Generalized Adversarial Training in Riemannian Space

Shufei Zhang
Department of Electrical and Electronic Engineering
Xi'an Jiaotong-Liverpool University
Suzhou, China
Shufei.Zhang@xjtlu.edu.cn

Kaizhu Huang*
Department of Electrical and Electronic Engineering
Xi'an Jiaotong-Liverpool University
Suzhou, China
Kaizhu.Huang@xjtlu.edu.cn

Rui Zhang
Department of Mathematical Sciences
Xi'an Jiaotong-Liverpool University
Suzhou, China
rui.zhang02@xjtlu.edu.cn

Amir Hussain
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, U.K.
a.hussain@napier.ac.uk

***Abstract*—Adversarial examples, referred to as augmented data points generated by imperceptible perturbations of input samples, have recently drawn much attention. Well-crafted adversarial examples may even mislead state-of-the-art deep neural network (DNN) models to make wrong predictions easily. To alleviate this problem, many studies have focused on investigating how adversarial examples can be generated and/or effectively handled. All existing works tackle this problem in the Euclidean space. In this paper, we extend the learning of adversarial examples to the more general Riemannian space over DNNs. The proposed work is important in that (1) it is a generalized learning methodology since Riemmanian space will be degraded to the Euclidean space in a special case; (2) it is the first work to tackle the adversarial example problem tractably through the perspective of Riemannian geometry; (3) from the perspective of geometry, our method leads to the steepest direction of the loss function, by considering the second order information of the loss function. We also provide a theoretical study showing that our proposed method can truly find the descent direction for the loss function, with a comparable computational time against traditional adversarial methods. Finally, the proposed framework demonstrates superior performance over traditional counterpart methods, using benchmark data including MNIST, CIFAR-10 and SVHN.**

I. Introduction

Recently Deep Neural Networks (DNN) achieve a big success on a wide range of challengeable tasks in both pattern recognition and data mining. However, recent studies have found that DNNs can be easily fooled by some special input called adversarial examples which are referred to as augmented data points generated by imperceptible perturbation of input samples [1]–[3].

There have been a lot of proposals studying how to generate more powerful adversarial examples, and how to build up robust networks to defend them [1], [4]. This interesting problem was first studied in [5]. Then a more powerful approach Fast Gradient Sign Method (FGSM) is later proposed in [4] and further extended to a more general case with l_p constraint for perturbation [6], [7]. The multi-step variant FGSM^k was proposed in [8] which is essentially projected gradient decent (PGD) on the negative loss. Aside from studying how to generate adversarial examples, some researchers developed methods to defend them. The adversarial training was proposed by [4], [6] which augmented the training set with adversarial examples. This method not only increases the model robustness for adversarial examples but also improves the generalization for benign samples. Some feature squeezing [9] and defensive distillation [10] were also exploited to resist adversarial attacking. An information regularized framework is proposed in [11] perturbing the input samples through the distribution. Furthermore, [12] have demonstrated that the adversarial examples are a dense region of pixel space instead of isolated points.

All these existing adversarial training methods simply consider the adversarial example problem in the Euclidean space with the orthonormal coordinate system. Specifically, these traditional adversarial training methods aim to solve a robust optimization problem [6]:

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_{(x,y)\backsim\mathcal{D}}[\max_{\epsilon} \mathcal{L}(x+\epsilon, y, \theta)] \\ s.t. \quad & d_E(x, x+\epsilon) \leq \sigma \end{aligned} \tag{1}$$

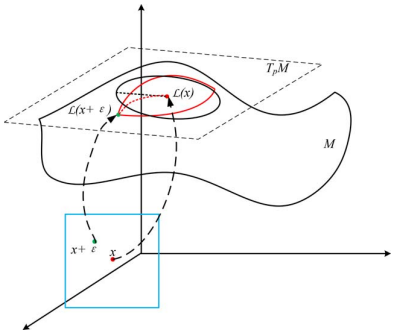
where $\mathcal{L}$ denotes a loss function, the pair of input and label (x, y) is assumed to be drawn from the data distribution $\mathcal{D}$, and $d_E(x, x+\epsilon)$ represents the Euclidean distance between the small perturbed sample $x+\epsilon$ and x, which are given as $\|\epsilon\|_2$.

The robust optimization problem is defined as a min-max problem with respect to the worst perturbation ϵ and the best model parameters θ. The adversarial example is restricted within the l_2-ball around benign example x. It can also be extended in l_p [6] where FGSM can be seen as a special case with $p = \infty$. Such restriction is defined in Euclidean space and

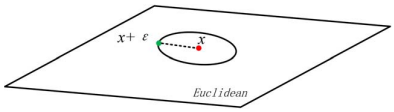
Corresponding Author: Kaizhu Huang

2374-8486/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDM.2019.00093

the similarity between two points is measured by Minkowski distance.



(a) Riemannian space



(b) Euclidean space

Fig. 1: Illustration of Riemannian metric and Euclidean metric. (a): M is a Riemannian manifold defined over DNN whose metric tensor is defined by G varying with different x. T_pM represents the tangent space of M at the point $\mathcal{L}(x)$. The blue square denotes the chart of the manifold M, where x and $x+\epsilon$ are two points on the chart. $\mathcal{L}(x)$ and $\mathcal{L}(x+\epsilon)$ are the images of x and $x+\epsilon$ on the manifold. The red circle is the region defined by $d(\mathcal{L}(x), \mathcal{L}(x+\epsilon)) \leq \sigma$ on manifold M. When ϵ is the vector with a small magnitude, the region in red circle can be approximated by the region in black circle $\epsilon^T G \epsilon \leq \sigma^2$ in tangent space. (b): The region defined in Euclidean space $\|\epsilon\|_2 \leq \sigma$.

However, data points may be in practice attached on a geometric manifold which cannot be appropriately described with Euclidean coordinate system. In other words, Euclidean distance can merely be regarded as an approximation to a more complex notion of distance. One typical example can be seen in many geographical maps where the world is depicted as a flat two-dimensional plane rather than a curved surface [13]. In these cases, the Euclidean metric would be inaccurate and even inappropriate. Moreover, existing adversarial training methods usually search the worst perturbation through the gradient of loss function with respect to x, since the gradient is considered as the steepest direction. However, in a geometric manifold, particularly in Riemannian space, the gradient of a loss function unnecessarily presents the steepest direction which means this direction cannot lead to the largest variation of loss function with constant step size. Figure 3 illustrates the difference between a Riemannian space and the Euclidean space, where the detailed mathematical notation can be seen in Section 2. Clearly, In this figure, assuming that the data are attached in the manifold as defined in Figure 3(a), the Euclidean distance may not appropriately reflect the true distance between two points (e.g., the dashed red curve connecting $\mathcal{L}(x)$ and $\mathcal{L}(x+\epsilon)$).

In this paper, we extend the traditional adversarial problem to Riemannian space and propose a novel adversarial method called Generalized Adversarial Training (GAT) in the Riemannian space. GAT is regarded as a generalized framework in that Riemannian space contains the Euclidean space as a special case. In more details, we start with defining the local coordinate system and Riemannian metric tensor to evaluate the similarity between two points in Riemannian space. We then propose to restrict the adversarial example within l_2-ball (and then extended to l_p-ball) around natural examples x on Riemannian manifold which can be viewed as the more reliable trust region. Our proposed method is to solve the adversarial problem from the perspective of geometry which is similar to Natural Gradient methods [13], however, our method is implemented in the input space instead of parameter space of DNNs.

We list the main contributions of this paper as follows: 1) To our best knowledge, this is the first work to tackle the adversarial example problem through the perspective of Riemannian geometry. 2) We study the adversarial example in the more generalized Riemannian space of which Euclidean space is a special case. 3) Our method considers the curvature information of the loss function which can be viewed as the second order method, enabling a more accurate direction of adversarial perturbation. Importantly, from the perspective of geometry, our method leads to the steepest direction of loss function in Riemannian space. 4) We also provide a series of theory showing that our proposed method can truly find the decent direction for the loss function with a comparable computational time against traditional adversarial method (one more backward backward prorogation).

It should be noted that we do not try to find the best Riemannian space to fit the training data in this paper, which is very difficult (if not possible). Instead, we assume that the metric tensor determining a specific Riemannian space is available and we propose a series of theories how to handle adversarial examples in this Riemannian space. Importantly, in practice, there are various ways to define a reasonable metric tensor over DNN, which is detailed in Section II-C and leads to remarkable results as seen in Section III.

II. Main Methodology

A. Riemannian Geometry

With the Einstein notation used in this paper, the Riemannian Manifold is defined as follows:

Definition II.1. (Riemannian Manifold [14]) In differential geometry, a Riemannian manifold (M, g) is a real smooth manifold M equipped with inner product in tangent space T_pM at each point p varying smoothly on M, defined by positive definite metric tensor g_p.

Lemma II.1. *Let $f : U \to S(\subset \mathbb{R}^{N+1})$ with open set $U \subset \mathbb{R}^N$ be a patch of a manifold S and $v(t) : (a, b) \to U$ be a curve on U. Then $\gamma(t) = f(v(t)) : (a, b) \to S \subset \mathbb{R}^{N+1}$ is a curve on S. Let $v(t_1)$ and $v(t_2)$ be two closed points on U,*

then the distance between these two points can be computed by:

$$ds^2 = g_{ij}(v(t_1))d\theta^i d\theta^j \tag{2}$$

where $d\theta^i = v^i(t_2) - v^i(t_1)$ is the i^{th} element of $d\theta$ with small magnitude and $g_{ij}(v(t_1))$ is the metric tensor of manifold S at point $f(v(t_1))$.

According to Lemma II.1, given two close data point x and $x+\epsilon$ on the patch U of manifold S (where ϵ is small), the distance between images of x and $x+\epsilon$ on Riemannian manifold S, defined as $d_R(x, x+\epsilon)$, can be practically calculated as $g_{ij}(x)\epsilon^i\epsilon^j$. Note that again, in Einstein notation, $g_{ij}(x)d\theta^i d\theta^j$ is a short form of $\sum_{i,j} g_{ij}(x)d\theta^i d\theta^j$, so do other similar formulations in this paper.

We can also rewrite ds^2 in form of inner production:

$$ds^2 = \langle d\theta^i e_i, d\theta^j e_j\rangle = \langle e_i, e_j\rangle d\theta^i d\theta^j \tag{3}$$

where basis vectors $\{e_i\}$ is the set of tangent vectors along the coordinate curves. Combining these two equations, we have $g_{ij} = \langle e_i, e_j\rangle$. Therefore, the metric tensor $G = (g_{ij})$ is the inner product of basis vectors. In the case of Euclidean space (orthonormal coordinate system), the metric tensor is:

$$g_{ij} = \delta_{ij} = \begin{cases} 1 & i{=}j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where δ_{ij} represents the Kronecker delta. The Euclidean space can be viewed as a special case of Riemannian space with the identity metric tensor. For traditional adversarial training, the data are assumed to be attached in Euclidean space with the identity metric tensor. In this paper, we consider a more general case that the data are attached in Riemannian space with positive definite metric tensor. Figure 1 illustrates the difference between the two spaces.

B. Adversarial Perturbation within l_2-ball in Riemannian Space

In this subsection, we first review the traditional adversarial example methods, analyze their limitation, and then describe our generalized framework.

To search the adversarial examples in the traditional method, one need to solve first the inner optimization problem of (1) with the constraint defined by the Euclidean norm. However, it is difficult to solve (1) directly. Usually, one can solve its relaxed version by using the first order Taylor expansion approximation for the loss function:

$$\arg\max_{\epsilon} \quad \mathcal{L}(x) + \nabla_x \mathcal{L}^T \epsilon \qquad s.t. \quad d_E(x, x+\epsilon) \le \sigma \tag{5}$$

where ϵ is a small perturbation and σ is a small constant. $\mathcal{L}(x)$ is a short form of $\mathcal{L}(x, y, \theta)$. $d_E(x, x+\epsilon)$ is defined by $\|\epsilon\|_2$. In these traditional methods, the trust region ($d_E(x, x+\epsilon) \le \sigma$) is defined in Euclidean space with the orthonormal coordinate system. In other words, in this small region, the first order Taylor expansion can approximate the loss function well. However, in most cases, this trust region is not accurate enough since the curvature information of the loss function w.r.t. x is ignored. Generally, with considering the curvature information, the coordinate system of trust region is not always orthonormal and varies with different input x. Therefore, it is more reasonable to define the constraint by a quadratic form:

$$\arg\max_{\epsilon} \quad \mathcal{L}(x) + \nabla_x \mathcal{L}^T \epsilon \quad s.t. \quad g_{ij}(x)\epsilon^i\epsilon^j \le \sigma^2 \tag{6}$$

where $G = (g_{ij}(x))$ denotes the metric tensor of Riemannian manifold over neural network varying with different x. The metric tensor G is a positive definite matrix made of the scalar products of the basis vectors as discussed in Section II-A which describes the relationship among different basis. For different x, the trust region is defined in different local coordinate system of Riemannian manifold.

Though a nonlinear Riemannian metric is involved in the new problem (6), we have developed the following theorem showing that the worst-case perturbation can be explicitly obtained.

Theorem II.2. *The optimal solution of the problem (6), i.e, the worst perturbation, is achieved when*

$$\epsilon \propto G^{-1}\nabla_x \mathcal{L} \tag{7}$$

Theorem II.2 can be solved with the Lagrangian multiplier method. The detailed proof of (7) can be seen in Appendix **1**. To calculate the worst perturbation given by (7), one needs know G, the metric tensor beforehand. In the next subsection, we describe how to define a reasonable G over neural networks.

C. Defining Riemannian Manifold Over DNN

In Section II-B, we have derived in (7) that the direction of the worst perturbation is relevant to a Riemannian manifold (associated with the metric tensor G). It is then crucially important on how to define the Riemannian manifold over a specific neural network. Since, the adversarial examples are closely related to the loss function and classification boundary, a Riemannian manifold can be reasonably defined in a way that is associated with the loss function, which usually correlates the output given by the neural network and the actual label of x. In the following, we will show how to define a reasonable G through a metric related to the loss function $\mathcal{L}$.

We first define the following metric reflecting how far between the first and second order Taylor expansions or the curvature information of the loss function at x [15]:

$$\begin{aligned} d_R^2(x, x+\epsilon) &:= 2|\mathcal{L}(x) + \epsilon^T \nabla_x \mathcal{L}(x) + \frac{1}{2}\epsilon^T H \epsilon \\ &\quad - \mathcal{L}(x) - \epsilon^T \nabla_x \mathcal{L}(x)| \\ &= |\epsilon^T H \epsilon| \end{aligned} \tag{8}$$

where H denotes the Hessian matrix of the loss function at x and $d_R(x, x+\epsilon)$ is the distance between images of x and $x+\epsilon$ on manifold which is the same as $d(\mathcal{L}(x), \mathcal{L}(x+\epsilon))$. Intuitively, given an equal ϵ, the distance between two close

points should be bigger in a local manifold with a larger curvature than in that with a smaller curvature. Using Lemma II.3, we can even simplify it with its upper bound.

Lemma II.3. *Assume H be a symmetric square matrix in $\mathbb{R}^{n\times n}$ and $r \in \mathbb{R}^n$ be a vector. Then we have $|r^T H r| \leq r^T |H| r$ and $|H|$ represents the matrix with taking the absolute value of each eigenvalue of H (proof can be seen in Appendix **4**).*

From Lemma II.3, instead of requiring $d_R^2(x, x+\epsilon) \leq \sigma^2$ (where σ is a small value), we can use its upper bound which immediately results in a metric tensor.

$$|\epsilon^T H \epsilon| \leq \epsilon^T |H| \epsilon \leq \sigma^2 \tag{9}$$

We can formulate the new optimization problem as

$$\arg\max_{\epsilon} \quad \mathcal{L}(x) + \nabla_x \mathcal{L}^T \epsilon \quad s.t. \quad \epsilon^T |H| \epsilon \leq \sigma^2 \tag{10}$$

By comparing (10) with (6), we know that the absolute Hessian matrix is a metric tensor that is associated with the loss function $\mathcal{L}$. The space defined with this metric tensor is hence a Riemmanian space.

As a short summary, starting from the curvature considering the Hessian matrix of the loss function $\mathcal{L}$, we eventually exploit its upper bound, leading to a manifold associated with the metric tensor $|H|$. It is noted that around the non-degenerate minimum of the loss function, the Hessian matrix H is positive definite matrix and the absolute Hessian matrix is the same as the Hessian matrix. Otherwise, the non-negative eigenvalues of absolute Hessian matrix keep the same as the Hessian matrix while the negative eigenvalues are changed to positive ones. The curvature information is partially kept.

According to (7), we can directly get the worst perturbation by substituting the metric tensor G with $|H|$:

$$\epsilon \propto |H|^{-1} \nabla_x \mathcal{L} \tag{11}$$

In contrast to the traditional adversarial training methods, the metric tensor $|H|$ of our method involves the curvature information of the loss function which can be seen as the second order method. Through the perspective of geometry, the direction of gradient is not guaranteed to be steepest in Riemannian space, however, the metric tensor adjusts it to the steepest one as illustrated in Figure 2. We also develop a theory in Lemma II.6 showing that the proposed GAT could indeed offer the decent direction. Note that, for DNNs, it is easy to evaluate the Hessian matrix of $\mathcal{L}$ with respect to input x by back propagation. Details can be seen in Section II-E.

Remarks: Besides the Riemannian metric discussed above, some other metrics can also be defined over neural networks. In particular, a Fisher metric defining a natural gradient [13] can also be chosen for defining the Riemannnian manifold over neural networks. The metric tensor, given as the Fisher information matrix, basically measures the change of the data that induce on the probability distribution of the output of the neural network model. We will leave the discussion of Fisher metric tensor as future work.

D. Adversarial Perturbation within l_p-ball on Manifold

In the previous subsections, we have calculated the worst perturbation within l_2-ball on Riemannian manifold. We now show how we can extend our method to l_p-ball on manifold. First, we introduce Lemma II.4:

Lemma II.4. *Let A a real symmetric positive definite matrix in $\mathbb{R}^n \times \mathbb{R}^n$. Then we have a unique positive definite matrix S in $\mathbb{R}^n \times \mathbb{R}^n$ so that $A = S^2$ (proof can be seen in Appendix **5**).*

Using Lemma II.4, we can reformulate the constraint of (10) as $\epsilon^T |H| \epsilon = \epsilon^T S S^T \epsilon = (\epsilon^T S)^2 \leq \sigma^2$, where S is a positive definite matrix called as a transformation matrix. Then we can easily extend it to l_p-ball on manifold:

$$\|\epsilon^T S\|_p = (\sum_i |\epsilon^T S|_i^p)^{1/p} \leq \sigma \tag{12}$$

Substituting (12) with the constraint of (6), we can easily evaluate the corresponding worst perturbation (details are provided in Appendix **2**):

$$\epsilon = \sigma sign(\nabla \mathcal{L}^T S^{-1}) (\frac{|\nabla \mathcal{L}^T S^{-1}|}{\|\nabla \mathcal{L}^T S^{-1}\|_{p^*}})^{\frac{1}{p-1}} S^{-1} \tag{13}$$

where p^* is the dual of p, i.e., $\frac{1}{p^*} + \frac{1}{p} = 1$. Clearly, when $p = 2$, the worst perturbation is reduced to (11) which is the case of the perturbation within l_2-ball on manifold. When $p = \infty$, our method reduces to the generalized FGSM:

$$\begin{aligned} \epsilon &= \sigma \lim_{p\to\infty} sign(\nabla \mathcal{L}^T S^{-1}) (\frac{|\nabla \mathcal{L}^T S^{-1}|}{\|\nabla \mathcal{L}^T S^{-1}\|_{p^*}})^{\frac{1}{p-1}} S^{-1} \\ &= \sigma sign(\nabla \mathcal{L}^T S^{-1}) S^{-1} \end{aligned} \tag{14}$$

Though we can evaluate the adversarial perturbation for any l_p-ball on Riemannanian manifold through (13). In this paper, we focus on the constraint of l_2-ball and will develop the corresponding adversarial training method to improve the performance of DNNs in the next subsection.

E. Adversarial Training Method with Riemannian Manifold

After studying the adversarial examples on manifold, we now consider to design an optimization method to improve the DNNs using the theory in the previous subsections. We first define the overall optimization as the robust optimization problem in a way similar to the traditional adversarial training:

$$\min_{\theta} \quad \mathbb{E}_{(x,y)\backsim\mathcal{D}}[\max_{\epsilon} \mathcal{L}(x+\epsilon, y, \theta)] \quad s.t. \quad \|\epsilon^T S\|_p \leq \sigma \tag{15}$$

In this paper, we focus on the l_2-ball constraint, while it is easily extended to the l_p-ball constraint. In the case of l_2, we can reduce the constraint in (15) to $\epsilon^T |H| \epsilon \leq \sigma$. To optimize this problem, we can first solve the inner optimization problem then followed by the outer one. We then repeat this process until it converges. The whole process is shown as Algorithm 1. On the other hand, Algorithm 2 demonstrates the function of approximating $|H|^{-1} \nabla_x \mathcal{L}(y_i, x_i, \theta)$. Hessian matrix may require a large amount of computation. In Algorithm 1, we approximate $|H|^{-1} \nabla_x \mathcal{L}(y_i, x_i, \theta)$ with the first derivative of loss

function with respect to input. Specifically, we simply modify the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method. More details can be seen in Appendix **3**.

Algorithm 1 Framework of GAT.

for number of training iterations **do**
  Sample a batch of labeled data (x_i, y_i) with size N.
  for i in 1...N **do**
    $d_i \leftarrow \overline{|H|^{-1}\nabla_x \mathcal{L}(y_i, x_i, \theta)}$
    $\epsilon^i_{adv} = \xi d_i$
  end for
  Update the parameters of neural network with stochastic gradient:
  $-\nabla_\theta \frac{1}{N} \sum_{i=1}^{N} log\mathcal{L}(y_i, x_i + \epsilon^i_{adv}, \theta)$
end for=0

Algorithm 2 Approximation for $|H|^{-1}\nabla_x \mathcal{L}(y_i, x_i, \theta)$.

function APPROHD(y_i, x_i, θ, ζ)
  Give ζ very small value
  $g_0 = \nabla_x \mathcal{L}(y_i, x_i, \theta)$
  $g_1 = \nabla_x \mathcal{L}(y_i, x_i + \zeta g_0, \theta)$
  $y = g_1 - g_0$
  $s = \zeta g_0$
  $\rho = \frac{1}{y^T s}$
  $\alpha = \rho s^T g_1$
  $q = g_1 - \alpha y$
  $r_0 = q_0$
  $\beta = \rho y^T r_0$
  $r_1 = r_0 + (\alpha - \beta)s$
  return r_1
end function=0

In Algorithm 1, $\overline{|H|^{-1}\nabla_x \mathcal{L}(y_i, x_i, \theta)}$ represents the normalization of $|H|^{-1}\nabla_x \mathcal{L}(y_i, x_i, \theta)$. We now provide the theoretical analysis showing that our proposed method truly offers the decent direction for the optimization problem as [16]. In order to do this, we first present Theorem II.5.

Theorem II.5. *(Danskin). Let S be nonempty compact topological space and $g : \mathbb{R} \times S \rightarrow \mathbb{R}$ such that $g(., \delta)$ is differentiable for every $\delta \in S$ and $\nabla_\theta g(\theta, \delta)$ is continuous on $\mathbb{R}^n \times S$. Also assume $\delta^*(\theta) = \{\delta \in \arg\max_{\delta \in S} g(\theta, \delta)\}$. Then the max-function $\phi(\theta) = \max_{\delta \in S} g(\theta, \delta)$ is locally Lipschitz continuous, directionally differentiable, and its directional derivatives satisfy: $\phi'(\theta) = \sup_{\delta \in \delta^*(\theta)} h^T \nabla g(\theta, \delta)$ In particular, if for some $\theta \in \mathbb{R}^n$ the set $\delta^*(\theta) = \{\delta^*_\theta\}$ is a singleton, the max-function is differentiable at θ and $\nabla\phi(\theta) = \nabla_\theta g(\theta, \delta^*_\theta)$.*

This theorem states that the gradients of $\phi(\theta)$ are local objects and the gradients are locally the same as that of $g(\theta, \delta^*_\theta)$. With Theorem II.5, we describe the theory showing that our proposed optimization method truly offers the decent direction:

Lemma II.6. *Let $\hat{\delta} \in S$ be a maximizer of $max_\delta \mathcal{L}(\theta, x+\theta, y)$. Then, we have that $-\nabla_{theta}\mathcal{L}(\theta, x+\hat{\delta}, y)$ is a decent direction for $\phi(\theta) = \max_{\delta \in S} \mathcal{L}(\theta, x + \delta, y)$.*

Proof. We apply Theorem II.5 that $g(\theta, \delta) := \mathcal{L}(\theta, x + \delta, y)$ and $S = B_p(\sigma)$, which is defined as the l_p-ball with radius σ on Riemannian manifold. The directional derivative in the direction of $h = \nabla_\theta \mathcal{L}(\theta, x + \hat{\delta}, y)$ satisfies:

$$\begin{aligned} \phi'(\theta, h) &= \sup_{\delta \in \hat{\delta}(\theta)} h^T \nabla_\theta L(\theta, x + \delta.y) \geq h^T h \\ &= \|\nabla_\theta \mathcal{L}(\theta, x + \hat{\delta}, y)\|_2^2 \geq 0 \end{aligned} \tag{16}$$

□

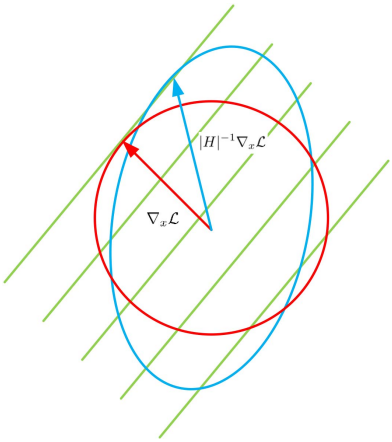


Fig. 2: The red circle denotes the region of $\|\epsilon\|_2 \leq \sigma$, while the red arrow illustrates the direction of gradient. The blue ellipse shows the region of $\epsilon^T|H|\epsilon \leq \sigma^2$, the blue arrow means the direction found by GAT, and green lines present the contour lines. The steepest direction is given by the gradient $\nabla_x \mathcal{L}$ which is orthogonal to contour lines only when an orthonormal coordinate system is used in Euclidean space. In Riemannian space, the steepest direction is not guaranteed to be orthogonal to contour lines. However, adjusted with $|H|^{-1}$, the direction of gradient can approximate the steepest one.

F. Computational Analysis

Compared with traditional adversarial training methods, our proposed GAT need compute the Hessian matrix of the loss function with respect to input additionally. It may cost extra computation to calculate the Hessian matrix. Nonetheless, we exploit in this paper the first-order derivative of the loss function to approximate the product of Hessian matrix and the vector, which is shown in Algorithm 2. Therefore, our proposed method requires backward propagation three times and forward prorogation once. Specifically, the first backward prorogation is used to approximate the Hessian matrix, the second one is used to evaluate the adversarial perturbation, and the last time is to update the parameters of DNNs. In contrast to traditional adversarial training methods, our proposed method need merely one additional backward propagation which is acceptable in practice.

III. EXPERIMENT

To validate the efficacy of our proposed method, we conduct a series of experiments on benchmark data MNIST, CIFAR-10, and SVHN. In these experiments, we compare our proposed GAT with other competitive methods. For MNIST we use the same baseline as [17]. The same base structure called 'conv-large' is used on dataset CIFAR-10 and SVHN, which follows [18].

A. Experimental Setup

We first implement our proposed GAT on handwriting dataset MNIST. Since there is no previous adversarial research on this baseline model on this dataset, we conduct the experiment on two methods (adversarial training with l_∞ and l_2 constraint) for comparison (note that adversarial training with l_∞ is just FGSM). The model is trained with $60,000$ labeled samples without any data augmentation and is tested with $10,000$ samples. We train the model with a batch size of 32 and the maximum 500 epochs. There are two hyper parameters $\{\zeta, \xi\}$ in Algorithm 1-2. We set ζ to a very small value, which is 10^{-6} in this paper. We tune the value of ξ in the range of $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$. We use the set of hyper parameters that achieved the best performance on the validation set of size $5,000$, selected randomly from the pool of training samples of size $60,000$.

For CIFAR-10, we train our proposed model GAT with $50,000$ labeled samples with data augmentation as conducted in [18] (translation and horizontal flip). The test dataset of CIFAR-10 involves $10,000$ samples. Similar to the experiment on MNIST, we set ζ to a small vaule 10^{-6}, and tune the value of ξ in the range of $\{1, 2, 5, 8, 10\}$. We run the experiments for five times and report the average performance and corresponding standard deviation.

The SVHN dataset contains 32×32 colored digit images. We train our model using the same setting as [19].

B. Evaluation on Benign Data

We first evaluate the performance of various methods on the benign data, i.e., the test data, of MNIST, CIFAR-10, and SVHN, to see if the adversarial training can lead to a classifier with better generalization.

Table I lists the performance of various methods including our proposed GAT and other competitive methods. Except for GAT, we also conduct the same experiment for the baseline model and traditional adversarial training methods with l_2-ball and l_∞-ball constraint. We conduct the experiment with this same setting for five times and calculate the mean and standard deviation. As observed, our proposed GAT demonstrates the best performance. In particular, all the adversarial methods achieve remarkably good performance, while the GAT shows a further improvement. This shows that the adversarial frameworks could actually increase the classification accuracy, due to their inherit robust property. Our proposed method could further increase the robustness and leads to better generalization over the other methods, especially the other traditional adversarial methods.

TABLE I: Test performance on MNIST

Method	MNIST Test error rate (%)
SVM	1.40
Dropout [20]	1.05
Ladder networks [17]	0.57 ± 0.02
VAT [18]	0.72
RPT [18]	0.82
Baseline [17]	0.32
Baseline+l_∞ adversarial training	0.30 ± 0.013
Baseline+l_2 adversarial training	0.26 ± 0.019
GAT	**0.22** ± 0.016

TABLE II: Test performance on CIFAR-10

Method	CIFAR-10 Test error rate (%)
Network in Network [21]	8.81
All-CNN [22]	7.25
Deeply Supervised Net [23]	7.97
Highway Network [24]	7.72
RPT [18]	6.25 ± 0.04
ResNet (1,001 layers) [25]	4.62 ± 0.2
DenseNet (190 layers) [19]	**3.46**
Baseline [18]	6.76 ± 0.07
VAT [18]	5.81 ± 0.02
Baseline+l_∞ adversarial training	6.35 ± 0.03
Baseline+l_2 adversarial training	5.82 ± 0.02
GAT	5.35 ± 0.03

Table II summarizes the results of different methods on the test set of CIFAR-10. In this experiment, we intentionally compare our proposed method GAT with the other two very deep models, i.e., the densely connected network (DenseNet) with 190 layers and very deep residual network (ResNet) with $1,001$ layers. Overall, our proposed method demonstrates competitive performance. Though not as good as the very deep networks DenseNet and ResNet, the proposed GAT shows superior performance to those adversarial learning methods and all the other remaining approaches. Once again, this shows that adversarial learning could lift the classification performance on "benign" data. This also confirms that adversarial training should be better conducted on the geometric manifold rather than the traditional Euclidean space.

TABLE III: Test performance on SVHN

Method	SVHN Test error rate (%)
Network in Network [21]	2.35
Deeply Supervised Net [23]	1.92
ResNet (110 layers) [25]	2.01
DenseNet (250 layers) [19]	1.74
Baseline [18]	2.09 ± 0.06
Baseline+l_∞ adversarial training	1.95 ± 0.05
Baseline+l_2 adversarial training	1.82 ± 0.04
GAT	**1.56** ± 0.05

We also report the performance of various methods on SVHN in Table III. Clearly observed again, our method demonstrates the best performance. It is even much better than the very deep networks (DenseNet and ResNet). More importantly, our method achieves an obvious improvement compared with the traditional adversarial training algorithms. This once

again shows the superiority of conducting adversarial training on a Riemannian space rather than the Euclidean space.

C. Robustness to Adversarial Examples

In this subsection, we compare our proposed GAT model with the other methods on their robustness against different adversarial attacks. In particular, we generate on MNIST, CIFAR-10, and SVHN adversarial examples with the number equal to their test set samples (at different perturbation levels) given by the adversarial attack methods, i.e., the proposed GAT, FGSM, and 2-norm adversarial attack. More specifically, we increase the amplitude of the adversarial perturbation ϵ from 0 to 8 with step 0.8 for all the three datasets. We then evaluate the performance of various methods on these adversarial examples.

These results are plotted in Figure 3 where our proposed GAT method achieves the best robustness against different attacks. It can be observed in Figure 3 (a)-(c), (e)-(g), (i)-(k) that the GAT model demonstrates overall flatter curves than the other adversarial frameworks on all the three attacks. One exception is at (j) where FGSM performs more robust than GAT though GAT appears robust as well. This is partially understandable, since (j) shows the robustness against the FGSM attack which the FGSM method was specially designed to defend.

Moreover, we try to examine which adversarial attack among the three (2-norm adversarial attack, FGSM attack, and the attack based on GAT) presents the worst perturbation on the traditional CNN. This result can be seen in Figure 3 (d), (h), (l). Interestingly, the attack generated by the proposed GAT is observed to degrade the performance of CNN the most, implying that our proposed method truly finds the worse perturbation than the other adversarial methods.

D. Convergence Analysis

Finally, we conduct the experiments to verify the convergence performance of our proposed method. Figure 4 shows that GAT converges well on both training and validation set of all the three datasets.

IV. Conclusion

We present a novel framework termed Generalized Adversarial Training (GAT) which generalizes the traditional adversarial training method to Riemannian space. For traditional adversarial training methods, the worst perturbation is often searched with the gradient $\nabla_x L$. However, when the data are attached on a geometric manifold defined as a Riemannian manifold, the gradient $\nabla_x L$ is not the steepest direction, leading to adversarial perturbation that is not the worst one. We present a theoretical study showing that our method leads to the steepest direction of the loss function in Riemannian space. We also develop a practical algorithm guaranteeing the decent direction for the loss function at each epoch. Comparative simulation experiments demonstrate encouraging results on benchmark datasets including MNIST, CIFAR-10 and SVHN.

Appendix

A. Find the adversarial perturbation within l_2-ball on manifold

Recall our goal is to maximize the value of the problem:

$$\begin{aligned} &\arg\max_{\epsilon} \quad \mathcal{L}(x) + \nabla_x \mathcal{L}^T \epsilon \\ s.t. \quad &g_{ij}(x)\epsilon^i \epsilon^j \leq \sigma^2 \end{aligned}$$

Since $\mathcal{L}(x)$ is independent of ϵ and the worst perturbation has the norm σ, then we have:

$$\begin{aligned} &\arg\max_{\epsilon} \nabla_x \mathcal{L}^T \epsilon \\ s.t. \quad &g_{ij}(x)\epsilon^i \epsilon^j = \sigma^2 \end{aligned}$$

Then we apply the Lagrangian multiplier method on this problem:

$$\arg\max_{\epsilon} \nabla_x \mathcal{L}^T \epsilon - \lambda(g_{ij}(x)\epsilon^i \epsilon^j - \sigma^2) \tag{17}$$

Making the first derivative of (17) with respect to ϵ zero, we have:

$$\nabla_x \mathcal{L} = \lambda g_{ij}(x)\epsilon^i$$

Then we have:

$$\epsilon = \frac{1}{\lambda} G^{-1} \nabla_x \mathcal{L}$$

Therefore:

$$\epsilon \propto G^{-1} \nabla_x \mathcal{L}$$

B. Find the adversarial perturbation with l_p-ball on manifold

The optimization problem is:

$$\begin{aligned} \epsilon = &\arg\max_{\epsilon} \quad \mathcal{L}(x) + \nabla_x \mathcal{L}^T \epsilon \\ s.t. \quad &\|\epsilon^T S\|_p \leq \sigma \end{aligned}$$

Similar to Appendix **A**, we reduce the problem to:

$$\begin{aligned} \epsilon = &\arg\max_{\epsilon} \quad \mathcal{L}(x) + \nabla_x \mathcal{L}^T \epsilon \\ s.t. \quad &\|\epsilon^T S\|_p = \sigma \end{aligned}$$

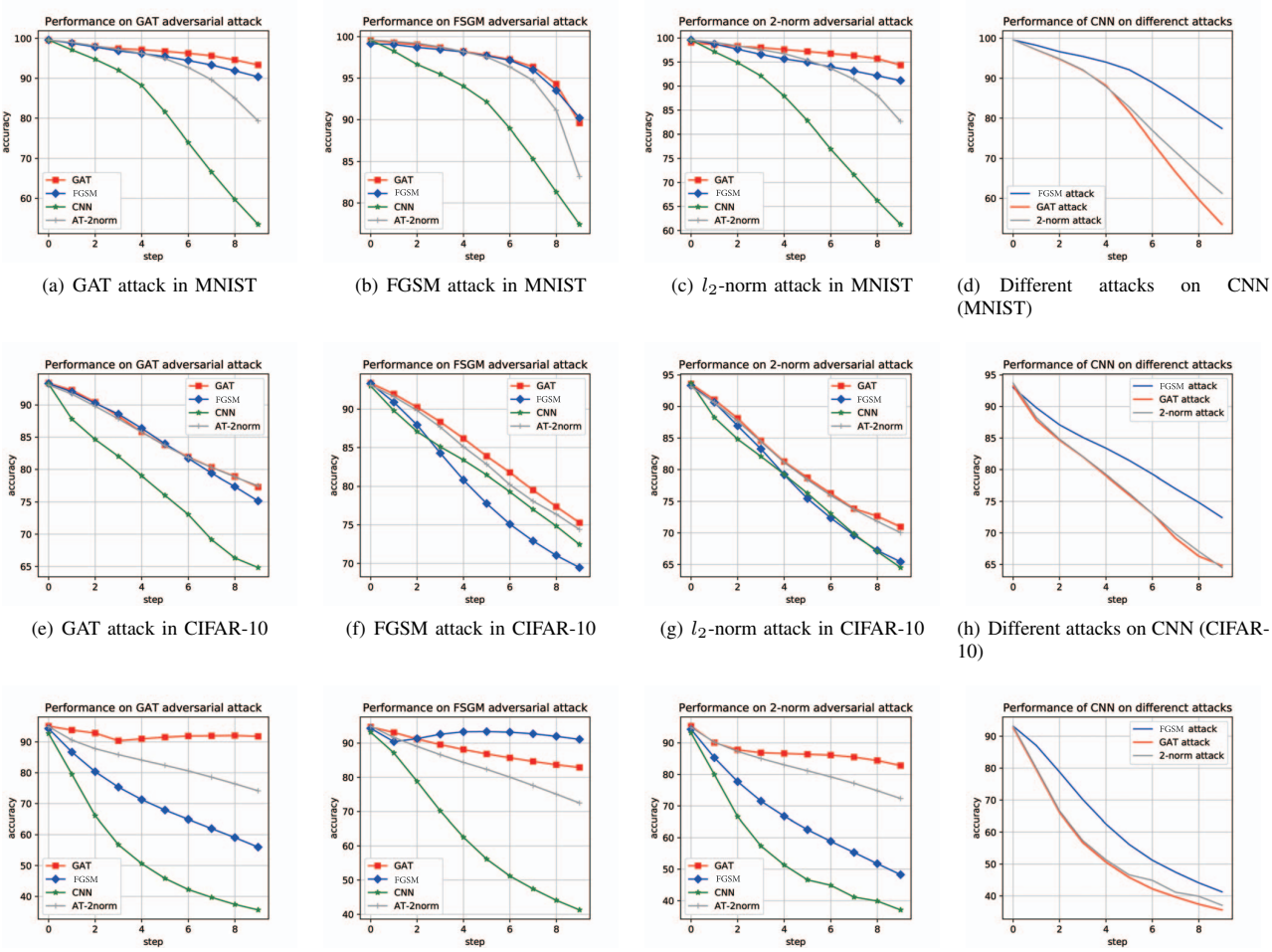
We solve it with the Lagrangian multiplier method again and set $r = \epsilon^T S$ and $f(r) \equiv \|r\|_p = \sigma$. We have

$$\nabla_x \mathcal{L} r S^{-1} = \lambda(f(r) - \sigma)$$

Then we make the first derivative respect to r:

$$\nabla_x \mathcal{L} S^{-1} = \lambda \frac{r^{p-1}}{p(\sum_i \epsilon_i^p)^{1-\frac{1}{p}}}$$

$$\nabla_x \mathcal{L} S^{-1} = \frac{\lambda}{p}(\frac{r}{\sigma})^{p-1}$$



(a) GAT attack in MNIST (b) FGSM attack in MNIST (c) l_2-norm attack in MNIST (d) Different attacks on CNN (MNIST)

(e) GAT attack in CIFAR-10 (f) FGSM attack in CIFAR-10 (g) l_2-norm attack in CIFAR-10 (h) Different attacks on CNN (CIFAR-10)

(i) GAT attack in SVHN (j) FGSM attack in SVHN (k) l_2-norm attack in SVHN (l) Different attacks on CNN (SVHN)

Fig. 3: Performance of various methods on different adversarial attacks as shown in (a)-(c) on MNIST, (e)-(h) on CIFAR-10 and (i)-(k) on SVHN. GAT shows overall the best robustness for adversarial attack (with the only exception in (j)). Performance of traditional CNN on different attacks is also shown in the last column,i.e., (d), (h), (i). Better viewed in color.

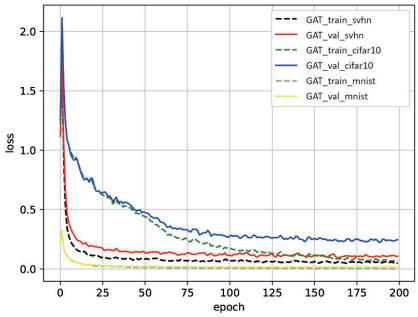


Fig. 4: The convergence curves on three datasets

$$(\nabla_x \mathcal{L} S^{-1})^{\frac{p}{p-1}} = (\frac{\lambda}{p})^{\frac{p}{p-1}} (\frac{r}{\sigma})^p \tag{18}$$

If we sum over two sides, we have

$$\sum (\nabla_x \mathcal{L} S^{-1})^{\frac{p}{p-1}} = \sum (\frac{\lambda}{p})^{\frac{p}{p-1}} (\frac{r}{\sigma})^p$$

$$\|\nabla_x \mathcal{L} S^{-1}\|_{p^*}^{p^*} = (\frac{\lambda}{p})^{p^*} * 1$$

$$(\frac{\lambda}{p}) = \|\nabla_x \mathcal{L} S^{-1}\|_{p^*} \tag{19}$$

By combining (18) and (19), we have

$$r = \sigma sign(\nabla \mathcal{L}^T S^{-1}) (\frac{|\nabla \mathcal{L}^T S^{-1}|}{\|\nabla \mathcal{L}^T S^{-1}\|_{p^*}})^{\frac{1}{p-1}}$$

Since $r = \epsilon^T S$, we have

$$\epsilon = \sigma sign(\nabla \mathcal{L}^T S^{-1}) (\frac{|\nabla \mathcal{L}^T S^{-1}|}{\|\nabla \mathcal{L}^T S^{-1}\|_{p^*}})^{\frac{1}{p-1}} S^{-1}$$

C. Find the approximation for $|\mathbf{H}|^{-1}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{y_i}, \mathbf{x_i}, \theta)$

We develop a modified BFGS method to approximate the $|H|^{-1}$. To solve the memory problem, we use the simplified Limited-memory BFGS (L-BFGS) to approximate $|H|^{-1}\nabla_x\mathcal{L}(y_i, x_i, \theta)$ directly (based on BFGS). We first present Theorem IV.1.

Theorem IV.1. *(Sherman Morrison formula [26]) Let $A \in \mathbb{R}^{n\times n}$ be an invertible square matrix and $v, u \in \mathbb{R}^n$ be column vectors. Then $A+uv^T$ is invertible if and only if $1+v^TA^{-1}u \neq 0$. And its inverse is given by*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u} \tag{20}$$

We then take the second order Taylor expansion for $\mathcal{L}(x + \epsilon_1)$ (where ϵ_1 is a small value):

$$\mathcal{L}(x + \epsilon_1) \approx \mathcal{L}(x) + \nabla\mathcal{L}(x)\epsilon_1 + \epsilon_1^T\nabla^2\mathcal{L}(x)\epsilon_1$$

If we make the first derivative on both sides with respect to ϵ_1, we can have:

$$\nabla\mathcal{L}(x + \epsilon_1) - \nabla\mathcal{L}(x) \approx H\epsilon_1$$

where H denotes the Hessian matrix of $\mathcal{L}(x)$ with respect to ϵ_1.

We can define $y = \nabla\mathcal{L}(x + \epsilon_1) - \nabla\mathcal{L}(x)$, then

$$y \approx H\epsilon_1 \tag{21}$$

Next, we use modified BFGS algorithm to approximate H and let

$$B = I + \Delta M \tag{22}$$

where B represents the approximation for H, I is identity matrix, and ΔM denotes difference matrix. Our aim is to evaluate ΔM. First we define

$$\Delta M = auu^T + bvv^T \tag{23}$$

where $a, b \in \mathbb{R}$ and $u, v \in \mathbb{R}^N$ are undetermined. It is easy to get that ΔM is symmetric. We combine (23), (22) and (21):

$$\begin{aligned} y &= I\epsilon_1 + auu^T\epsilon_1 + bvv^T\epsilon_1 \\ &= I\epsilon_1 + (au^T\epsilon_1)u + (bv^T\epsilon_1)v \end{aligned} \tag{24}$$

We can further assume

$$au^T\epsilon_1 = 1, \qquad bv^T\epsilon_1 = -1 \tag{25}$$

Therefore, we have

$$a = \frac{1}{u^T\epsilon_1}, \qquad b = -\frac{1}{v^T\epsilon_1} \tag{26}$$

If we combine (24) and (26), we have

$$u - v = y - I\epsilon_1 \tag{27}$$

Therefore, we can set

$$u = y, \qquad v = I\epsilon_1 \tag{28}$$

By combining (28) and (26), we have

$$a = \frac{1}{y^T\epsilon_1}, \qquad b = -\frac{1}{\epsilon_1^TI\epsilon_1} \tag{29}$$

Then, we combine (22), (28), and (29):

$$B = I + \frac{yy^T}{y^T\epsilon_1} - \frac{\epsilon_1\epsilon_1^T}{\epsilon_1^T\epsilon_1} \tag{30}$$

(30) can be viewed as a simplified BFGS method with one step approximation. When the dimension of s and ϵ_1 increases, a large amount of memory is needed to store the matrix ss^T and $\epsilon_1\epsilon_1^T$. To solve this problem, we then use the simplified L-BFGS method to evaluate directly $|H|^{-1}\nabla_x\mathcal{L}(y_i, x_i, \theta)$. We first use Theorem IV.1 to reformulate (30) as:

$$D = (I - \frac{\epsilon_1y^T}{y^T\epsilon_1})(I - \frac{y\epsilon_1^T}{y^T\epsilon_1}) + \frac{\epsilon_1\epsilon_1^T}{y^T\epsilon_1}$$

Then, we can easily implement the method L-BFGS as [27] and get Algorithm 3 below.

Algorithm 3 Approximation for $|H|^{-1}\nabla_x\mathcal{L}(y_i, x_i, \theta)$.

function APPROHD(y_i, x_i, θ, ζ)
  Set ζ to a very small value
  $g_0 = \nabla_x\mathcal{L}(y_i, x_i, \theta)$
  $g_1 = \nabla_x\mathcal{L}(y_i, x_i + \zeta g_0, \theta)$
  $y = g_1 - g_0$
  $s = \zeta g_0$
  $\rho = \frac{1}{y^Ts}$
  $\alpha = \rho s^Tg_1$
  $q = g_1 - \alpha y$
  $r_0 = q_0$
  $\beta = \rho y^Tr_0$
  $r_1 = r_0 + (\alpha - \beta)s$
  return r_1
end function=0

D. Proof for Lemma 2.3

Assume H be a symmetric square matrix in $\mathbb{R}^n \times \mathbb{R}^n$ and $r \in \mathbb{R}^n$ be a vector. Then we have $|r^THr| \leq r^T|H|r$ and $|H|$ represents the matrix with taking the absolute value of each eigenvalue of H.

Proof. Assume $\{e_1, e_2, ..., e_n\}$ and $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ are the eigenvectors and corresponding eigenvalues of matrix H. Then we reformulate $|r^THr|$ with eigenvectors and corresponding eigenvalues as:

$$|r^THr| = |\sum_i (r^Te_i)\lambda_i(e_i^Tr)| = |\sum_i \lambda_i(r^Te_i)^2| \tag{31}$$

We can now use the triangle inequality $|\sum_i r_i| \leq \sum_i |r_i|$ and we have:

$$\begin{aligned} |r^T H r| &\leq \sum_i |\lambda_i (r^T e_i)^2| = \sum_i (r^T e_i) |\lambda| (r^T e_i) \\ &= r^T |H| r \end{aligned} \tag{32}$$

□

V. Proof for Lemma 2.4

Let A be a real symmetric positive definite matrix in $\mathbb{R}^n \times \mathbb{R}^n$. Then we have a unique positive definite matrix S in $\mathbb{R}^n \times \mathbb{R}^n$ so that $A = S^2$.

Proof. To prove existence: Since A is a real symmetric positive definite matrix, we have $A = P^T diag(\lambda_1, ..., \lambda_n) P$, where P is an orthogonal matrix and $\{\lambda_i\}$ are the Eigen values of A ($\lambda_i > 0$). We can find $S = P^T diag(\lambda_1^{\frac{1}{2}}, ..., \lambda_n^{\frac{1}{2}})$ that $A = S^2$.

To prove uniqueness: Let B be another positive definite matrix and $A = B^2$. Since B is positive definite, we have $A = T^T diag(\mu_1, ..., \mu_n) T$, where T is an orthogonal matrix and $\{\mu_i\}$ are Eigen values of B ($\mu_i > 0$). We have $A = P^T diag(\lambda_1, ..., \lambda_n) P$, therefore

$$T^{-1} diag(\mu_1, ..., \mu_n) T = P^{-1} diag(\lambda_1, ..., \lambda_n) P \tag{33}$$

Let $U = (u_{ij})_{n \times n} = PT^{-1}$ and we have:

$$diag(\mu_1^2, ..., \mu_n^2) U = U diag(\lambda_1, ..., \lambda_n) \tag{34}$$

which is equivalent to:

$$\lambda_i u_{ij} = u_{ij} \mu_j^2 \tag{35}$$

When $\lambda \neq \mu_j^2$, $u_{ij} = 0$, we have $\lambda_i^{\frac{1}{2}} = u_{ij} \mu_j$. When $\lambda_i = \mu_j^2$, we also have $\lambda_i^{\frac{1}{2}} = u_{ij} \mu_j$. Therefore:

$$diag(\mu_1, ..., \mu_n) U = U diag(\lambda_1^{\frac{1}{2}}, ..., \lambda_n^{\frac{1}{2}}) \tag{36}$$

Then we have:

$$\begin{aligned} B &= T^{-1} diag(\mu_1, ..., \mu_n) T \\ &= P^{-1} diag(\lambda_1^{\frac{1}{2}}, ..., \lambda_n^{\frac{1}{2}}) P = S \end{aligned} \tag{37}$$

□

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[2] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[3] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 427–436.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[5] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[6] C. Lyu, K. Huang, and H.-N. Liang, "A unified gradient regularization family for adversarial examples," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 301–309.

[7] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[9] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[10] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[11] S. Zhang, K. Huang, J. Zhu, and Y. Liu, "Manifold adversarial learning," *arXiv preprint arXiv:1807.05832*, 2018.

[12] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, M. E. Houle, G. Schoenebeck, D. Song, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.

[13] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[14] G. Walschap, *Metric structures in differential geometry*. Springer Science & Business Media, 2012, vol. 224.

[15] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem inhigh-dimensional non-convex optimization," *arXiv preprint arXiv:1406.2572*, 2014.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[17] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.

[18] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[23] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[24] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[26] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 107–111, 1951.

[27] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-newton matrices and their use in limited memory methods," *Mathematical Programming*, vol. 63, no. 1-3, pp. 129–156, 1994.