6.4 that models trained with ZeroInit and

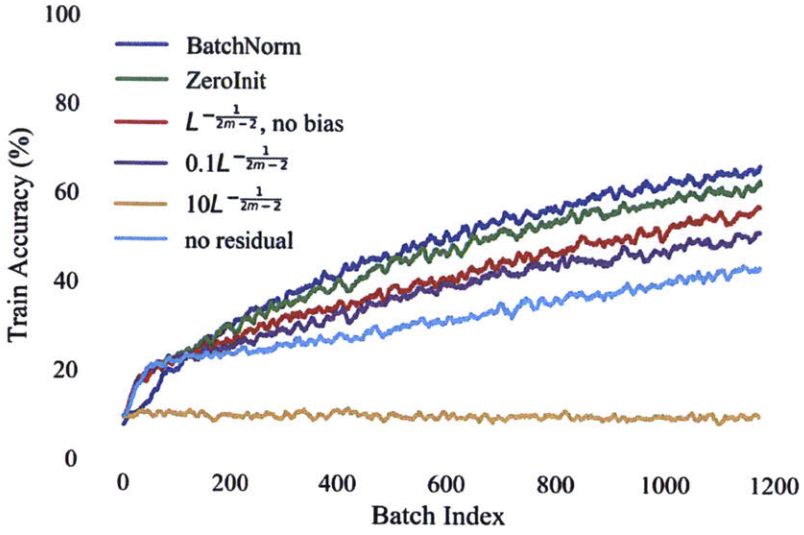


Figure 6-4: Minibatch training accuracy of ResNet-110 on CIFAR-10 dataset with different configurations in the first 3 epochs. We use minibatch size of 128 and smooth the curves using 10-step moving average.

strong regularization are competitive with state-of-the-art methods on CIFAR-10 and SVHN, as well as our baseline with batch normalization.

Dataset	Model	Normalization	Test Error (%)
CIFAR-10	[Zagoruyko and Komodakis, 2016b]	Yes	3.8
	[Yamada et al., 2018]		**2.3**
	BatchNorm + Mixup + Cutout		2.5
	[Graham, 2014]	No	3.5
	ZeroInit + Mixup + Cutout		**2.3**
SVHN	[Zagoruyko and Komodakis, 2016b]	Yes	1.5
	[DeVries and Taylor, 2017a]		**1.3**
	BatchNorm + Mixup + Cutout		1.4
	[Lee et al., 2016]	No	1.7
	ZeroInit + Mixup + Cutout		**1.4**

Table 6.4: Additional results on CIFAR-10, SVHN datasets.

6.7.3.3 Training and test curves on ImageNet

Figure 6-5 shows that without additional regularization ZeroInit fits the training set very well, but overfits significantly. We see in Figure 6-6 that ZeroInit is com-

petitive with networks trained with normalization when the Mixup regularizer is used.

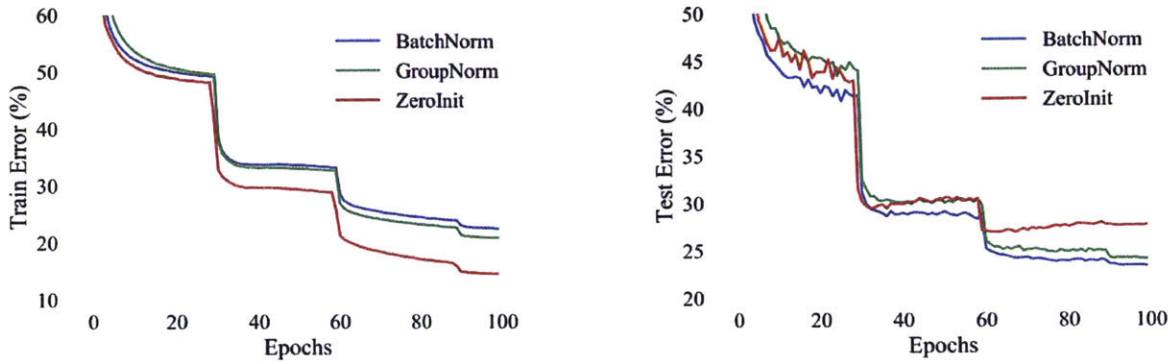


Figure 6-5: Training and test errors on ImageNet using ResNet-50 without additional regularization. We observe that ZeroInit is able to better fit the training data and that leads to overfitting - more regularization is needed. Results of BatchNorm and GroupNorm reproduced from [Wu and He, 2018].

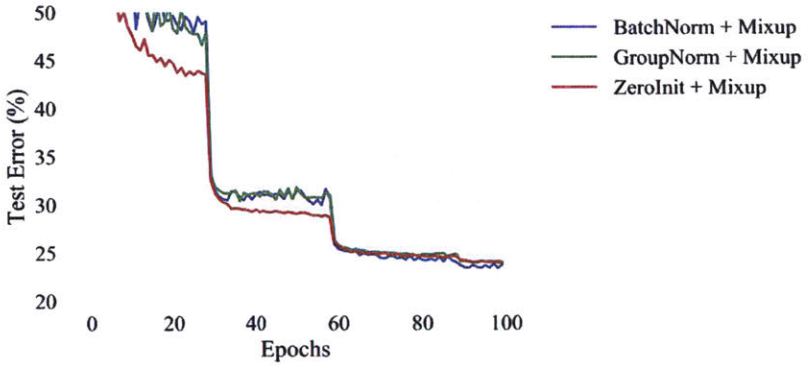


Figure 6-6: Test error of ResNet-50 on ImageNet with Mixup [Zhang et al., 2017]. ZeroInit closely matches the final results yielded by the use of GroupNorm, without any normalization.

6.7.4 Additional references: A brief history of normalization methods

The first use of normalization in neural networks appears in the modeling of biological visual system and dates back at least to Heeger [1992] in neuroscience and to Lyu and Simoncelli [2008], Pinto et al. [2008] in computer vision, where each

neuron output is divided by the sum (or norm) of all of the outputs, a module called divisive normalization. Recent popular normalization methods, such as local response normalization [Krizhevsky et al., 2012], batch normalization [Ioffe and Szegedy, 2015] and layer normalization [Ba et al., 2016] mostly follow this tradition of dividing the neuron activations by their certain summary statistics, often also with the activation mean subtracted. An exception is weight normalization [Salimans and Kingma, 2016], which instead divides the weight parameters by their statistics, specifically the weight norm; weight normalization also adopts the idea of activation normalization for weight initialization. The recently proposed actnorm [Kingma and Dhariwal, 2018] removes the normalization of weight parameters, but still use activation normalization to initialize the affine transformation layers.

7

Conclusions

In the previous chapters, I have presented two lines of my work in the field of non-convex optimization and learning. The first part consists of iteration complexity analysis of first-order Riemannian optimization algorithms, and the second part focuses on important practical issues in deep neural network training. How do these results fit into the global picture of non-convex optimization and learning? What insights do they tell us about machine learning and data science in general? I would like to conclude with some discussions on these questions.

7.1 A Zoomed-Out Summary

Optimization is an instrumental algorithmic tool in virtually every field of applied math and quantitative sciences, yet the difficulty of obtaining a solution can vary

drastically. On the one hand, there exist notoriously hard non-convex problems where even *testing* an approximate solution seems hopeless in the worst case; on the other hand, fruitful theory and algorithmic development has been made for interesting problem structures, such as convexity or submodularity, that allows for efficient solvers even in the worst case.

Why are certain problems easy to solve? Assumptions about the objective function, such as convexity or smoothness, are perhaps the most important factor. However, the role of the linear space assumption is often taken for granted. Is the benign vector space geometry an essential component in the analysis, or is it an irrelevant confounding factor? The results in this thesis support instead the middle ground, that the actual answer may be mixed. Indeed, for most first-order methods, convergence is global in both vector space and Riemannian manifolds, and the corresponding rates differ by at most a constant depending on the manifold curvature; whereas for Nesterov's accelerated gradient methods, convergence analysis seems to be much more difficult under nonlinear metric. This is interesting and perhaps surprising, since it could imply intricate interactions between the geometry of certain manifolds and well-known optimization algorithms. Such observation may also help us gain understanding of vector space optimization — a side-by-side comparison of vector space optimization with the analysis developed in this thesis could serve to put the former theory in perspective.

As neural networks achieve dominant performance across the board, gone are the golden days when convexifying a model suggests halfway to success. In contrast to the convex models, much less is known about the optimization and generalization properties of neural networks. In fact, a lot of the current best practices in deep learning are accumulated results of trial and error. The most notably example of this trend is Google's AutoML project. By systematically searching through the design space of network modules, data augmentations or hyperparameters, automatic machine learning can spare human experts from the tiring task of iteratively refining a design solution. For example, the recently proposed AmoebaNet [Real et al., 2018] was the result of running a new evolutionary algorithm to search

through the combinatorial space of network modules (called cells); the AutoAugment project [Cubuk et al., 2018] found out data augmentation policies for different datasets each consists of 25 sub-policies, each of which is in turn a pair of transformations chosen from about ten primitive data augmentation transformations.

While the promise of automatic machine learning appears attractive, in the long run it might serve to slow down machine learning research progress. The generated best performing models and augmentation policies are often a convoluted mix of various primitives and harder to interpret than manually designed models. It is difficult to attribute the success (or failure) of such automatic design process to what actually matters, as *we can hardly tell what matters from what does not*, except for the empirical observation that a diverse mixture of sub-modules helps to boost test results. Furthermore, these methods do not address the problem of inventing good learning primitives. The best primitives at this time, such as learning rate schedule, batch normalization, and various kinds of convolutions are often discovered through trial and error without fully understanding their effects and effectiveness. What if we could distinguish the "active ingredient" from the "inactive"? Even if AutoML is an effective method in its own right, it is likely that we could miss out many important primitives, unless we invest more efforts into further understanding.

In this thesis, the approach to deep learning follows a different route. Instead of conforming to existing primitives and let a machine figure out their best combinations, I prefer to identify problems and solve them by inventing new primitives. Seeing training set overfitting as a major problem of empirical risk minimization, I came up with mixup as a remedy. While exploring how batch normalization helps optimization, I came up with ZeroInit as an alternative without normalization. Importantly, neither of the two ideas came solely from an exhaustive search of existing hypotheses, a series of laborious experiments, or a relentless theoretical contemplation. Instead, they emerged from a combination of all three activities. *Without serious contemplation, the experimental results could often seem incomprehensible; without an unforgiving attitude towards performance, confounding theories cannot be*

ruled out; finally, something new is learned whenever we have exhausted and rejected all the existing hypotheses. We know much less about deep learning than we should, which means the field is full of opportunities. To me, the best way to moving forward is to take the approach of natural science, where *theories*, *hypotheses* and *experiments* interact and co-evolve with each other under ruthless scrutiny.

7.2 Open Problems

Related to the problems studied in this thesis, some important questions remain unanswered while exciting new problems emerge. Future research may help to elucidate on these topics:

Riemannian optimization. A recurring hurdle of Riemannian optimization analysis seems to be the lack of a sensible duality theory for functions on Riemannian manifolds. While the Fenchel duality argument *approximately* holds when the neighborhood is sufficiently small or the space sufficiently flat, more efforts are required to understand the nonasymptotic setting in practice. A related question is iteration complexity lower bounds of Riemannian optimization problems. The construction of such lower bounds may involve geometric arguments such as varying sectional curvatures along the iterate trajectory. It could be that a general theory that applies to arbitrary Riemannian manifolds is far beyond reach, in which case even a better understanding of a particular manifold (e.g. the compact Stiefel manifold) would be impressive.

Deep neural networks. The battle against overfitting remains one of the most important topics of machine learning in the deep learning era. While the deep learning revolution is arguably fueled by the availability of large datasets, in many applications training data are still a scarce resource. Moreover, state-of-the-art models typically have sufficient amount of capacity for overfitting even a large dataset such as the ImageNet. For vanilla classification tasks, regularization tech-

niques such as mixup have shown their superior performance. However, for tasks beyond vanilla classification, our regularization toolkit is rather limited. For example, with structured input (such as the graph representation of a molecule) or structured output (such as a sentence), the assumptions of mixup does not apply. While it is possible to embed the structured input and output in a vector space and apply mixup there, it is worth taking a step back and thinking about the basic question: *For our task at hand, what properties should a good predictor have?*

For vanilla classification tasks, a desirable property is that the model prediction is Lipschitz between training data, which is the rationale behind mixup. Hence a good question to ask first may be: what does Lipschitzness mean for a prediction task on graphs? Similarly, we should think about what kind of regularization is preferable for sequential prediction, object detection, or question answering. The more we think about these questions, the better control we will have on the models we design to ensure their desired behaviors.

While the exploding and vanishing gradient problem used to be daunting, it now appears solved thanks to the development of new architectures and initialization methods. However, we may be just at the starting point of understanding the effects of optimization algorithms to the generalization performance of a neural network. For example, is solving the exploding and vanishing gradient problem enough for achieving good performance? Or is updating the network at a constant speed (e.g. via ZeroInit) a good choice? Should the network function be initialized near zero? How can we stably train a network with very large minibatches and a large learning rate? How can we make efficient use of the representation of parameters, such as in a low-precision training setting? Further research into these topics will help to release the full potential of our models as well as hardware infrastructures.

A

Mathematical definitions

For completeness, this section provides additional technical definitions in building up mathematical concepts of Riemannian geometry, as presented in Section 1.2.1. Interested readers should also refer to a standard text, such as [Jost, 2011].

Definition A.1 (topological space). A *topological space* is a set M together with a family $\mathcal{S}$ of subsets of M satisfying the following properties:

1. $S_1, S_2 \in \mathcal{S} \implies S_1 \cap S_2 \in \mathcal{S}$
2. For any index set A, $(S_\alpha)_{\alpha \in A} \subset \mathcal{S} \implies \cup_{\alpha \in A} S_\alpha \in \mathcal{S}$
3. $\emptyset, M \in \mathcal{S}$

The sets from $\mathcal{S}$ are called *open*.

Definition A.2 (Hausdorff). A topological space is called *Hausdorff* if for any two distinct points $p_1, p_2 \in M$ there exists open sets $S_1, S_2 \in \mathcal{S}$ with $p_1 \in S_1, p_2 \in S_2, S_1 \cap S_2 = \emptyset$.

Definition A.3 (locally finite covering). A covering $(S_\alpha)_{\alpha \in A}$ is called *locally finite* if each $p \in M$ has a neighborhood that intersects only finitely many S_α.

Definition A.4 (paracompact). M is called *paracompact* if any open covering possesses a locally finite refinement, i.e. for any open covering $(S_\alpha)_{\alpha \in A}$ there exists a locally finite open covering $(S'_\beta)_{\beta \in B}$ with

$$\forall \beta \in B \; \exists \, \alpha \in A : S'_\beta \subset S_\alpha.$$

Definition A.5 (homeomorphism). A bijective map is called a *homeomorphism* if it maps every open set to an open set.

Definition A.6 (differentiable). An atlas $\{U_\alpha, x_\alpha\}$ on a manifold is called *differentiable* if all chart transitions $x_\beta \circ x_\alpha^{-1} : x_\alpha(U_\alpha \cap U_\beta) \to x_\beta(U_\alpha \cap U_\beta)$ are differentiable of class C^∞. A maximal differentiable atlas is called a differentiable structure. A *differentiable manifold* of dimension d is a manifold of dimension d with a differentiable structure.

We assume all atlases are differentiable. A chart is called compatible with an atlas if adding the chart to the atlas yields again an atlas. An atlas is called maximal if it consists of all charts that are compatible with it.

Definition A.7 (diffeomorphism). A map $h : M \to M'$ between differentiable manifolds M and M' with charts $\{U_\alpha, x_\alpha\}$ and $\{U'_\alpha, x'_\alpha\}$ is called *differentiable* if all maps $x'_\beta \circ h \circ x_\alpha^{-1}$ are differentiable of C^∞ where defined. Such a map is called a *diffeomorphism* if it is bijective and differentiable in both directions.

For purposes of differentiation, a differentiable manifold locally has the structure of Euclidean space. With a differentiable structure and a diffeomorphism, we can now define the notion of *tangent spaces*. From now on we adopt the *Einstein summation convention* that an index occuring twice in a product is to be summed over (but with the summation symbol omitted), e.g. $v^i g_{ij}$ represents $\sum_i v^i g_{ij}$.

Definition A.8 (tangent space, Euclidean). Let $x = (x^1, \ldots, x^d)$ be Euclidean coordinates of $\mathbb{R}^d$, $\mathcal{M} \subset \mathbb{R}^d$ an open set, $x_0 \in \mathcal{M}$. The tangent space of S at the point x_0, denoted $T_{x_0}\mathcal{M}$, is the space $\{x_0\} \times E$ where E is the d-dimensional vector space spanned by the partial derivatives $\{\frac{\partial}{\partial x^i}\}_{i=1}^d$ at the point x_0. If $\mathcal{M} \subset \mathbb{R}^d, \mathcal{M}' \subset \mathbb{R}^c$ are open, and $f : \mathcal{M} \to \mathcal{M}'$ is differentiable, we define the *derivative* $df(x_0)$ for $x_0 \in \mathcal{M}$ as the induced linear map between the tangent space

$$\begin{aligned} df(x_0) : T_{x_0}\mathcal{M} &\to T_{f(x_0)}\mathcal{M}' \\ v = v^i \frac{\partial}{\partial x^i} &\mapsto v^i \frac{\partial f^j}{\partial x^i}(x_0) \frac{\partial}{\partial f^j}, \end{aligned}$$

The mapping $\pi : T\mathcal{M} \to \mathcal{M}, (x, v) \mapsto x$ is called a tangent bundle of $\mathcal{M}$. $T\mathcal{M}$ is called the total space of the tangent bundle.

Definition A.9 (immersion). A differentiable map $f : \mathcal{M} \to \mathcal{N}$ is called an *immersion*, if for any $x \in \mathcal{M}$ the derivative $df : T_x\mathcal{M} \to T_{f(x)}\mathcal{N}$ is injective.

If an immersion $f : \mathcal{M} \to \mathcal{N}$ maps $\mathcal{M}$ homeomorphically onto its image in $\mathcal{N}$, f is called a *differentiable embedding*. It can be shown that any immersion is locally a differentiable embedding.

Definition A.10 (submanifold). If $f : \mathcal{M} \to \mathcal{N}$ is a differentiable embedding, $f(\mathcal{M})$ is called a *differentiable submanifold* of $\mathcal{N}$.

A subset $\mathcal{N}'$ of $\mathcal{N}$, equipped with the relative topology, is a differentiable submanifold of N, if $\mathcal{N}'$ is a manifold and the inclusion is a differentiable embedding. Charts on $\mathcal{N}'$ are given by restrictions of charts of $\mathcal{N}$ to $\mathcal{N}'$.

Definition A.11 (length). Let $[a, b]$ be a closed interval in $\mathbb{R}$ and $\gamma : [a, b] \to \mathcal{M}$ a smooth curve. The *length* of γ is defined as

$$L(\gamma) \triangleq \int_a^b \left\| \frac{d\gamma}{dt}(t) \right\| dt = \int_a^b \sqrt{g_{ij}(x(\gamma(t)))\dot{x}^i(t)\dot{x}^j(t)}dt.$$

The length of a piecewise smooth curve may be defined as the sum of the lengths of the smooth pieces.

Definition A.12 (energy). The *energy* of a curve γ is defined as

$$E(\gamma) \triangleq \frac{1}{2}\int_a^b \left\| \frac{d\gamma}{dt}(t) \right\|^2 dt = \int_a^b g_{ij}(x(\gamma(t)))\dot{x}^i(t)\dot{x}^j(t)dt.$$

Definition A.13 (distance). The *distance* between two points p, q on a Riemannian manifold can be defined as

$$d(p,q) \triangleq \inf\{L(\gamma) : \gamma : [a,b] \to \mathcal{M} \text{ is piecewise smooth with } \gamma(a) = p, \gamma(b) = q\}.$$

Definition A.14 (geodesic). A smooth curve $\gamma = [a,b] \to \mathcal{M}$ is called a *geodesic* if it satisfies

$$\ddot{x}^i(t) + \Gamma^i_{jk}(x(t))\dot{x}^j(t)\dot{x}^k(t) = 0, \quad \text{for } i = 1, \ldots, d.$$

where $\Gamma^i_{jk} \triangleq \frac{1}{2}g^{il}(g_{jl,k} + g_{kl,j} - g_{jk,l})$ are called *Christoffel symbols* with $(g^{ij})_{i,j=1,\ldots,d} = (g_{ij})^{-1}$ and $g_{jl,k} = \frac{\partial}{\partial x^k} g_{jl}$.

This definition of geodesic is derived from computing the stationary points of the energy functional. It can be shown that for a geodesic, $\frac{d}{dt}\langle \dot{x}, \dot{x} \rangle \equiv 0$, i.e. the curve is parametrized proportionally to arc length. Furthermore, for any compact Riemannian manifold, there is always more than one geodesic connection between any two points. However, if p and q are sufficiently close, then the shortest geodesic is unique. Indeed, an alternative characterization of geodesics is that they are *locally distance minimizing* curves on a manifold.

Definition A.15 (geodesically complete). A Riemannian manifold $\mathcal{M}$ is *geodesically complete* if for all $p \in \mathcal{M}$, the exponential map Exp_p is defined on all of $T_p\mathcal{M}$.

The Hopf-Rinow Theorem shows that any Riemannian manifold that is complete as a metric space is also geodesically complete. The next definition characterizes the domain of the inverse of an exponential map, and is important for implementing and analyzing a Riemannian optimization algorithm.

Definition A.16 (injectivity radius). Let $\mathcal{M}$ be a Riemannian manifold and $p \in \mathcal{M}$. The *injectivity radius* of p is defined as $\mathrm{inj}(p) \triangleq \sup\{\rho > 0 : \mathrm{Exp}_p \text{ is defined on } d_\rho(0) \subset T_p\mathcal{M} \text{ and is injective}\}$ where $d_\rho(0)$ is the Euclidean ball of radius ρ centered at 0 in the tangent space of p. The *injective radius* of $\mathcal{M}$ is $\mathrm{inj}(\mathcal{M}) \triangleq \inf_{p\in\mathcal{M}} i(p)$.

To study tangent vectors on Riemannian manifolds, we further introduce:

Definition A.17 (vector field). A differentiable *vector field* X on a differentiable manifold $\mathcal{M}$ is an association $\mathcal{M} \ni p \mapsto X_p \in T_p\mathcal{M}$ such that in every chart $\varphi : U \to V$ with coordinates $x^1, \ldots, x^n$, the coefficients $\xi^i : U \to \mathbb{R}$ in the representation $X_p = \xi^i(p) \left.\frac{\partial}{\partial x^i}\right|_p$ are differentiable functions.

We now recall the definition of directional derivative for a vector field:

Definition A.18 (directional derivative, vector field). Let Y be a differentiable vector field, defined on an open set of $\mathbb{R}^d$, and let X be a fixed directional vector at some fixed point p of this open set, i.e. $(p, X) \in T_p\mathbb{R}^d$. The the expression $D_X Y|_p \triangleq DY|_p(X) = \lim_{t\to 0} \frac{1}{t}(Y(p + tX) - Y(p))$ is called the *directional derivative* of Y in the direction of X, where DY denotes the Jacobi matrix.

It can be shown that $D_X Y|_p = \sum_i X^i D_{e_i} Y|_p = \sum_i X^i \lim_{t\to 0}(Y(p + te_i) - Y(p))$, where $\{e_i\}_{i=1}^d$ are the standard basis.

The definition of directional derivative is intuitive and natural; however, it has the disadvantage that even the derivative of tangential vectors in the tangent direction may have a normal component, which depends on how the manifold is embedded. Therefore, a better definition to characterize the intrinsic geometry of a manifold is to consider only the component of that directional derivative which is tangent to the surface. This leads to the definition of covariant derivative.

Definition A.19 (covariant derivative, vector field). Let $X : \mathcal{M} \to T\mathcal{M}, Y : \mathcal{M} \to T\mathcal{M}$ be two vector fields on a Riemannian manifold $\mathcal{M}$. The *covariant derivative* of Y in the direction of X is again a vector field of $\mathcal{M}$, defined as $\nabla_X Y \triangleq D_X Y - \langle D_X Y, \nu\rangle \nu$, where ν is the surface normal (at the base of Y). Intuitively, covariant derivative is the projection of directional derivative into the tangent space of its base.

It can be shown that the covariant derivative is a quantity of the intrinsic geometry of the surface.

That a vector field Y in Euclidean space is constant is equivalent to saying that the directional derivative $D_X Y$ vanishes in all directions X. On a Riemannian manifold, generalization of the concept of a constant vector field can be developed using covariant derivative in the following sense:

Definition A.20 (parallel vector field). Let Y be a tangent vector field of a Riemannian manifold. Y is called *parallel* if $\nabla_X Y \equiv 0$ for every tangent vector X.

If Y is a vector field along a regular curve $\gamma(t)$, then Y is said to be *parallel along* γ, if $\nabla_X Y = 0$ for every X which is tangent to γ.

Notably, this definition provides an alternative (and more general) way to define geodesics for manifolds not necessarily equipped with a Riemannian metric: A non-constant curve γ on a surface is called a *geodesic* (or *auto-parallel*) if $\nabla_{\dot\gamma}\dot\gamma \equiv 0$ holds along the curve γ.

In general there is no non-trivial parallel vector field on open sets of surfaces, however there are always parallel vector fields along given curves, which leads to the definition of parallel transport.

Bibliography

Pierre-Antoine Absil and Kyle A. Gallivan. Accelerated line-search and trust-region methods. *SIAM J. Numer. Anal.*, 47(2):997–1018, 2009.

Pierre-Antoine Absil and Ivan Oseledets. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 2014.

Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Appl. Math.*, 80(2):199–220, 2004. ISSN 0167-8019.

Pierre-Antoine Absil, Christopher G. Baker, and Kyle A. Gallivan. Trust-region methods on Riemannian manifolds. *Found. Comput. Math.*, 7(3):303–330, July 2007. doi: 10.1007/s10208-005-0179-9.

Pierre-Antoine Absil, Mariya Ishteva, Lieven De Lathauwer, and Sabine Van Huffel. A geometric Newton method for Oja's vector field. *Neural Comput.*, 21(5): 1415–1433, May 2009a.

Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009b.

Pierre-Antoine Absil, Luca Amodei, and Gilles Meyer. Two Newton methods on the manifold of fixed-rank matrices endowed with Riemannian quotient geometries. *Computational Statistics*, 29(3-4):569–590, 2014. doi: 10.1007/s00180-013-0441-6.

Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 78–86, 2015.

Naman Agarwal, Zeyuan Allen Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima for nonconvex optimization in linear time. *CoRR*, abs/1611.01146, 2016.

Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. *arXiv:1603.05643*, 2016.

Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv:1407.1537*, 2014.

Luigi Ambrosio, Nicola Gigli, Giuseppe Savaré, et al. Metric measure spaces with Riemannian Ricci curvature bounded from below. *Duke Mathematical Journal*, 163 (7):1405–1490, 2014.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Vaino Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, ICML, 2016.

Yossi Arjevani, Shai Shalev-Shwartz, and Ohad Shamir. On lower and upper bounds for smooth and strongly convex optimization problems. *arXiv:1503.06833*, 2015.

Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *ICML*, 2017.

Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Miroslav Bacák. *Convex analysis and optimization in Hadamard spaces*. Berlin, Boston: De Gruyter, 2014.

Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In *Advances in Neural Information Processing Systems*, pages 773–781, 2013.

Christopher G. Baker, Pierre-Antoine Absil, and Kyle A. Gallivan. An implicit trust-region method on Riemannian manifolds. *IMA J. Numer. Anal.*, 28(4):665–689, 2008. doi: doi:10.1093/imanum/drn029.

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If ResNets are the answer, then what is the question? *arXiv preprint arXiv:1702.08591*, 2017.

Peter Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *NIPS*, 2017.

Glaydston C Bento, Orizon P Ferreira, and Jefferson G Melo. Iteration-complexity of gradient, subgradient and proximal point methods on Riemannian manifolds. *Journal of Optimization Theory and Applications*, 173(2):548–562, 2017.

Rajendra Bhatia. *Positive Definite Matrices*. Princeton University Press, 2007.

Louis J Billera, Susan P Holmes, and Karen Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

Dario A Bini and Bruno Iannazzo. Computing the Karcher mean of symmetric positive definite matrices. *Linear Algebra and its Applications*, 438(4):1700–1710, 2013.

Richard L Bishop and Barrett O'Neill. Manifolds of negative curvature. *Transactions of the American Mathematical Society*, 145:1–49, 1969.

Silvère Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

Pierre B. Borckmans, S. Easter Selvan, Nicolas Boumal, and Pierre-Antoine Absil. A Riemannian subgradient algorithm for economic dispatch with valve-point effect. *J. Comput. Applied. Math.*, 255:848–866, 2013. doi: 10.1016/j.cam.2013.07.002.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Nicolas Boumal and Pierre-Antoine Absil. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. *Linear Algebra and its Applications*, 475:200–239, 2015. doi: 10.1016/j.laa.2015.02.027. URL http://www.sciencedirect.com/science/article/pii/S0024379515001342.

Nicolas Boumal, Pierre-Antoine Absil, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 2016a.

Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems*, pages 2757–2765, 2016b.

Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 8(1):67–127, 2007.

Martin R Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319. Springer, 1999.

Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to Nesterov's accelerated gradient descent. *arXiv:1506.08187*, 2015.

Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33. American Mathematical Society Providence, 2001.

Yu Burago, Mikhail Gromov, and Grigori Perelman. A.D. Alexandrov spaces with curvature bounded below. *Russian mathematical surveys*, 47(2):1–58, 1992.

Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2):21–30, 2008.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for non-convex optimization. *CoRR*, abs/1611.00756, 2016.

Yair Carmon, Oliver Hinder, John C Duchi, and Aaron Sidford. "Convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. *arXiv preprint arXiv:1705.02766*, 2017.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir N. Vapnik. Vicinal risk minimization. *NIPS*, 2000.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013. URL http://arxiv.org/abs/1312.3005.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849, 2018. URL http://arxiv.org/abs/1804.09849.

Anoop Cherian and Suvrit Sra. Riemannian dictionary learning and sparse coding for positive definite matrices. *arXiv preprint arXiv:1507.02772*, 2015.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. *ICML*, 2017.

Marco Congedo, Bijan Afsari, Alexandre Barachant, and Maher Moakher. Approximate joint diagonalization and geometric mean of symmetric positive definite matrices. *PloS one*, 10(4):e0121423, 2015.

Dario Cordero-Erausquin, Robert J McCann, and Michael Schmuckenschläger. A Riemannian interpolation inequality à la Borell, Brascamp and Lieb. *Inventiones mathematicae*, 146(2):219–257, 2001.

Stephen C Cowin and Guoyu Yang. Averaging anisotropic elastic constant data. *Journal of Elasticity*, 46(2):151–180, 1997.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks. *NIPS*, 2017.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M Rush. Latent alignment and variational attention. *Thirty-second Conference on Neural Information Processing Systems (NIPS)*, 2018.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with Cutout. *arXiv preprint arXiv:1708.04552*, 2017a.

Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *ICLR Workshops*, 2017b.

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Harris Drucker and Yann LeCun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Gamaleldin F Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. *arXiv preprint arXiv:1803.05598*, 2018.

Orizon P. Ferreira and Paulo Roberto Oliveira. Subgradient algorithm on Riemannian manifolds. *Journal of Optimization Theory and Applications*, 97(1):93–104, 1998.

Orizon P. Ferreira and Paulo Roberto Oliveira. Proximal point algorithm on Riemannian manifolds. *Optimization*, 51(2):257–270, 2002.

Nicolas Flammarion and Francis Bach. From averaging to acceleration, there is only a step-size. In *Conference on Learning Theory*, pages 658–695, 2015.

P Thomas Fletcher and Sarang Joshi. Riemannian geometry for the statistical analysis of diffusion tensor data. *Signal Processing*, 87(2):250–262, 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001.

Daniel Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, 1982.

Dan Garber and Elad Hazan. Fast and simple PCA via convex optimization. *arXiv preprint arXiv:1509.05647*, 2015.

Rong Ge, Chi Jin, and Yi Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. *arXiv:1704.00708*, 2017.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Pinghua Gong and Jieping Ye. Linear convergence of variance-reduced stochastic gradient without strong convexity. *arXiv preprint arXiv:1406.1102*, 2014.

Ian Goodfellow. Tutorial: Generative adversarial networks. *NIPS*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 2014.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv*, 2017.

Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*. IEEE, 2013.

Robert E. Greene and Katsuhiro Shiohama. Convex functions on complete noncompact manifolds: topological structure. *Inventiones mathematicae*, 63(1):129–157, 1981.

Mikhail Gromov. Manifolds of negative curvature. *J. Differential Geom*, 13(2):223–230, 1978.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. *NIPS*, 2017.

Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *arXiv preprint arXiv:1803.01719*, 2018.

Mehrtash T Harandi, Conrad Sanderson, Richard Hartley, and Brian C Lovell. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *Computer Vision–ECCV 2012*, pages 216–229. Springer, 2012.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

Nick Harvey, Chris Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. *JMLR*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

David J Heeger. Normalization of cell responses in cat striate cortex. *Visual neuroscience*, 9(2):181–197, 1992.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *NIPS*, 2017.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. *arXiv preprint arXiv:1803.01814*, 2018.

Reshad Hosseini and Suvrit Sra. Matrix manifold optimization for Gaussian mixtures. In *Advances in Neural Information Processing Systems (NIPS)*, 2015a.

Reshad Hosseini and Suvrit Sra. Manifold optimization for mixture modeling. *arXiv:1506.07677*, 2015b.

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

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

Wen Huang, Pierre-Antoine Absil, and Kyle A. Gallivan. A Riemannian symmetric rank-one trust-region method. *Mathematical Programming*, 2014. doi: 10.1007/s10107-014-0765-1.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Mariya Ishteva, P-A Absil, Sabine Van Huffel, and Lieven De Lathauwer. Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme. *SIAM Journal on Matrix Analysis and Applications*, 32 (1):115–135, 2011.

Ben Jeuris, Raf Vandebril, and Bart Vandereycken. A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *Electronic Transactions on Numerical Analysis*, 39:379–402, 2012.

Bo Jiang, Shiqian Ma, Anthony Man-Cho So, and Shuzhong Zhang. Vector transport-free SVRG with general retraction for Riemannian optimization: Complexity analysis and practical implementation. *arXiv preprint arXiv:1705.09059*, 2017.

Chi Jin, Sham M Kakade, Cameron Musco, Praneeth Netrapalli, and Aaron Sidford. Robust shift-and-invert preconditioning: Faster and more sample efficient algorithms for eigenvector computation. *arXiv:1510.08896*, 2015.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

Jürgen Jost. *Riemannian Geometry and Geometric Analysis*. Springer Science & Business Media, 2011.

Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

Hiroyuki Kasai, Hiroyuki Sato, and Bamdev Mishra. Riemannian stochastic variance reduced gradient on Grassmann manifold. *arXiv preprint arXiv:1605.07367*, 2016.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

Jakub Konečnỳ and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv:1312.1666*, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *NIPS*, 2012.

Wolfgang Kühnel. *Differential geometry*, volume 77. American Mathematical Soc., 2015.

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 2001.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472, 2016.

Bas Lemmens and Roger Nussbaum. *Nonlinear Perron-Frobenius Theory*, volume 189. Cambridge University Press, 2012.

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

Kuang Liu, 2017. URL https://github.com/kuangliu/pytorch-cifar.

Xin-Guo Liu, Xue-Feng Wang, and Wei-Guo Wang. Maximization of matrix trace function of product Stiefel manifolds. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1489–1506, 2015.

Xiuwen Liu, Anuj Srivastava, and Kyle Gallivan. Optimal linear representations of images for object recognition. *IEEE TPAMI*, 26(5):662–666, 2004.

Yuanyuan Liu, Fanhua Shang, James Cheng, Hong Cheng, and Licheng Jiao. Accelerated first-order methods for geodesically convex optimization on Riemannian manifolds. In *Advances in Neural Information Processing Systems*, pages 4868–4877, 2017.

Siwei Lyu and Eero P Simoncelli. Nonlinear image representation using divisive normalization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Wolfgang Meyer. Toponogov's theorem and applications. *SMR*, 404:9, 1989.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Bamdev Mishra and Rodolphe Sepulchre. Riemannian preconditioning. *SIAM Journal on Optimization*, 26(1):635–660, 2016.

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

Maher Moakher. Means and averaging in the group of rotations. *SIAM journal on matrix analysis and applications*, 24(1):1–16, 2002.

Maher Moakher. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3):735–747, 2005.

Arkadiĭ Semenovich Nemirovsky and David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. New York: Wiley, 1983.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, pages 372–376, 1983.

Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

Erkki Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, 1992.

Yann Ollivier. A visual introduction to Riemannian curvatures and some discrete generalizations. *Analysis and Geometry of Metric Measure Spaces: Lecture Notes of the 50th Séminaire de Mathématiques Supérieures (SMS), Montréal*, 56:197–219, 2011.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*, 2018.

Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pages 4785–4795, 2017.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *ICLR Workshops*, 2017.

Peter Petersen. *Riemannian geometry*, volume 171. Springer Science & Business Media, 2006.

Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, 2008.

Boris T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, January 1963.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, pages 314–323, 2016.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 707. John Wiley & Sons, 2011.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

Chafik Samir, Pierre-Antoine Absil, Anuj Srivastava, and Eric Klassen. A gradient-descent method for curve fitting on Riemannian manifolds. *Foundations of Computational Mathematics*, pages 49–73, 2012. ISSN 1615-3375.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

Ohad Shamir. A stochastic PCA and SVD algorithm with an exponential convergence rate. In *International Conference on Machine Learning (ICML-15)*, pages 144–152, 2015.

Wenling Shang, Justin Chiu, and Kihyuk Sohn. Exploring normalization in deep residual networks with concatenated rectified linear units. In *AAAI*, pages 1509–1516, 2017.

Hao Shen, Stefanie Jegelka, and Arthur Gretton. Fast kernel-based independent component analysis. *Signal Processing, IEEE Transactions on*, 57(9):3498–3511, 2009.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham,

Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Patrice Simard, Yann LeCun, John Denker, and Bernard Victorri. Transformation invariance in pattern recognition—tangent distance and tangent propagation. *Neural networks: tricks of the trade*, 1998.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

Steven T Smith. Optimization techniques on Riemannian manifolds. *Fields institute communications*, 3(3):113–135, 1994.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *ICLR Workshops*, 2015.

Suvrit Sra. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *Advances in Neural Information Processing Systems*, pages 144–152, 2012.

Suvrit Sra and Reshad Hosseini. Geometric optimisation on positive definite matrices for elliptically contoured distributions. In *Advances in Neural Information Processing Systems*, pages 2562–2570, 2013.

Suvrit Sra and Reshad Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM J. Optimization (SIOPT)*, 25(1):713–739, 2015.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere II: Recovery by Riemannian trust-region method. *arXiv:1511.04777*, 2015.

Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere I: Overview and the geometric picture. *IEEE Transactions on Information Theory*, 63(2):853–884, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Mingkui Tan, Ivor W Tsang, Li Wang, Bart Vandereycken, and Sinno J Pan. Riemannian pursuit for big matrix recovery. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1539–1547, 2014.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal of Optimization*, 2009.

Constantin Udriste. *Convex functions and optimization methods on Riemannian manifolds*, volume 297. Springer Science & Business Media, 1994.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 1971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Andreas Veit, 2017. URL https://github.com/andreasveit.

Peter Warden, 2017. URL https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html.

Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, page 201614734, 2016.

Ami Wiesel. Geodesic convexity and covariance estimation. *Signal Processing, IEEE Transactions on*, 60(12):6182–6189, 2012.

Yuxin Wu and Kaiming He. Group normalization. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CVPR*, 2016.

Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. ShakeDrop regularization. *arXiv preprint arXiv:1802.02375*, 2018.

Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pages 7103–7114, 2017.

Xinru Yuan, Wen Huang, Pierre-Antoine Absil, and Kyle Gallivan. A Riemannian limited-memory BFGS algorithm for computing the matrix geometric mean. *Procedia Computer Science*, 80:2147–2157, 2016.

Sergey Zagoruyko and Nikos Komodakis, 2016a. URL https://github.com/szagoruyko/wide-residual-networks.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016b.

Chiyuan Zhang, 2017. URL https://github.com/pluskid/fitting-random-labels.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. In *Conference on Learning Theory*, pages 1617–1638, 2016.

Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds. In *Advances in Neural Information Processing Systems*, pages 4592–4600, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Junyu Zhang and Shuzhong Zhang. A cubic regularized Newton's method over Riemannian manifolds. *arXiv preprint arXiv:1805.05565*, 2018.

Teng Zhang, Ami Wiesel, and Maria S Greco. Multivariate generalized gaussian distribution: Convexity and graphical models. *IEEE Transactions on Signal Processing*, 61(16):4141–4148, 2013.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv*, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.