# Adversarial Training in High-Dimensional Regression: Generated Data and Neural Networks

**Yue Xing**
Michigan State University

## Abstract

In recent years, studies such as Carmon et al. (2019); Gowal et al. (2021) demonstrate that incorporating additional real or generated data with pseudo-labels can enhance adversarial training through a two-stage training framework. In this paper, we perform a theoretical analysis of the asymptotic behavior of this method in high-dimensional regression problem when using two-layer neural networks. We first derive the asymptotics of the two-stage training framework using linear regression as a preliminary. Then, we analyze the convergence of two-layer neural networks in the two-stage framework. To analyze adversarial training, we track the change of the adversarial attack, and reveal that training with two-layer neural networks gives a prediction performance similar to training a linear model with some particular $\mathcal{L}_2$ regularization. To highlight our technical contribution, we are the first to investigate adversarial training in two-layer neural networks under moderate attack strength, which is different from most existing literature in vanishing attack strength.

## 1 Introduction

The development of machine learning and deep learning methods has led to breakthrough performances in various applications. However, recent studies, e.g., Goodfellow et al. (2014), have observed that these models are vulnerable when the data are perturbed by adversaries. These attacked inputs can be imperceptibly different from clean inputs to humans but can mislead the model to make incorrect predictions.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. 

Adversarial training is a popular and promising approach to improve the adversarial robustness of modern machine learning models against the aforementioned perturbations. In each training iteration, adversarial training first generates attacked samples given the current model parameters, then calculates the gradient of the model based on these augmented data. This procedure aims to make the trained model less susceptible to adversarial attacks in the testing data.

There have been fruitful results in the theoretical justification and methodology development of adversarial training. Among various research directions, one promising aspect is improving adversarial training with additional labeled or unlabeled data. While it is intuitive that additional labeled data can improve model performance, recent works have also demonstrated significant improvements in adversarial robustness with additional unlabeled data. For example, Carmon et al. (2019); Xing et al. (2021b) show that additional external unlabeled real data help improve adversarial robustness. Similarly, Gowal et al. (2021); Wang et al. (2023) use synthetic data to improve adversarial robustness and achieve 65% to 70% adversarial testing accuracy for the CIFAR-10 dataset under the benchmark AutoAttack (AA) in Croce et al. (2020)[1], achieving almost 20% improvement compared to earlier works.

To explain why external unlabeled data improve adversarial robustness, a recent study Xing et al. (2022) reveals that adversarial training gains greater benefits from unlabeled data than clean (natural) training. Based on Xing et al. (2022), adversarially robust models rely on the conditional distribution of the response given the features $(Y|X)$ and the marginal distribution of the features $(X)$. In contrast, clean training only depends on $Y|X$ in their study. Consequently, adversarial training can benefit more from unlabeled data than clean training.

Observing the benefit of unlabeled data, this paper aims to extend the existing analysis from simple to

[1] `https://robustbench.github.io`

more complex scenarios. In particular, we consider high-dimensional data and two-layer neural networks. For high-dimensional data, while Xing et al. (2022) considers large-sample regime with low-dimensional data, we study how the high data dimension affects the performance of using extra unlabeled data in adversarial training. In terms of neural networks in adversarial training, existing studies require either an uncommon initialization scheme (Xing et al., 2021a) or a vanishing attack strength (Gao et al., 2019; Zhang et al., 2020b; Allen-Zhu and Li, 2020).

Our contributions are summarized as follows:

First, as a preliminary, we extend the analysis of Hastie et al. (2019) (clean training) to the two-stage adversarial training framework of Xing et al. (2022) to use extra unlabeled data in adversarial training. Our theoretical comparisons reveal that utilizing additional unlabeled data improves the final robustness for large attack strengths ($\epsilon$). This observation holds true for the regression with no regularization and the $\mathcal{L}_2$-regularized regression. (Section 3)

Second, we extend our analysis to two-layer neural networks. We consider training a class of non-linear neural networks with lazy training. Unlike existing works which simplifies the analysis using vanishing initialization or small attack strength, we use a common non-vanishing initialization and assume a moderate attack strength. Our analysis reveals that the prediction performance of the two-layer neural network is almost the same as that of an adversarially robust linear regression with a specific ridge penalty. This result holds for both vanilla adversarial training and the two-stage method. While we mainly borrow the technical tools from Ba et al. (2020), to connect the neural network and linear model, there are some additional non-trivial steps to handle the adversarial attack. (Section 4)

In this paper, to distinguish between linear models and neural networks, we always specify "neural networks" to emphasize the results for neural networks.

## 2 Related Works

Below summarizes related works in adversarial training, high-dimensional statistics, and cross validation.

**Adversarial Training** There are many studies in the area of adversarial training. Methodology-wise, some studies improve the performance of adversarial training from optimization and statistics aspects, e.g., Goodfellow et al. (2014); Zhang et al. (2019); Wang et al. (2019b); Cai et al. (2018); Zhang et al. (2020a); Carmon et al. (2019); Gowal et al. (2021).

Theoretical investigations have also been conducted from different perspectives. For instance, Dan et al. (2020); Javanmard and Soltanolkotabi (2020) study the generalization when considering a binary classification task. Javanmard et al. (2020); Xing et al. (2021a) consider the generalization for the linear regression task. In particular, Xing et al. (2021a) investigates both the low-dimensional case (constant $d$) using the empirical process as the technical tool and the ultra-high dimensional case ($d \gg n_1$) using LASSO-related technical tools, and Javanmard and Montanari (2014) studies the high-dimensional case ($d \asymp n_1$) using technical tools of Thrampoulidis et al. (2015) from an optimization perspective. Carmon et al. (2019); Xing et al. (2022); Deng et al. (2021) work on the two-stage adversarial trianing framework for either regression or classification and all of them consider the low-dimensional case. Other studies can be found in Taheri et al. (2021); Yin et al. (2018); Raghunathan et al. (2019); Najafi et al. (2019); Min et al. (2020); Hendrycks et al. (2019); Wu et al. (2020) for the statistical properties of adversarial training, Sinha et al. (2018); Wang et al. (2019a); Xiao et al. (2022) for optimization, and Gao et al. (2019); Zhang et al. (2020b); Zhang and Li (2023); Mianjy and Arora (2022); Lv and Zhu (2021); Xiao et al. (2021) on deep learning.

**High-Dimensional Statistics** Double descent phenomenon is an observation in the learning curves of machine learning models as introduced in Section 1. This phenomenon has been observed in various machine learning settings. Comprehensive investigations into the double descent phenomenon can be found in Belkin et al. (2019); Hastie et al. (2019); Ba et al. (2020); d'Ascoli et al. (2020); Adlam and Pennington (2020); Liu et al. (2021); Rocks and Mehta (2022). Compared to the previous literature on the double descent phenomenon, which studies clean training, we take one step further to analyze the behavior of adversarial training in Section 3.

## 3 Preliminaries in Linear Regression

In this section, we introduce the assumptions and the two-stage method in linear regression scenario, and present the convergence of the two-stage adversarial training in the high-dimensional linear regression.

### 3.1 Data generation model

Denote $\|\cdot\| = \|\cdot\|_2$. Throughout this paper, we impose the following Gaussian assumption on $(X, Y)$:

**Definition 1.** *The attributes $X \sim N(\boldsymbol{0}, \boldsymbol{I}_d)$, and the response $Y = X^\top \boldsymbol{\theta}_0 + \varepsilon$ for some $\boldsymbol{\theta}_0$. The white noise $\varepsilon$ follows a zero-mean Gaussian distribution with $Var(\varepsilon) = \sigma^2$ for some constant $\sigma > 0$.*

Definition 1 is commonly used in the literature of high-dimensional regression, as evidenced by Belkin et al. (2019); Hastie et al. (2019); Ba et al. (2020). When the data dimension $d$ is comparable to the sample size, the sample covariance matrix is not consistent with the true covariance matrix but remains tractable. On the other hand, for the noise term, although it is possible to relax the assumption on the noise $\varepsilon$ for clean training, Javanmard et al. (2020); Xing et al. (2021b) demonstrate that constructing the best robust population model depends on the exact distribution of $\varepsilon$. Consequently, we follow Javanmard et al. (2020); Xing et al. (2021b) and assume $\varepsilon \sim N(0, \sigma^2)$.

Note that the notation $\varepsilon$ represents the noise in the response $Y$, and the notation $\epsilon$ in adversarial training is a different symbol and represents the attack strength.

In terms of the training data, we assume there are $n_1$ i.i.d. labeled samples, i.e., $(\boldsymbol{x}_i, y_i)$ for $i = 1, \ldots, n_1$, and $n_2$ i.i.d. unlabeled samples, i.e., $\boldsymbol{x}_i$ for $i = n_1 + 1, \ldots, n_1 + n_2$. We assume $d/n_1 \to \gamma$ asymptotically. Although Xing et al. (2022) provides a theoretical explanation for the benefits of unlabeled data in the large sample regime ($n_1 \gg d$), the asymptotic behavior of the two-stage method in other scenarios (e.g., $n_1 \asymp d$) remains unclear. On the other hand, since the unlabeled data are usually much cheaper than labeled data (Carmon et al., 2019), in this paper, we assume $n_2 \to \infty$ to simplify the derivation.

### 3.2 Formulations in Linear Regression

**Vanilla adversarial training** Given the labeled samples $(\boldsymbol{x}_i, y_i)$ for $i = 1, \ldots, n_1$, vanilla adversarial training with ridge penalty aims to minimize the penalized empirical adversarial loss as follows:

$$\frac{1}{n_1}\sum_{i=1}^{n_1} \sup_{\boldsymbol{z}\in\mathcal{B}_2(\boldsymbol{x}_i,\epsilon)} (\boldsymbol{z}^\top\boldsymbol{\theta} - y_i)^2 + \lambda\|\boldsymbol{\theta}\|^2 \tag{3.1}$$

with some $\lambda \geq 0$. Denote $\widehat{\boldsymbol{\theta}}_\epsilon(\lambda)$ as the solution.

Compared with clean training, the formulation in (3.1) uses the corrupted data attributes $\boldsymbol{z}$ instead of the original clean data $\boldsymbol{x}_i$. The corruption $\boldsymbol{z} - \boldsymbol{x}_i$ aims to maximize the loss value for the $i$th sample, and the perturbation is within a $\mathcal{L}_2$ ball with radius $\epsilon$, i.e., the attack strength is $\epsilon$.

Note that the above definition in (3.1) as well as (3.2) and (3.3) are under the context of linear regression. For the two-layer neural network, the exact formulations are postponed to Section 4.

**Two-stage adversarial training** The two-stage method is considered in some literature, e.g., Carmon et al. (2019); Gowal et al. (2021); Xing et al. (2022). There are two main stages in this framework. In the first stage, one solves the clean training problem

$$\frac{1}{n_1}\sum_{i=1}^{n_1}(\boldsymbol{x}_i^\top\boldsymbol{\theta} - y_i)^2 + \lambda\|\boldsymbol{\theta}\|^2 \tag{3.2}$$

and obtain the clean estimate $\widehat{\boldsymbol{\theta}}_0(\lambda)$.

In the second stage, with the set of extra unlabeled data $\boldsymbol{x}_i$ for $i = n_1+1, \ldots, n_1+n_2$, one uses the trained model $\widehat{\boldsymbol{\theta}}_0(\lambda)$ to generate pseudo response for them, i.e.,

$$\widehat{y}_i = \boldsymbol{x}_i^\top\widehat{\boldsymbol{\theta}}_0(\lambda) + \varepsilon_i.$$

The noise $\varepsilon_i$ is randomly generated from $N(0, \sigma^2)$ and in general we assume $\sigma^2$ is known. The final adversarially robust model is trained using the extra data with pseudo response, i.e., $(\boldsymbol{x}_i, \widehat{y}_i)$. We minimize the following adversarial loss

$$\frac{1}{n_2}\sum_{i=n_1+1}^{n_1+n_2} \sup_{\boldsymbol{z}\in\mathcal{B}_2(\boldsymbol{x}_i,\epsilon)} (\boldsymbol{z}^\top\boldsymbol{\theta} - \widehat{y}_i)^2. \tag{3.3}$$

Denote the solution of (3.3) as $\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)$. We also supplement a simulation with a ridge penalty injecting in both stages in Appendix B, in which the best performance is achieved when no penalty is injected in the second stage. As a result, we only consider injecting a penalty in the first stage.

**Expected Adversarial Risk** Under Definition 1, denoting the population adversarial risk for any given estimate $\boldsymbol{\theta}$ as

$$R_\epsilon(\boldsymbol{\theta}, \boldsymbol{\theta}_0) = \mathbb{E}_{(\boldsymbol{x},y)} \sup_{\boldsymbol{z}\in\mathcal{B}_2(\boldsymbol{x},\epsilon)} (y - \boldsymbol{z}^\top\boldsymbol{\theta})^2,$$

then

$$\begin{aligned} R_\epsilon(\boldsymbol{\theta}, \boldsymbol{\theta}_0) \quad = \quad & \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2 + \sigma^2 \\ & + 2c_0\epsilon\|\boldsymbol{\theta}\|\sqrt{\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2 + \sigma^2} + \epsilon^2\|\boldsymbol{\theta}\|^2, \end{aligned} \tag{3.4}$$

where $\|\cdot\|$ is the $\mathcal{L}_2$ norm, and $c_0 = \sqrt{2/\pi}$ is derived from the exact distribution of $(X, Y)$. We rewrite $R_\epsilon(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$ as $R_\epsilon(\boldsymbol{\theta})$ for simplicity when no confusion arises. Denote $\boldsymbol{\theta}_\epsilon$ as the minimizer of $R_\epsilon(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$. Throughout this paper, we use $R_\epsilon(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$ to compare different methods, i.e., we measure $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|$ and $\|\boldsymbol{\theta}\|$.

### 3.3 Convergence Results

To study $\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)$, we follow Hastie et al. (2019) to denote the following function

$$m_\gamma(-\lambda) = \frac{-(1-\gamma+\lambda) + \sqrt{(1-\gamma+\lambda)^2 + 4\lambda\gamma}}{2\gamma\lambda},$$

which is used to describe the asymptotic behavior of $tr\left(\left(\sum_{i=1}^{n_1}\boldsymbol{x}_i\boldsymbol{x}_i^\top+\lambda\boldsymbol{I}_d\right)^{-1}\right)$. Also denote $\boldsymbol{m}'_\gamma(-\lambda)$ as the derivative w.r.t. $-\lambda$.

Together with the definition of $m_\gamma$, the proposition below presents the convergence of the two-stage method:

**Proposition 1** (Convergence of Two-Stage Adversarial Training)**.** *Under Definition 1, and further assuming that $\widehat{\theta}_0 \sim N(\boldsymbol{0}, r^2\boldsymbol{I}_d/d)$, when $d/n_1 \to \gamma$ asymptotically, in the first stage, the clean estimate $\widehat{\boldsymbol{\theta}}_0(\lambda)$ satisfies that, almost surely,*

$$\begin{aligned}\|\widehat{\boldsymbol{\theta}}_0(\lambda)-\boldsymbol{\theta}_0\|^2 \quad\to\quad & \lambda^2r^2m'_\gamma(-\lambda) \\ & +\sigma^2\gamma\left(m_\gamma(-\lambda)-\lambda m'_\gamma(-\lambda)\right),\end{aligned} \tag{3.5}$$

*and*

$$\begin{aligned}\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2 \quad\to\quad & r^2[1-2\lambda m_\gamma(-\lambda)+\lambda^2m'_\gamma(-\lambda)] \\ & +\sigma^2\gamma[m_\gamma(-\lambda)-\lambda m'_\gamma(-\lambda)].\end{aligned} \tag{3.6}$$

*In the second stage, when $n_2 \to \infty$, almost surely, $\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)$ satisfies*

$$\begin{aligned}\|\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)-\boldsymbol{\theta}_0\|_2^2 \quad\to\quad & \frac{1}{(1+\alpha_\epsilon(\lambda))^2}\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|_2^2+r^2 \\ & -\frac{2}{(1+\alpha_\epsilon(\lambda))}\widehat{\boldsymbol{\theta}}_0(\lambda)^\top\boldsymbol{\theta}_0,\end{aligned}$$

*and*

$$\|\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)\|_2^2 \quad\to\quad \frac{1}{(1+\alpha_\epsilon(\lambda))^2}\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|_2^2,$$

*where $\alpha_\epsilon(\lambda)$ is the solution of $\alpha$ in*

$$\begin{aligned}&\alpha+\epsilon c_0\frac{\alpha\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|}{\sqrt{\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2\alpha^2+\sigma^2(1+\alpha)^2}} \\ =\;& \epsilon c_0\frac{\sqrt{\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2\alpha^2+\sigma^2(1+\alpha)^2}}{\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|}+\epsilon^2,\end{aligned}$$

*where the convergence of $\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|$ and $\widehat{\boldsymbol{\theta}}_0(\lambda)^\top\boldsymbol{\theta}_0$ are from (3.5) and (3.6).*

The proof of Proposition 1 is in Appendix C.1. In short, we first study the convergence of the first-stage estimate $\widehat{\boldsymbol{\theta}}_0(\lambda)$, and then evaluate the second-stage estimate $\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)$. The proof is mainly extended from the results in Hastie et al. (2019). The main difference is that we further utilize $\widetilde{\theta}_0(\lambda)$ to derive $\widetilde{\theta}_\epsilon(\lambda)$, i.e., starting from "In the second stage" in Appendix C.1.

From Proposition 1, similar to $\widehat{\boldsymbol{\theta}}_0$, one can see that $\|\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)-\boldsymbol{\theta}_0\|_2^2$ and $\|\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)\|_2^2$ converges to some constant value. As will be shown in Figure 1, the two-stage method can sometimes out-perform the vanilla adversarial training. One may use the theoretical values to determine the method to be used in their applications. In general, when $d$ gets larger, the two-stage estimate can learn more from the marginal distribution of $X$, thus gets a better performance.

### 3.4 Simulation

We conduct a simulation to verify Proposition 1 and compare the risk of the vanilla adversarial training and the two-stage method. In the experiment, we take $n_1 = 100$ and $n_2 = \infty$. For $n_2 = \infty$, we use the population adversarial risk in the second stage. We change the data dimension $d$ to obtain different $\gamma = d/n_1$. The data follows $X \sim N(\boldsymbol{0}, \boldsymbol{I}_d)$, $Y = X^\top\boldsymbol{\theta}_0 + \varepsilon$ with $\boldsymbol{\theta}_0 \sim N(0, \boldsymbol{I}_d/d)$ and $\varepsilon \sim N(0, 1)$. The adversarial attack is taken as $\epsilon = 0.3$. We repeat the experiment for 100 times to obtain the average performance. We use the excess adversarial risk, i.e., $R_\epsilon(\boldsymbol{\theta}) - R_\epsilon(\boldsymbol{\theta}_\epsilon)$ for $\boldsymbol{\theta} \in \{\widehat{\boldsymbol{\theta}}_0(\lambda), \widehat{\boldsymbol{\theta}}_\epsilon(\lambda), \widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)\}$, to compare the performance of the three methods. The results are in Figure 1, 2.

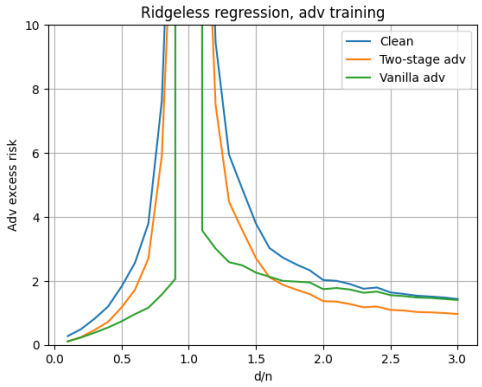


Figure 1: Simulation: Excess adversarial risk of clean training, vanilla adversarial training, and the two-stage adversarial training, without ridge penalty.

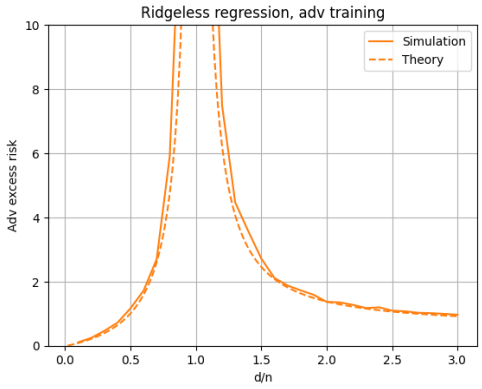


Figure 2: Theoretical value vs simulation results for the two-stage method.

There are several observations from Figure 1. First, if we compare the performance of the two-stage adver-

sarial training and the clean training, the two-stage adversarial training is better than clean training because it is tailored for the adversarial attack. Second, when $d/n_1$ gets larger, the performance of the two-stage adversarial training is better than the vanilla adversarial training, indicating that the information of the additional extra unlabeled data matters. Finally, for all the three methods, they all observe a double-descent phenomenon, i.e., the corresponding testing loss drops when further increasing $d/n$ from 1.

In addition, we compare the theoretical value of the two-stage method against the simulation result. From Figure 2, the curves of the theoretical results and the simulation outcome are almost the same. This verifies the correctness of Proposition 1.

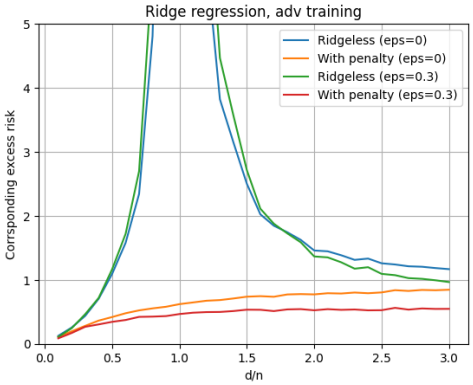


Figure 3: Simulation: Ridgeless regression and ridge regression with the best penalty in clean training and the two-stage adversarial training. The two-stage method benefits from from a proper penalty.

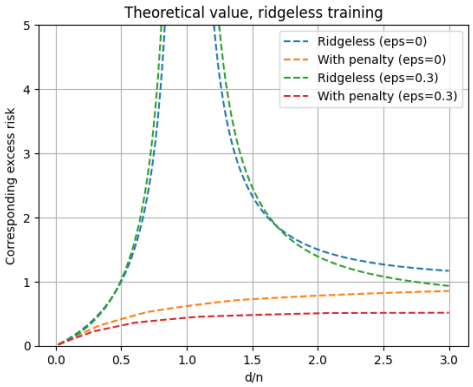


Figure 4: Theoretical value corresponding to Figure 3.

Finally, we examine how the ridge penalty affects the performance. In the simulation in Figure 3, we take $\epsilon = 0, 0.3$ and compare the performance when $\lambda = 0$ and $\lambda$ is taken to minimize the risk. In Figure 3, the y-axis is the corresponding excess adversarial risk, i.e., $\epsilon = 0, 0.3$ for the corresponding groups. The corresponding theoretical curves can be found in Figure 4.

From Figure 3, one can see that the excess risk for the ridgeless regression is similar for $\epsilon = 0, 0.3$, while $\epsilon = 0.3$ benefits more than $\epsilon = 0$ when taking a proper ridge penalty.

## 4 Two-Layer Neural Networks

In this section, we shift our focus towards the implementation of adversarial training using two-layer neural networks. Section 4.1 presents the formulation of the two-layer neural network and how to train the neural network. Since the theoretical understanding of how neural networks works in vanilla adversarial training is not fully developed, in Section 4.2, we discuss about the convergence of vanilla adversarial training using two-layer neural networks. In Section 4.3, we further discuss about utilizing neural networks in the two-stage training method.

### 4.1 Notation and Preliminary Results

This section introduces the mathematical formulation of two-layer neural networks and various notations and definitions for later discussions.

**Two-layer neural networks** We consider a two-layer neural network to learn the relationship between $X$ and $Y$. In this scenario, we assume $(X, Y)$ follows the linear relationship in Definition 1, and use a neural network to learn from the data, and predict $Y$ given $X$. Given an input $\boldsymbol{x}$, the neural network outputs the following prediction:

$$f_{\boldsymbol{W}}(\boldsymbol{x}) = \frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h \phi(\boldsymbol{x}^\top \boldsymbol{w}_h),$$

where $\phi$ is the activation function and $a_h \in \mathbb{R}$. In this paper, we consider $H \gg d^2$ and use the following variant of the Sigmoid activation function:

$$\phi(t) = 2\left(\frac{1}{1+\exp(t)} - \frac{1}{2}\right).$$

Besides $\phi$, the notation $\boldsymbol{W} \in \mathbb{R}^{d\times H}$ is the weight matrix for the whole hidden layer, $\boldsymbol{w}_h \in \mathbb{R}^d$ is the $h$th column of $\boldsymbol{W}$ and is the weight for the hidden node $h$.

**How to train the neural network** Unlike the previous sections where one can always solve the exact numerical solution of $\widehat{\boldsymbol{\theta}}$ in the linear model, in the neural networks, the convergence of the neural networks depends on the initialization and the optimization.

We rewrite $\boldsymbol{w}_h$ and $\boldsymbol{W}$ as $\boldsymbol{w}_h(t)$ and $\boldsymbol{W}(t)$ to represent the neural network parameters during the training. The notation $t$ represents the current iteration and $t = 1, \ldots, T$. For the initialization, we take $t = 0$.

The training configurations considered are as follows:

**C1** Non-vanishing initialization scheme: The initial weight $\boldsymbol{w}_h(0) \sim N(0, \boldsymbol{I}_d/d)$ for $h = 1, \ldots, H/2$ so that $\|\boldsymbol{w}_h(0)\| = O_p(1)$. In addition, we consider a symmetric initialization scheme, i.e., $\boldsymbol{w}_h(0) = -\boldsymbol{w}_{H-h}(0)$ for $h = H/2+1, \ldots, H$ so that $f_{\boldsymbol{W}(0)}(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$. We take $a_h = 1$ for all $h = 1, \ldots, H$.

**C2** Gradient descent: We use gradient descent to iteratively update the neural network. In vanilla adversarial training, in iteration $t = 1, \ldots, T$, we calculate the gradient of

$$\frac{1}{n_1}\sum_{i=1}^{n_1} \sup_{\boldsymbol{z}\in\mathcal{B}_2(\boldsymbol{x}_i,\epsilon)} (f_{\boldsymbol{W}(t)}(\boldsymbol{z}) - y_i)^2 + \lambda\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F^2$$

w.r.t. $\boldsymbol{W}(t)$ and denote it as $\Delta\boldsymbol{W}(t)$. The weight is then updated as $\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \eta\Delta\boldsymbol{W}(t)$. The learning rate is denoted as $\eta$ and $\eta = \Theta(1)$. The total number of iterations is in $\Theta(\log n_1)$.

**C3** Lazy training: We use lazy training to train the neural network, i.e., we only train the first layer (the hidden layer), and do not update the second layer after the initialization.

**C4** Fast gradient method (FGM) attack: Instead of $\arg\max_{\boldsymbol{z}\in\mathcal{B}_2(\boldsymbol{x}_i,\epsilon)}(f_{\boldsymbol{W}(t)}(\boldsymbol{z}) - y_i)^2$ (the strongest attack), we use a weaker but simpler attack. The FGM attack is defined as

$$\boldsymbol{z} = \boldsymbol{x}_i + \epsilon\frac{\delta}{\|\delta\|_2}, \text{ where } \delta = \frac{\partial(f_{\boldsymbol{W}(t)}(\boldsymbol{x}_i) - y_i)^2}{\partial \boldsymbol{x}_i}.$$

We take $\boldsymbol{z} = \boldsymbol{x}_i$ if $\|\delta\|_2 = 0$.

In existing literature, there are some studies working in a similar list of configurations. For example, Ba et al. (2020) studies the convergence of clean training in two-layer neural networks under high-dimensional data using either vanishing or non-vanishing initialization. Another work Xing et al. (2021a) studies the convergence of adversarial training in two-layer neural networks with vanishing initialization. For others, they study adversarial training with a vanishing attack strength, i.e., $\epsilon = o(1)$ or an even smaller $\epsilon$ (Allen-Zhu and Li, 2020; Gao et al., 2019; Zhang et al., 2020b). Condition **C1**, **C3** are the same as the previous literature. Condition **C4** uses a fast-gradient method as Allen-Zhu and Li (2020) to simplify the derivation.

Compared to the above literature, the main difference in this paper is **C2**. First, without a regularization term, the attack cannot be correctly calculated in adversarial training due to over-fitting. On the other hand, the penalty term $\|\boldsymbol{W} - \boldsymbol{W}(0)\|_F^2$ has a slight difference than the traditional $\mathcal{L}_2$ regularization. Based on our initialization, the initial weight satisfies $\|\boldsymbol{W}(0)\|_F^2 = \Theta(H)$. However, as in Ba et al. (2020), the increment of $\boldsymbol{w}_h$ is of only $O(1/\sqrt{H})$, i.e., $\|\boldsymbol{W} - \boldsymbol{W}(0)\|_F^2 = \Theta(1)$. If we add a penalty term $\|\boldsymbol{W}\|_F^2$, the penalty will force the model parameters to shift away from the initialization, which deviates from the initialization, and cannot borrow the idea of the neural tangent kernel (NTK). Second, in terms of the learning rate $\eta$ and iteration $T$ in **C2**, since we use probability bounds to quantify the attacks, we need a small $T$ to control the overall tail probability. And we need $\eta T \to \infty$ to ensure the convergence.

For the clean training stage, we follow Ba et al. (2020) and obtain the following result:

**Proposition 2** (Ba et al. (2020))**.** *Denote* $g \sim N(0,1)$*, and*

$$b_0 = \mathbb{E}\phi'(g),\ b_1^2 = \mathbb{E}\phi'(g)^2 - b_0^2.$$

*Under configuration **C1** to **C3**, with probability tending to 1 over the randomness of the initialization and the training data, the neural network performs approximately the same as a ridge regression with a ridge penalty of scale* $(\lambda + b_1^2 d/n_1)/b_0^2$.

To mathematically define "approximately the same as" in the above proposition, we say the prediction performance of the neural network "approximately the same as" the linear regression if $R_\epsilon(f_{\boldsymbol{W}}, \boldsymbol{\theta}_0) = (1 + o(1))R_\epsilon(\widehat{\boldsymbol{\theta}}_\epsilon((\lambda + b_1^2 d/n_1)/b_0^2), \boldsymbol{\theta}_0)$.

### 4.2 Convergence and Generalization of Vanilla Adversarial Training

Our results in the convergence and generalization performance of adversarial training in two-layer neural networks are as follows:

**Theorem 1.** *With probability tending to 1 over the randomness of the initialization and the training data, the convergence and generalization performance of the two-layer neural network under **C1** to **C4** is approximately the same as an adversarially robust linear regression model with a ridge penalty of scale* $(\lambda + b_1^2 d/n_1)/b_0^2$ *when* $\lambda = \Theta(1)$.

While the final result in Theorem 1 is simple and is similar to Proposition 2, there are many extra steps when proving the above theorem. The proof can be found in Appendix D and we show some key observations below:

**O1** During the training, when the number of hidden nodes $H$ is large enough, there is only a $O(1/\sqrt{H})$ change in $\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F$ in the training.

**O2** The attack for each sample during the training is "similar" to each other. Denote $\boldsymbol{\delta}_i(t)$ as the attack for the $i$th sample in $t$th iteration, then $|\cos(\boldsymbol{\delta}_i(t), \boldsymbol{\delta}_j(t))|$ is $1 - o(1)$ (either the similar or reversed direction).

The observation **O1** is intuitive and is similar to its clean training counterpart. The implication of **O2** is that, although the attacks on neural networks and linear models have different formulas, the attacks for all samples still share the same particular pattern. Observations **O1** and **O2** together imply that one may reduce the neural network optimization into a neural tangent kernel (NTK) problem: First, with **O1**, one can use Taylor expansion to approximate the neural network using NTK. Second, **O2** guarantees that the approximation of $f_{\boldsymbol{W}}$ gives a proper approximation on the attack as well.

**Simulation Study** We conduct simulation to check **O1** and **O2**, and whether the adversarially robust linear regression with $\mathcal{L}_2$ regularization of $(\lambda + b_1^2 d/n_1)/b_0^2$ gives similar adversarial training and testing performance as training neural network using adversarial training with a $\mathcal{L}_2$ regularization of $\lambda$ (**O3**).

We generated data from $X \sim N(0, \boldsymbol{I}_d)$, $Y = \boldsymbol{\theta}_0^\top X + \varepsilon$ where $\boldsymbol{\theta}_0 = \mathbf{1}/\sqrt{d}$ and $Var(\varepsilon) = 1$. The sample size is $n_1 = 50$ and $\epsilon = 0.3$ and repeat the experiment 50 times. We train the non-linear and linear networks both for 200 iterations. The learning rate for the Sigmoid network is 0.01 and is 0.05 for the linear network.

To examine **O1**, we calculate the ratio of $\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F / \|\boldsymbol{W}(0)\|_F$ through out the training.

We fix the size of the neural network $H = 10000$ and change the value of the data dimension $d$. The results of $\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F / \|\boldsymbol{W}(0)\|_F$ are in Figure 5. From Figure 5, one can see that for different $d$, the change in the weight of the neural network is small.

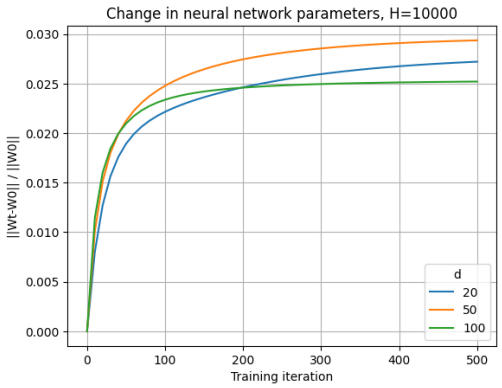


Figure 5: How far $\boldsymbol{W}(t)$ moves away from $\boldsymbol{W}(0)$ when the data dimension $d$ changes. For all $d$, $\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F / \|\boldsymbol{W}(0)\|_F$ is small.

To check **O2**, we conduct the following experiment. We take $d = 100, H = 10000, \lambda = 0.05$. For linear model, we know that all samples share the same (or reverse) attack direction. Denoting $A_\epsilon(\boldsymbol{x}, y, f)$ as the attack for the sample $(\boldsymbol{x}, y)$ given a model $f$, then in the linear regression setup, $A_\epsilon(\boldsymbol{x}_i, y_i, f)$ are parallel for all $i$s. Therefore, we calculate the following:

- For each pair of samples $((\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j))$, we calculate the correlation of the two attacks, i.e., $cor(A_\epsilon(\boldsymbol{x}_i, y_i, f), A_\epsilon(\boldsymbol{x}_j, y_j, f))$. There are total $n_1(n_1 - 1)/2$ and $1000 * 999/2$ pairs of $((\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j))$ respectively in the training and testing data, and we get a *cor* value for each pair. (Correlation for samples)

- For each pair of coordinates $(X_i, X_j)$, we calculate the correlation of the attack on these two coordinates, i.e., $cor([A_\epsilon(\boldsymbol{X}, \boldsymbol{y}, f)]_i, [A_\epsilon(\boldsymbol{X}, \boldsymbol{y}, f)]_j)$. There are $d(d-1)/2$ pairs of features, and we get a *cor* value for each pair. (Correlation for features)

| Data set | Correlation | P($\|cor\| > 0.99$) | P($\|cor\| > 0.95$) |
|---|---|---|---|
| Train | Samples | 1.0000 | 1.0000 |
| Train | Features | 0.8586 | 0.9293 |
| Test | Samples | 0.9899 | 1.0000 |
| Test | Features | 0.8586 | 0.9192 |

Table 1: Correlation of the attacks.

The results are summarized in Table 1. From Table 1, one can see that the absolute value of the correlation for both samples and features are almost around 1. This is a natural outcome if all the attacks are almost parallel with each other.

To verify **O3**, we conduct the following experiment. We run the adversarial training on a non-linear neural network and a linear neural network. Based on Xing et al. (2021a), the linear neural network is equivalent to a linear model. In the experiment, our choice of activation function gives $b_0^2 \approx 0.2$ and $b_1^2/b_0^2 \approx 0.05$.

Theorem 1 indicates that the adversarial training with non-linear activation function with a penalty $\lambda\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F^2$ gives a similar result as an adversarially robust linear regression with $\mathcal{L}_2$ penalty $(\lambda + b_1^2 d/n_1)\|\boldsymbol{\theta}\|^2/b_0^2$, which is also the same as a linear network with a penalty $[(\lambda + b_1^2 d/n_1)/b_0^2]\|\boldsymbol{W}(t) - \boldsymbol{W}(0)\|_F^2$.

The results for the adversarial training and testing performance are summarized in Figure 6. One can see that with different choices of $d$, the non-linear network (solid line) and linear network (dashed line) always give a similar performance. When $d = 100$, the two robust testing loss differ a lot because the $\lambda$ is relatively small compared to the large data dimension. When $\lambda$ gets larger, the two networks behave similarly.

**Remark 1.** *To clarify, since there is no existing result about adversarial training in neural networks in the high-dimensional scenario, Theorem 1 aims to explain how adversarial training works in this case and is unrelated to the two-stage method. The later Section 4.3 considers the two-stage method.*

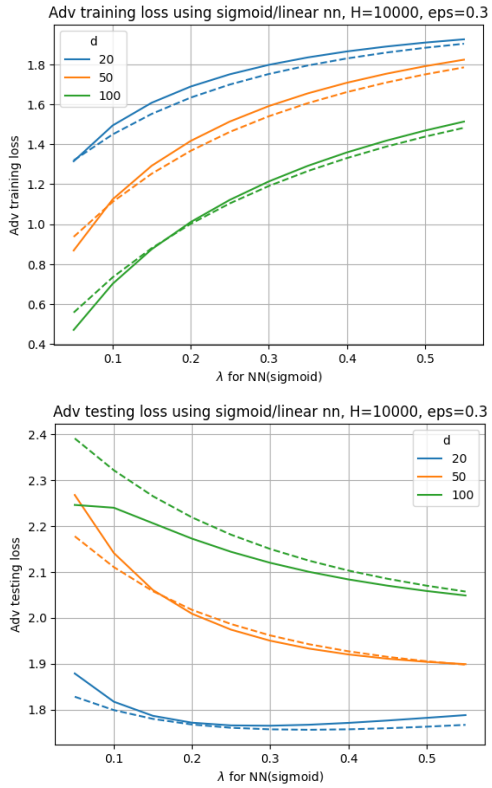


Figure 6: Adversarial training (top, including regularization) and testing (bottm) loss of non-linear and linear neural networks. Solid line: non-linear network. Dashed line: linear network. The X axis shows the $\lambda$ used in the non-linear network ($\lambda\|\boldsymbol{W}(t)-\boldsymbol{W}(0)\|_F^2$). For the dashed lines (linear networks), the penalty is $[(\lambda+b_1^2 d/n_1)/b_0^2]\|\boldsymbol{W}(t)-\boldsymbol{W}(0)\|_F^2$. $n_1 = 50$.

### 4.3 Two-Stage Adversarial Training

For the two-stage method, we consider utilizing either linear model or a two-layer neural network for the first stage of the clean training. In the second stage, we use a two-layer neural network for the adversarial training.

The convergence of the two-stage method is as follows and is similar to Theorem 1:

**Theorem 2.** *With probability tending to 1 over the randomness of the initialization and the training data, the convergence and generalization performance of the neural network in the two-stage adversarial training is approximately the same as an adversarially robust linear regression model with a ridge penalty of $\lambda/b_0^2$.*

The statement of Theorem 2 is almost the same as Theorem 1 while $d \asymp n_1$ but $n_2 = \infty$. Such a discrepancy leads to a change in the final penalty term and some technical changes in the proof.

Similar to Section 4.2, we conduct a simulation study to verify Theorem 2. To verify Theorem 2, instead of utilizing the two-stage method, it suffices to verify that the adversarial training and testing performance are similar for linear and Sigmoid networks. The settings are the same as Section 4.2, and we take $n_2 = 10000$ and train for 10 iterations. From the left panel of Figure 7, the adversarial training loss is the same for the two networks under different $d$. From the right panel of Figure 7, the two testing losses in each $d$ have only around 2% difference.

**Remark 2.** *To compare the technical challenges, the additional technical challenge of Theorem 2 than Theorem 1 is on the relationship between $d$ and $n_2$ ($d \ll n_2$). In Theorem 1, we directly follow the idea of Ba et al. (2020) since $d \asymp n_1$. For Theorem 2, we make changes in the technical steps to handle $d \ll n_2$.*

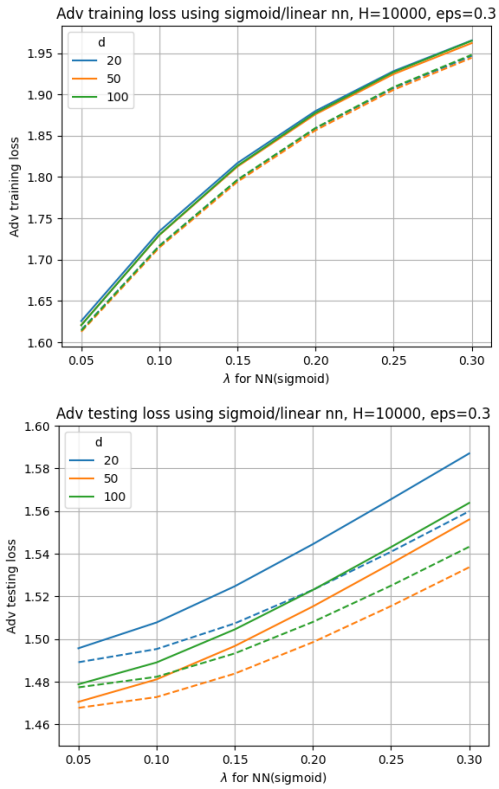


Figure 7: Adversarial training (top, including regularization) and testing (bottom) loss of non-linear and linear neural networks. Solid line: non-linear network. Dashed line: linear network. $n_2 = 10000$.

## 5 Conclusion

In this paper, we extend the analysis of the two-stage method in adversarial training from a simple large-sample scenario to high-dimensional linear regression and two-layer neural networks. Our result indicates that the vanilla high-dimensional adversarial training in two-layer neural networks in fact is similar to the vanilla adversarial training with a ridge penalty. Our insights can further deepen the understanding on how the attack affects the training in neural networks. We provide discussions for ReLU in Appendix E for future exploration.

## References


Adlam, B. and Pennington, J. (2020), "Understanding double descent requires a fine-grained bias-variance decomposition," *Advances in neural information processing systems*, 33, 11022–11032.

Allen-Zhu, Z. and Li, Y. (2020), "Feature Purification: How Adversarial Training Performs Robust Deep Learning," *arXiv preprint arXiv:2005.10190*.

Ba, J., Erdogdu, M., Suzuki, T., Wu, D., and Zhang, T. (2020), "Generalization of two-layer neural networks: an asymptotic viewpoint," in *8th International Conference on Learning Representations*.

Belkin, M., Hsu, D., and Xu, J. (2019), "Two models of double descent for weak features," *arXiv preprint arXiv:1903.07571*.

Cai, Q.-Z., Du, M., Liu, C., and Song, D. (2018), "Curriculum adversarial training," *arXiv preprint arXiv:1805.04807*.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. (2019), "Unlabeled data improves adversarial robustness," in *Advances in Neural Information Processing Systems*, pp. 11192–11203.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. (2020), "RobustBench: a standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*.

Dan, C., Wei, Y., and Ravikumar, P. (2020), "Sharp Statistical Guaratees for Adversarially Robust Gaussian Classification," in *International Conference on Machine Learning*, PMLR, pp. 2345–2355.

Deng, Z., Zhang, L., Ghorbani, A., and Zou, J. (2021), "Improving adversarial robustness via unlabeled out-of-domain data," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 2845–2853.

Dobriban, E. and Wager, S. (2018), "High-dimensional asymptotics of prediction: Ridge regression and classification," *The Annals of Statistics*, 46, 247–279.

d'Ascoli, S., Refinetti, M., Biroli, G., and Krzakala, F. (2020), "Double trouble in double descent: Bias and variance (s) in the lazy regime," in *International Conference on Machine Learning*, PMLR, pp. 2280–2290.

Gao, R., Cai, T., Li, H., Wang, L., Hsieh, C.-J., and Lee, J. D. (2019), "Convergence of adversarial training in overparametrized networks," *arXiv preprint arXiv:1906.07916*.

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

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. (2021), "Improving Robustness using Generated Data," *Advances in Neural Information Processing Systems*, 34.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. (2019), "Surprises in high-dimensional ridgeless least squares interpolation," *arXiv preprint arXiv:1903.08560*.

Hendrycks, D., Lee, K., and Mazeika, M. (2019), "Using pre-training can improve model robustness and uncertainty," *arXiv preprint arXiv:1901.09960*.

Javanmard, A. and Montanari, A. (2014), "Confidence Intervals and Hypothesis Testing for High-Dimensional Regression," *The Journal of Machine Learning Research*, 15, 2869–2909.

Javanmard, A. and Soltanolkotabi, M. (2020), "Precise statistical analysis of classification accuracies for adversarial training," *arXiv preprint arXiv:2010.11213*.

Javanmard, A., Soltanolkotabi, M., and Hassani, H. (2020), "Precise tradeoffs in adversarial training for linear regression," in *Conference on Learning Theory*, PMLR, pp. 2034–2078.

Liu, F., Liao, Z., and Suykens, J. (2021), "Kernel regression in high dimensions: Refined analysis beyond double descent," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 649–657.

Lv, B. and Zhu, Z. (2021), "Implicit Bias of Adversarial Training for Deep Neural Networks," in *International Conference on Learning Representations*.

Mianjy, P. and Arora, R. (2022), "Robustness Guarantees for Adversarially Trained Neural Networks," .

Min, Y., Chen, L., and Karbasi, A. (2020), "The curious case of adversarially robust models: More data

can help, double descend, or hurt generalization," *arXiv preprint arXiv:2002.11080*.

Najafi, A., Maeda, S.-i., Koyama, M., and Miyato, T. (2019), "Robustness to adversarial perturbations in learning from incomplete data," in *Advances in Neural Information Processing Systems*, pp. 5542–5552.

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., and Liang, P. (2019), "Adversarial training can hurt generalization," *arXiv preprint arXiv:1906.06032*.

Rocks, J. W. and Mehta, P. (2022), "Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models," *Physical Review Research*, 4, 013201.

Rodriguez Rubio, F. and Mestre, X. (2011), "Spectral convergence for a general class of random matrices," *Statistics & Probability Letters, 81 (5), 592-602.*

Sinha, A., Namkoong, H., and Duchi, J. (2018), "Certifying some distributional robustness with principled adversarial training," .

Taheri, M., Xie, F., and Lederer, J. (2021), "Statistical guarantees for regularized neural networks," *Neural Networks*, 142, 148–161.

Thrampoulidis, C., Oymak, S., and Hassibi, B. (2015), "Regularized linear regression: A precise analysis of the estimation error," in *Conference on Learning Theory*, PMLR, pp. 1683–1709.

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. (2019a), "On the convergence and robustness of adversarial training," in *International Conference on Machine Learning*, pp. 6586–6595.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. (2019b), "Improving adversarial robustness requires revisiting misclassified examples," in *International Conference on Learning Representations*.

Wang, Z., Pang, T., Du, C., Lin, M., Liu, W., and Yan, S. (2023), "Better diffusion models further improve adversarial training," *arXiv preprint arXiv:2302.04638*.

Wu, D., Wang, Y., and Xia, S.-t. (2020), "Adversarial Weight Perturbation Helps Robust Generalization," *arXiv preprint arXiv:2004.05884*.

Xiao, J., Fan, Y., Sun, R., and Luo, Z.-Q. (2021), "Adversarial Rademacher Complexity of Deep Neural Networks," .

Xiao, J., Fan, Y., Sun, R., Wang, J., and Luo, Z.-Q. (2022), "Stability analysis and generalization bounds of adversarial training," *arXiv preprint arXiv:2210.00960*.

Xing, Y., Song, Q., and Cheng, G. (2021a), "On the generalization properties of adversarial training," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 505–513.

— (2022), "Why Do Artificially Generated Data Help Adversarial Robustness," *Neurips*.

Xing, Y., Zhang, R., and Cheng, G. (2021b), "Adversarially Robust Estimate and Risk Analysis in Linear Regression," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 514–522.

Yin, D., Ramchandran, K., and Bartlett, P. (2018), "Rademacher complexity for adversarially robust generalization," *arXiv preprint arXiv:1810.11914*.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. (2019), "Theoretically Principled Trade-off between Robustness and Accuracy," in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, vol. 97 of *Proceedings of Machine Learning Research*, pp. 7472–7482.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. (2020a), "Attacks which do not kill training make adversarial learning stronger," in *International Conference on Machine Learning*, PMLR, pp. 11278–11287.

Zhang, T. and Li, K. (2023), "Understanding Overfitting in Adversarial Training in Kernel Regression," *arXiv preprint arXiv:2304.06326*.

Zhang, Y., Plevrakis, O., Du, S. S., Li, X., Song, Z., and Arora, S. (2020b), "Over-parameterized Adversarial Training: An Analysis Overcoming the Curse of Dimensionality," *arXiv preprint arXiv:2002.06668*.

## Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]

(c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] We use simple simulations to verifies the theories in the paper.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [No] We use simple simulations to verifies the theories in the paper.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No] We repeat the experiments for multiple times and the empirical curves are almost the same as the theoretical curves.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No] The simulations do not cost much computational resources.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

The structure of the appendix is as follows. Section A provides some additional simulation when injecting a ridge penalty for both the stages in the two-stage method. In Section B, we list some existing results in Hastie et al. (2019) which will be used in our results. Section C contains the proof for the two-stage method in the high-dimensional linear regression scenario corresponding to Section 3. In Section D, we demonstrate the proof for the convergence of the two-layer neural networks in vanilla adversarial training and the two-stage method corresponding to Section 4. Section E provides discussions in ReLU networks, and Section F provides some additional experiment results.

# A Preliminary Simulation

In Section 3, when introducing the two-stage method, it is mentioned that injecting a ridge penalty in the second stage does not improve the performance. This section will provide a detailed simulation to illustrate this.

In the simulation, we take $d = 50$, $n_1 = 100$, $\sigma^2 = 1$, and $\epsilon = 0.1$. In each repetition, we randomly generate $\boldsymbol{\theta}_0 \sim N(\mathbf{0}, \boldsymbol{I}_d/d)$. We take the $\mathcal{L}_2$ penalty with $\lambda_1, \lambda_2 \in \{0, 0.1, 0.2, 0.5, 1, 2, 5\}$ for the two stages respectively, and report smallest adversarial excess risk for each $\lambda_1$. The results are summarized in Table A. From Table A, one can see that when achieving the minimal adversarial excess risk is achieved, the corresponding $\lambda_1 = 0.5$, while the corresponding $\lambda_2 = 0$.

| $\lambda_1$ | Best $\lambda_2$ | Minimal adv excess risk |
|---|---|---|
| 0.0 | 1.0 | 0.4668 |
| 0.1 | 0.5 | 0.4342 |
| 0.2 | 0.5 | 0.4256 |
| 0.5 | 0.0 | **0.4120** |
| 1.0 | 0.0 | 0.4534 |
| 2.0 | 0.0 | 0.5548 |
| 5.0 | 0.0 | 0.7140 |

Table 2: Model performance when injecting ridge penalty for both stages.

# B Preliminary Results from Existing Literature

This section lists some existing results to be used in our derivation.

Based on Hastie et al. (2019), Theorem 1 of Rodriguez Rubio and Mestre (2011) implies the following result:

**Proposition 3** (A Generalized Marchenko-Pastur Theorem)**.** *For any $z > 0$, and any deterministic sequence of matrices $\Theta_n \in \mathbb{R}^{d\times d}$, $n = 1, \ldots$ with uniformly bounded trace norm, it holds as $n, d \to \infty$, almost surely,*

$$tr\left(\Theta_n\left((\widehat{\Sigma} + z\boldsymbol{I}_d)^{-1} - c_n(z)\boldsymbol{I}_d\right)\right) \to 0,$$

*for a deterministic sequence $c_n(z) > 0$, $n = 1, 2, 3, \ldots$.*

*In addition, $c_n(z) = m_\gamma(-z)$.*

In addition, Lemma C.3 in Dobriban and Wager (2018) shows the following result:

**Proposition 4.** *Assume $\delta \in \mathbb{R}^n$ has independent Gaussian entries with zero mean and variance $c/n$. Moreover, let $A_d$ be a sequence of random $d \times d$ matrices independent of $\delta$, with uniformly bounded eigenvalues, then almost surely,*

$$\delta^\top A_d \delta \to c \times tr(A_d).$$

# C Proofs for High-Dimensional Linear Regression

## C.1 Two-Stage Adversarial Training

*Proof of Proposition 1.* Denote $\boldsymbol{X}$ and $\boldsymbol{y}$ as the data matrix and the corresponding response for all the $n_1$ labeled data, and $\boldsymbol{\varepsilon}$ as the corresponding noise.

To show the asympsotics of $\widetilde{\boldsymbol{\theta}}_0\epsilon$, we first analyze the convergence of $\|\widehat{\boldsymbol{\theta}}_0(\lambda)-\boldsymbol{\theta}_0\|^2$ and $\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2$, and then use the closed-form solution when $n_2=\infty$ to connect $\widehat{\boldsymbol{\theta}}_0(\lambda)$ to $\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda)$.

For $\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2$, it can be decomposed as

$$\begin{aligned}\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2 &= \boldsymbol{y}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{y}\\ &= \boldsymbol{\theta}_0^\top\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}_0+\boldsymbol{\varepsilon}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{\varepsilon}\\ &\quad +2\boldsymbol{\theta}_0^\top\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{\varepsilon}.\end{aligned}$$

We look at the each term respectively. Notation-wise, for a symmetric matrix $A$, we define $A^{-2}=A^{-1}A^{-1}$ for simplicity.

Note that when taking $\Theta_n=\boldsymbol{\theta}_0\boldsymbol{\theta}_0^\top/d$ in Proposition 3, we have

$$\boldsymbol{\theta}_0^\top(\boldsymbol{X}^\top\boldsymbol{X}/n_1+\lambda\boldsymbol{I}_d)^{-1}\boldsymbol{\theta}_0\xrightarrow{a.s.}m_\gamma(-\lambda),$$

and taking the derivative w.r.t $\lambda$, it becomes

$$\boldsymbol{\theta}_0^\top(\boldsymbol{X}^\top\boldsymbol{X}/n_1+\lambda\boldsymbol{I}_d)^{-2}\boldsymbol{\theta}_0\xrightarrow{a.s.}m'_\gamma(-\lambda),$$

thus

$$\begin{aligned}&\boldsymbol{\theta}_0^\top\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}_0\\ =\;& r^2-2\lambda n_1\boldsymbol{\theta}_0^\top(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-1}\boldsymbol{\theta}_0+\lambda^2n_1^2\boldsymbol{\theta}_0^\top(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{\theta}_0\\ \xrightarrow{a.s.}\;& r^2\left[1-2\lambda m_\gamma(-\lambda)+\lambda^2m'_\gamma(-\lambda)\right].\end{aligned}\tag{C.1}$$

Furthermore, based on Proposition 4,

$$\begin{aligned}\boldsymbol{\varepsilon}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{\varepsilon}\;&\xrightarrow{a.s.}\;\sigma^2\left[\frac{1}{n_1}tr((\widehat{\Sigma}+\lambda\boldsymbol{I}_d)^{-1})-\frac{1}{n_1}\lambda tr((\widehat{\Sigma}+\lambda\boldsymbol{I}_d)^{-2})\right]\\ &\xrightarrow{a.s.}\;\sigma^2\left[\gamma m_\gamma(-\lambda)-\lambda\gamma m'_\gamma(-\lambda)\right],\end{aligned}\tag{C.2}$$

where $\widehat{\Sigma}:=\boldsymbol{X}^\top\boldsymbol{X}/n$.

For the cross term, following Proposition 4, we obtain

$$\begin{aligned}&\left[\boldsymbol{\theta}_0^\top\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{\varepsilon}\right]^2\\ \xrightarrow{a.s.}\;&\sigma^2tr\left[\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-2}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}_0\boldsymbol{\theta}_0^\top\right]\\ \xrightarrow{a.s.}\;&0.\end{aligned}\tag{C.3}$$

As a result, inserting (C.1), (C.2), and (C.3) into $\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2$, we obtain

$$\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2\xrightarrow{a.s.}r^2\left[1-2\lambda m_\gamma(-\lambda)+\lambda^2m'_\gamma(-\lambda)\right]+\sigma^2\gamma\left[m_\gamma(-\lambda)-\lambda m'_\gamma(-\lambda)\right].$$

For $\|\widehat{\boldsymbol{\theta}}_0(\lambda)-\boldsymbol{\theta}_0\|^2$, we have

$$\|\widehat{\boldsymbol{\theta}}_0(\lambda)-\boldsymbol{\theta}_0\|^2=\|\widehat{\boldsymbol{\theta}}_0(\lambda)\|^2+\|\boldsymbol{\theta}_0\|^2-2\widehat{\boldsymbol{\theta}}_0(\lambda)^\top\boldsymbol{\theta}_0,$$

where following a similar step as (C.3), we have

$$\left[\boldsymbol{\varepsilon}\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-1}\boldsymbol{\theta}_0\right]^2\xrightarrow{a.s.}0,\tag{C.4}$$

and

$$\begin{aligned}\widehat{\boldsymbol{\theta}}_0(\lambda)^\top\boldsymbol{\theta}_0\;&=\;\boldsymbol{\theta}_0\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-1}\boldsymbol{\theta}_0+\boldsymbol{\varepsilon}\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-1}\boldsymbol{\theta}_0\\ &\xrightarrow{a.s.}\;r^2-\lambda n_1\boldsymbol{\theta}_0(\boldsymbol{X}^\top\boldsymbol{X}+\lambda n_1\boldsymbol{I}_d)^{-1}\boldsymbol{\theta}_0\end{aligned}$$

$$\xrightarrow{a.s.} \quad r^2 - \lambda r^2 m_\gamma(-\lambda). \tag{C.5}$$

Consequently, inserting (C.4) and (C.5) into $\|\widehat{\boldsymbol{\theta}}_0(\lambda) - \boldsymbol{\theta}_0\|^2$, we obtain

$$\|\widehat{\boldsymbol{\theta}}_0(\lambda) - \boldsymbol{\theta}_0\|^2 \xrightarrow{a.s.} r^2\lambda^2 m'_\gamma(-\lambda) + \sigma^2\gamma[m_\gamma(-\lambda) - \lambda m'_\gamma(-\lambda)].$$

For adversarial training, from Javanmard et al. (2020); Xing et al. (2021b) we know that the minimizer of $R_\epsilon(\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}}_0(\lambda))$ is

$$\widetilde{\boldsymbol{\theta}}_\epsilon(\lambda) = (\boldsymbol{I}_d + \alpha \boldsymbol{I}_d)^{-1}\Sigma\widehat{\boldsymbol{\theta}}_0(\lambda),$$

where $c_0 = \sqrt{2/\pi}$ and $\alpha$ satisfies

$$\alpha\left(1 + \epsilon c_0 \frac{\|\widetilde{\boldsymbol{\theta}}\|}{\sqrt{\|\widetilde{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}_0\|^2 + \sigma^2}}\right) = \left(\epsilon c_0 \frac{\sqrt{\|\widetilde{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}_0\|^2 + \sigma^2}}{\|\widetilde{\boldsymbol{\theta}}\|} + \epsilon^2\right).$$

When $\boldsymbol{I}_d = \boldsymbol{I}_d$, the above is reduced to

$$\alpha + \epsilon c_0 \frac{\alpha\|\widehat{\boldsymbol{\theta}}_0\|}{\sqrt{\|\widehat{\boldsymbol{\theta}}_0\|^2\alpha^2 + \sigma^2(1+\alpha)^2}} = \epsilon c_0 \frac{\sqrt{\|\widehat{\boldsymbol{\theta}}_0\|^2\alpha^2 + \sigma^2(1+\alpha)^2}}{\|\widehat{\boldsymbol{\theta}}_0\|} + \epsilon^2.$$

□

# D Two-Layer Neural Networks

To show the convergence of the two-layer neural network, we first define the two models:

**NTK Model** Under **C1** and assume $\|\boldsymbol{w}_h - \boldsymbol{w}_h(0)\|_2 = O(1/\sqrt{H})$ for all $h = 1, \ldots, H$, we use the following formula to make prediction for each sample:

$$\begin{aligned} f^{NTK}_{\boldsymbol{W}}(\boldsymbol{x}) &= \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\left(\phi(\boldsymbol{w}_h(0)^\top\boldsymbol{x}) + (\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top\boldsymbol{x}\phi'(\boldsymbol{w}_h(0)^\top\boldsymbol{x})\right) && \text{(D.1)}\\ &= \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top\boldsymbol{x}\phi'(\boldsymbol{w}_h(0)^\top\boldsymbol{x}) && \text{(D.2)}\\ &:= \boldsymbol{V}(\boldsymbol{x})^\top(\boldsymbol{W} - \boldsymbol{W}(0)). \end{aligned}$$

We eliminate the term $\phi(\boldsymbol{w}_h(0)^\top\boldsymbol{x})$ from (D.1) to (D.2) because of the symmetric initialization. Denote $\boldsymbol{W}^{NTK}(t)$ as the training trajectory of NTK.

For the FGM of NTK, we have

$$\begin{aligned} \frac{\partial f^{NTK}_i(\boldsymbol{W})}{\partial \boldsymbol{x}_i} &= \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)[(\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top\boldsymbol{x}_i]\phi''(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0))\\ &\quad + \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h - \boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0)) \end{aligned}$$

which inspires use to define the following model:

**NTK Model with "Linear Attack" (lNTK)** We still use the same idea in (D.1) to make prediction for clean data. However, we make two changes compared to the NTK in terms of the attack and the formula for adversarial loss.

First, during the training, when calculating the attack, we use an attack parallel to

$$\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)).$$

Second, for the adversarial loss, denote $\widetilde{\boldsymbol{x}}$ as the data under attack. When taking the attack into the loss function, instead of inserting $\widetilde{\boldsymbol{x}}$ everywhere, we use the following perturbed prediction:

$$\begin{aligned} & f^{lNTK}_{\boldsymbol{W},adv}(\boldsymbol{x}) \\ = \;& \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\left(\phi(\boldsymbol{w}_h(0)^\top\boldsymbol{x})+(\boldsymbol{w}_h-\boldsymbol{w}_h(0))^\top\widetilde{\boldsymbol{x}}\phi'(\boldsymbol{w}_h(0)^\top\boldsymbol{x})\right) \\ = \;& \boldsymbol{V}(\boldsymbol{x})^\top(\boldsymbol{W}-\boldsymbol{W}(0)) \\ & +\epsilon\mathrm{sgn}(f^{NTK}_{\boldsymbol{W}}(\boldsymbol{x})-y)\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\left(\phi(\boldsymbol{w}_h(0)^\top\boldsymbol{x})+(\boldsymbol{w}_h-\boldsymbol{w}_h(0))\phi'(\boldsymbol{w}_h(0)^\top\boldsymbol{x})\right)\right\|_2. \end{aligned}$$

Denote $\boldsymbol{W}^{lNTK}(t)$ as the corresponding training trajectory.

**Overall Logic of the Proof for Theorem 1** We start from the simplest case, i.e., the convergence of lNTK. Under **C1**, lNTK can be reduced to a linear model, and the adversarial training converges to the same place as the adversarial training in linear regression. We will also show that, although NTK and lNTK uses different attacks, there is only a small difference in the final attacked samples, and the training trajectory are closed with each other. For the NN model, it is closed to NTK as well.

### D.1 lNTK Convergence

The following result illustrate the convergence of lNTK.

**Lemma 1.** *With probability tending to 1 over the randomness of the training data, initialization of the neural network, and the randomness of the testing data, using the following training configuration:*

- *gradient descent method,*
- *learning rate $\eta = \Theta(1)$ for a small enough constant so that $\eta \le L$ with $L$ as the largest Lipschitz constant of the gradient throughout training,*
- *training for $\Theta(\log n_1)$ iterations,*

*the solution of lNTK is close to the adversarially robust model with $\mathcal{L}_2$ regularization with multiplier $(\lambda + b_1^2 d/n_1)/b_0^2$.*

*Proof of Lemma 1.* To prove Lemma 1, our main target is to figure out the optimal solution of $\boldsymbol{W}$. Since one can check that the lNTK adversarial loss is convex w.r.t. $\boldsymbol{W}$, it will converge to the optimal solution easily when taking a small enough learning rate $\eta$ such that $\eta \le 1/L$ where $L$ is the Lipschitz constant of the lNTK loss w.r.t. $\boldsymbol{W}$.

Denote $\widetilde{\boldsymbol{X}}^{lNTK}(t)$ as the attack of the $n_1$ samples under the lNTK model at $t$th iteration. When no confusion arrises, we use $\widetilde{\boldsymbol{X}}$ for simplicity. Further denote $\boldsymbol{V}_{adv} \in \mathbb{R}^{Hd}$ as $\frac{1}{\sqrt{H}}\mathrm{vec}(a_h\widetilde{\boldsymbol{x}}\phi'(\boldsymbol{x}^\top\boldsymbol{w}_h(0)))$ and $\boldsymbol{U} \in \mathbb{R}^{d\times(Hd)}$ so that

$$\boldsymbol{U}(\boldsymbol{x})(\boldsymbol{W}(t)-\boldsymbol{W}(0)) = \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0))$$

which is the attack direction given a sample $\boldsymbol{x}$.

Denoting $\widetilde{\boldsymbol{X}}$ as the corresponding attack for the $n_1$ data, the best solution satisfies

$$\boldsymbol{W}^{lNTK}-\boldsymbol{W}(0) = \boldsymbol{V}_{adv}(\boldsymbol{X})^\top(\boldsymbol{V}_{adv}(\boldsymbol{X})\boldsymbol{V}_{adv}(\boldsymbol{X})^\top+\lambda n\boldsymbol{I}_{n_1})^{-1}\boldsymbol{y},$$

and

$$\boldsymbol{V}(\boldsymbol{x})(\boldsymbol{W}^{lNTK}-\boldsymbol{W}(0)) \quad = \quad \boldsymbol{V}(\boldsymbol{x})\boldsymbol{V}_{adv}(\boldsymbol{X})^\top(\boldsymbol{V}_{adv}(\boldsymbol{X})\boldsymbol{V}_{adv}(\boldsymbol{X})^\top+\lambda n\boldsymbol{I}_{n_1})^{-1}\boldsymbol{y}.$$

Based on Lemma 3, we have over the randomness of the neural network initialization,

$$\boldsymbol{V}_{adv}(\boldsymbol{x}_i)\boldsymbol{V}_{adv}(\boldsymbol{x}_j)=\widetilde{\boldsymbol{x}}_i^\top\widetilde{\boldsymbol{x}}_j c_\phi\left(\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}},\frac{\|\boldsymbol{x}_j\|}{\sqrt{d}},\frac{\boldsymbol{x}_i^\top\boldsymbol{x}_j}{d}\right)+O_p(1/\sqrt{H}).$$

When $i=j$,

$$\begin{aligned}&\boldsymbol{V}_{adv}(\boldsymbol{x}_i)\boldsymbol{V}_{adv}(\boldsymbol{x}_i)\\ =\;&\widetilde{\boldsymbol{x}}_i^\top\widetilde{\boldsymbol{x}}_i c_\phi\left(\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}},\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}},\frac{\boldsymbol{x}_i^\top\boldsymbol{x}_i}{d}\right)+O_p(1/\sqrt{H})\\ =\;&\widetilde{\boldsymbol{x}}_i^\top\widetilde{\boldsymbol{x}}_i\left(c_\phi\left(1,1,1\right)+c_\phi\left(\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}},\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}},\frac{\boldsymbol{x}_i^\top\boldsymbol{x}_i}{d}\right)-c_\phi\left(1,1,1\right)\right)+O_p(1/\sqrt{H}).\end{aligned}$$

When $i\neq j$,

$$\begin{aligned}&\boldsymbol{V}_{adv}(\boldsymbol{x}_i)\boldsymbol{V}_{adv}(\boldsymbol{x}_j)\\ =\;&\widetilde{\boldsymbol{x}}_i^\top\widetilde{\boldsymbol{x}}_i\left(c_\phi\left(1,1,0\right)+c_\phi\left(\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}},\frac{\|\boldsymbol{x}_j\|}{\sqrt{d}},\frac{\boldsymbol{x}_i^\top\boldsymbol{x}_j}{d}\right)-c_\phi\left(1,1,0\right)\right)+O_p(1/\sqrt{H}).\end{aligned}$$

As result, one can see that the eigenvalues of

$$\boldsymbol{V}_{adv}(\boldsymbol{X})\boldsymbol{V}_{adv}(\boldsymbol{X})^\top-\left(c_\phi(1,1,0)\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+(c_\phi(1,1,1)-c_\phi(1,1,0))\mathrm{diag}(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top)\right)$$

are all in $o_p(d)$ over the randomness of $\boldsymbol{X}$. In addition, the eigenvalues of

$$\begin{aligned}&c_\phi(1,1,0)\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+(c_\phi(1,1,1)-c_\phi(1,1,0))\mathrm{diag}(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top)\\ =\;&b_0^2\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2\mathrm{diag}(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top)+\lambda n\boldsymbol{I}_d\end{aligned}$$

are in $O_p(d)$.

Consequently, for a new data point $\boldsymbol{x}$ which satisfies that $\boldsymbol{x}_i^\top\boldsymbol{x}=o(d)$ for all $i=1,\ldots,n$, the prediction is

$$\begin{aligned}\boldsymbol{V}(\boldsymbol{x})(\boldsymbol{W}^{lNTK}-\boldsymbol{W}(0)) \quad &= \quad \boldsymbol{V}(\boldsymbol{x})\boldsymbol{V}_{adv}(\boldsymbol{X})^\top(\boldsymbol{V}_{adv}(\boldsymbol{X})\boldsymbol{V}_{adv}(\boldsymbol{X})^\top+\lambda n\boldsymbol{I}_{n_1})^{-1}\boldsymbol{y}\\ &= \quad \boldsymbol{x}^\top\widetilde{\boldsymbol{X}}^\top(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2d/b_0^2\boldsymbol{I}_{n_1}+\lambda n/b_0^2\boldsymbol{I}_{n_1})^{-1}\boldsymbol{y}+o,\end{aligned}$$

i.e., with probability tending to 1 over the randomness of the new sample $\boldsymbol{x}$, the prediction performance is similar to the adversarially robust ridge regression model when a penalty term $(b_1^2d/n+\lambda)/b_0^2$.

While the formula is similar to a linear predictor, we also need to verify that the attack gives by this linear predictor is the same as the one in lNTK. When taking the optimal solution of $\boldsymbol{W}^{lNTK}$, one can also verify that

$$\begin{aligned}&\boldsymbol{U}(\boldsymbol{x}_i)(\boldsymbol{W}^{lNTK}-\boldsymbol{W}(0))\\ =\;&\widetilde{\boldsymbol{X}}^\top(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2d/b_0^2\boldsymbol{I}_{n_1}+\lambda n/b_0^2\boldsymbol{I}_{n_1}))^{-1}\boldsymbol{y}\\ &+\frac{b_1^2}{b_0^2}\left[\boldsymbol{0}|\ldots|\widetilde{\boldsymbol{x}}_i|\ldots|\boldsymbol{0}\right](\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2d/b_0^2\boldsymbol{I}_{n_1}+\lambda n/b_0^2\boldsymbol{I}_{n_1}))^{-1}\boldsymbol{y}+o.\end{aligned}$$

One can see that with probability tending to 1,

$$\left\|\widetilde{\boldsymbol{X}}^\top(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2d/b_0^2\boldsymbol{I}_{n_1}+\lambda n/b_0^2\boldsymbol{I}_{n_1}))^{-1}\boldsymbol{y}\right\|^2=\Theta(1),$$

and

$$\left\|\left[\boldsymbol{0}|\ldots|\widetilde{\boldsymbol{x}}_i|\ldots|\boldsymbol{0}\right](\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2d/b_0^2\boldsymbol{I}_{n_1}+\lambda n/b_0^2\boldsymbol{I}_{n_1}))^{-1}\boldsymbol{y}\right\|^2=\Theta(1/d).$$

Consequently,

$$\boldsymbol{U}(\boldsymbol{x}_i)(\boldsymbol{W}^{lNTK}-\boldsymbol{W}(0))=\widetilde{\boldsymbol{X}}^\top(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top+b_1^2d/b_0^2\boldsymbol{I}_{n_1}+\lambda n/b_0^2\boldsymbol{I}_{n_1}))^{-1}\boldsymbol{y}+o.$$

From the above, we show that the optimal solution of lNTK gives similar predictions to adversarially robust linear regression with ridge penalty. To further justify that this solution can be achieved using gradient descent, we have

- the lNTK problem is strongly convex: the square loss term is convex and the regularization is strongly convex;
- denoting $L$ as the Lipschitz constant of the gradient throughout training, from the decomposition of the $\widetilde{\boldsymbol{y}}(t)$ in Proposition 5 below, one can also see that the the eigenvalues of $\boldsymbol{M}(t)$ are all positive and less than 1, and thus $L=O_p(1)$ throughout the training.

Finally, for the tail probability considered in the $o_p$, $O_p$ notations, the tail probability are all in $o(1/n_1)$. Following standard arguments in gradient descent, we take a learning rate which is less than $1/L$, and run the gradient descent for much more than $O(\log n_1)$ iterations, then the difference between the trained model and the optimal solution is in $o_p(1)$ with the tail probability $(\log n_1)/n_1\to 0$. □

After obtaining the optimal solution and algorithmic convergence of lNTK in Lemma 1, we look at the updating formula. Based on the updating criteria of lNTK, the updating gradient satisfies

$$\begin{aligned}\boldsymbol{w}_h(t)-\boldsymbol{w}_h(t-1) &= \frac{2\eta a_h}{n\sqrt{H}}\widetilde{\boldsymbol{X}}(t-1)\mathrm{diag}\left(\phi'(\boldsymbol{X}^\top\boldsymbol{w}_h(0))\right)(\boldsymbol{y}-\widetilde{\boldsymbol{y}}(t-1))\\ &\quad -2\eta\lambda(\boldsymbol{w}_h(t-1)-\boldsymbol{w}_h(0)),\end{aligned}$$

where the operator "diag" receives a vector and outputs a diagonal matrix and the diagonal elements are from the input.

Using the above, we can obtain the following results:

**Proposition 5.** *The change in the adversarial fitted value of the training data satisfies*

$$\begin{aligned}&\widetilde{\boldsymbol{y}}(t)-\boldsymbol{y}\\ =\ &\underbrace{\left((1-2\lambda\eta)\boldsymbol{I}_d-\frac{2\eta}{Hn}\sum_{h=1}^H a_h^2 diag\left(\phi'(\boldsymbol{X}\boldsymbol{w}_h(0))\right)\widetilde{\boldsymbol{X}}(t-1)^\top\widetilde{\boldsymbol{X}}(t-1)diag\left(\phi'(\boldsymbol{X}\boldsymbol{w}_h(0))\right)\right)}_{:=\boldsymbol{M}(t-1)}\times(\widetilde{\boldsymbol{y}}(t-1)-\boldsymbol{y})\\ &-2\lambda\eta\boldsymbol{y},\end{aligned}$$

*thus*

$$\widetilde{\boldsymbol{y}}(t)-\boldsymbol{y}=-\left(\prod_{\tau=1}^t\boldsymbol{M}(\tau-1)\right)\boldsymbol{y}-(2\lambda\eta)\left[\sum_{\tau=1}^t\prod_{\kappa=\tau-1}^{t-1}\boldsymbol{M}(\kappa)\right]\boldsymbol{y}.$$

*For $\partial f_i^{lNTK}(t)/\partial\boldsymbol{x}_i$, it satisfies that*

$$\begin{aligned}\frac{\partial f_i^{lNTK}(t)}{\partial\boldsymbol{x}_i} &= \frac{1}{\sqrt{H}}\sum_{h=1}^H a_h(\boldsymbol{w}_h-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0))\\ &= \frac{2\eta}{n}\sum_\tau\widetilde{\boldsymbol{X}}(\tau-1)diag\left(\frac{1}{H}\sum_{h=1}^H a_h^2\phi'(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0))\phi'(\boldsymbol{X}^\top\boldsymbol{w}_h(0))\right)(\boldsymbol{y}-\widetilde{\boldsymbol{y}}(\tau-1))\\ &\quad -2\eta\lambda\sum_\tau\frac{1}{\sqrt{H}}\sum_{h=1}^H a_h(\boldsymbol{w}_h(\tau-1)-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0)),\end{aligned}$$

*and over the randomness of the training data and the neural network initialization, for all pairs of* $(i, j)$*,*

$$\left\| \frac{\partial f_i^{lNTK}(t)}{\partial \boldsymbol{x}_i} - \frac{\partial f_j^{lNTK}(t)}{\partial \boldsymbol{x}_j} \right\| / \left\| \frac{\partial f_i^{lNTK}(t)}{\partial \boldsymbol{x}_i} \right\| = O_p(1/\sqrt{d}),$$

*i.e., the attack for all samples are almost parallel with each other, which further indicates that the eigenvalues of* $\boldsymbol{M}(t)$ *are all* $1 - \Theta(\eta)$*.*

### D.2 NTK Convergence

There are two differences comparing NTK and lNTK:

- Attack: for NTK, due to symmetric initialization, we have

$$\begin{aligned} & \frac{\partial f_i^{NTK}(\boldsymbol{W})}{\partial \boldsymbol{x}_i} \\ = \; & \frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h \boldsymbol{w}_h(0) \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) + a_h \boldsymbol{w}_h(0) [(\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top \boldsymbol{x}_i] \phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \\ & + \frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h (\boldsymbol{w}_h - \boldsymbol{w}_h(0)) \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \\ = \; & \frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h \boldsymbol{w}_h(0) [(\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top \boldsymbol{x}_i] \phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \\ & + \frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h (\boldsymbol{w}_h - \boldsymbol{w}_h(0)) \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)), \end{aligned} \tag{D.3}$$

  i.e., compared to lNTK, there is an extra term of

$$\frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h \boldsymbol{w}_h(0) [(\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top \boldsymbol{x}_i] \phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)).$$

- Adversarial loss: after obtaining $\widetilde{\boldsymbol{x}}$, lNTK only replaces one term as

$$f_{\boldsymbol{W},adv}^{lNTK}(\boldsymbol{x}) = \frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h \left( \phi(\boldsymbol{w}_h(0)^\top \boldsymbol{x}) + (\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top \widetilde{\boldsymbol{x}} \phi'(\boldsymbol{w}_h(0)^\top \boldsymbol{x}) \right),$$

  while NTK replaces all $\boldsymbol{x}$ with $\widetilde{\boldsymbol{x}}$ in the adversarial prediction

$$\frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h (\boldsymbol{w}_h - \boldsymbol{w}_h(0))^\top \widetilde{\boldsymbol{x}} [\phi'(\boldsymbol{w}_h(0)^\top \widetilde{\boldsymbol{x}}) - \phi'(\boldsymbol{w}_h(0)^\top \widetilde{\boldsymbol{x}})].$$

The following lemma shows that the difference in lNTK and NTK is negligible.

**Lemma 2.** *When the regularization* $\lambda$ *is a positive constant, the gradient* $\frac{\partial f_i^{NTK}(\boldsymbol{W})}{\partial \boldsymbol{x}_i}$ *is dominated by* $\frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h (\boldsymbol{w}_h(t) - \boldsymbol{w}_h(0)) \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))$ *throughout the training, and given the same attack, the difference between the adversarial prediction from lNTK and NTK is*

$$\frac{1}{\sqrt{H}} \sum_{h=1}^{H} a_h (\boldsymbol{w}_h(t) - \boldsymbol{w}_h(0))^\top \widetilde{\boldsymbol{x}} [\phi'(\boldsymbol{w}_h(0)^\top \boldsymbol{x}) - \phi'(\boldsymbol{w}_h(0)^\top \widetilde{\boldsymbol{x}})] = O_p(f_{\boldsymbol{W}}^{NTK}(\boldsymbol{x})/\sqrt{d}). \tag{D.4}$$

*Proof of Lemma 2.* When $\lambda > 0$, for the two terms in $\frac{\partial f_i^{NTK}(\boldsymbol{W})}{\partial \boldsymbol{x}_i}$ from (D.3), expanding $\boldsymbol{w}_h(t) - \boldsymbol{w}_h(0)$ as

$$\boldsymbol{w}_h(t) - \boldsymbol{w}_h(0) = \sum_{\tau} \boldsymbol{w}_h(\tau) - \boldsymbol{w}_h(\tau - 1),$$

and denoting $\widetilde{\boldsymbol{X}}(t)$ as the attacked samples, we have

$$\begin{aligned}
&\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \\
= \;& \frac{2\eta}{n}\sum_{\tau}\widetilde{\boldsymbol{X}}(\tau-1)\mathrm{diag}\left(\frac{1}{H}\sum_{h=1}^{H} a_h^2\phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\phi'(\widetilde{\boldsymbol{X}}(\tau-1)^\top \boldsymbol{w}_h(0))\right)(\boldsymbol{y}-\widetilde{\boldsymbol{y}}(\tau-1)) \\
& -2\eta\lambda\sum_{\tau}\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h(\tau-1)-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)),
\end{aligned}$$

and

$$\begin{aligned}
&\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)[(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))^\top \boldsymbol{x}_i]\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \\
= \;& \frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)\left[\frac{2\eta a_h}{n\sqrt{H}}\sum_{\tau}(\boldsymbol{y}-\widetilde{\boldsymbol{y}}(\tau-1))^\top \mathrm{diag}\left(\phi'(\widetilde{\boldsymbol{X}}(\tau-1)^\top \boldsymbol{w}_h(0))\right)\widetilde{\boldsymbol{X}}(\tau-1)^\top \boldsymbol{x}_i\right]\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \\
& -2\lambda\eta\sum_{\tau}\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)(\boldsymbol{w}_h(\tau-1)-\boldsymbol{w}_h(0))^\top \boldsymbol{x}_i\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)).
\end{aligned}$$

One can use induction to prove Lemma 2.

When $t=0$, since the predicted value is zero for all $\boldsymbol{x}$ due to symmetric initialization, there is no attack in the system, and one can check that for $t=1$,

$$\begin{aligned}
&\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\| \\
= \;& \left\|\frac{2\eta}{n}\boldsymbol{X}\mathrm{diag}\left(\frac{1}{H}\sum_{h=1}^{H} a_h^2\phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\phi'(\boldsymbol{X}^\top \boldsymbol{w}_h(0))\right)(\boldsymbol{y}-\boldsymbol{0})\right\| \\
= \;& O_p(\eta),
\end{aligned}$$

and

$$\begin{aligned}
&\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)[(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))^\top \boldsymbol{x}_i]\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\| \\
= \;& \left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)\left[\frac{2\eta a_h}{n\sqrt{H}}(\boldsymbol{y}-\boldsymbol{0})^\top \mathrm{diag}\left(\phi'(\boldsymbol{X}^\top \boldsymbol{w}_h(0))\right)\boldsymbol{X}^\top \boldsymbol{x}_i\right]\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\|.
\end{aligned}$$

One can see that $(\boldsymbol{y}-\boldsymbol{0})^\top \mathrm{diag}\left(\phi'(\boldsymbol{X}^\top \boldsymbol{w}_h(0))\right)\boldsymbol{X}^\top \boldsymbol{x}_i$ is dominated by some function $v_1$ of $\boldsymbol{x}_i$ and $\|\boldsymbol{w}_h(0)\|$, i.e., the effect from the direction of $\boldsymbol{w}_h(0)$ is negligible, and

$$\begin{aligned}
&\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)\left[\frac{2\eta a_h}{n\sqrt{H}}(\boldsymbol{y}-\boldsymbol{0})^\top \mathrm{diag}\left(\phi'(\boldsymbol{X}^\top \boldsymbol{w}_h(0))\right)\boldsymbol{X}^\top \boldsymbol{x}_i\right]\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\| \\
:= \;& \frac{\eta}{\sqrt{H}}\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\boldsymbol{w}_h(0)v_1(\boldsymbol{x}_i,\|\boldsymbol{w}_h(0)\|)\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\| + O_p(\eta/\sqrt{d}) \\
= \;& \frac{\eta}{\sqrt{H}}\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h\left[\boldsymbol{x}_i\frac{\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)}{\|\boldsymbol{x}_i\|^2}+\left(\boldsymbol{w}_h(0)-\boldsymbol{x}_i\frac{\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)}{\|\boldsymbol{x}_i\|^2}\right)\right]v_1(\boldsymbol{x}_i,\|\boldsymbol{w}_h(0)\|)\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\| + O_p(\eta/\sqrt{d}) \\
\leq \;& \frac{\eta}{\sqrt{H}}\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H} a_h \underbrace{\boldsymbol{x}_i\frac{\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)}{\|\boldsymbol{x}_i\|^2}}_{\text{parallel to } \boldsymbol{x}_i, O_p(1/\sqrt{d})} v_1(\boldsymbol{x}_i,\|\boldsymbol{w}_h(0)\|)\phi''(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0))\right\|
\end{aligned}$$

$$\frac{\eta}{\sqrt{H}}\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h\underbrace{\left(\boldsymbol{w}_h(0)-\boldsymbol{x}_i\frac{\boldsymbol{x}_i^\top\boldsymbol{w}_h(0)}{\|\boldsymbol{x}_i\|^2}\right)}_{\text{perp to }\boldsymbol{x}_i,O_p(1)}v_1(\boldsymbol{x}_i,\|\boldsymbol{w}_h(0)\|)\phi''(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0))\right\|+O_p(\eta/\sqrt{d})$$
$$= O_p(\eta/\sqrt{d})+O_p(\eta/\sqrt{H})+O_p(\eta/\sqrt{d})$$
$$= O_p(\eta/\sqrt{d}).$$

As a result, $\frac{\partial f_i^{NTK}(\boldsymbol{W})}{\partial \boldsymbol{x}_i}$ is dominated by

$$\frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h(\boldsymbol{w}_h-\boldsymbol{w}_h(0))\phi'(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0)).$$

In terms of (D.4), we have

$$\frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))^\top\widetilde{\boldsymbol{x}}[\phi'(\boldsymbol{w}_h(0)^\top\boldsymbol{x})-\phi'(\boldsymbol{w}_h(0)^\top\widetilde{\boldsymbol{x}})]$$
$$= \frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))^\top\widetilde{\boldsymbol{x}}\underbrace{(\boldsymbol{x}-\widetilde{\boldsymbol{x}})^\top\boldsymbol{w}_h(0)}_{=O_p(\epsilon/\sqrt{d})}\phi''(\boldsymbol{w}_h(0)^\top\boldsymbol{x})+o$$
$$= O_p(f_{\boldsymbol{W}(t)}^{NTK}(\boldsymbol{x})/\sqrt{d}).$$

The above derivation implies that $\|\boldsymbol{W}^{NTK}(t)-\boldsymbol{W}^{lNTK}(t)\|_F=O_P(\|\boldsymbol{W}^{lNTK}(t)-\boldsymbol{W}(0)\|_F/\sqrt{d})$ for $t=1$, and NTK is closed to lNTK. Thus one can use the lNTK properties in Proposition 5 to show that

$$\left\|\frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h\boldsymbol{w}_h(0)[(\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0))^\top\boldsymbol{x}_i]\phi''(\boldsymbol{x}_i^\top\boldsymbol{w}_h(0))\right\|=O_p(\eta t^2/\sqrt{d})$$

where $t$ is at most in $O(\log n_1)$, and the NTK attack is dominated by lNTK attack.

□

### D.3 NN Convergence

For the two-layer neural network, we have

- Prediction:

  $$f_{\boldsymbol{W}}^{NN}(\boldsymbol{x})-f_{\boldsymbol{W}}^{NTK}(\boldsymbol{x})=\frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h\phi(\boldsymbol{x}^\top\boldsymbol{w}_h)-a_h\phi(\boldsymbol{x}^\top\boldsymbol{w}_h(0))-a_h(\boldsymbol{w}_h-\boldsymbol{w}_h(0))^\top\boldsymbol{x}\phi'(\boldsymbol{x}^\top\boldsymbol{w}_h(0)).$$

  When $\|\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0)\|=o(1/\sqrt{d})$ for all $h=1,\ldots,H$, with probability tending to 1, $\|\boldsymbol{x}\|^2=O(d)$, and we have

  $$f_{\boldsymbol{W}}^{NN}(\boldsymbol{x})-f_{\boldsymbol{W}}^{NTK}(\boldsymbol{x})=O(\|\boldsymbol{x}\|^2\|\boldsymbol{w}_h(t)-\boldsymbol{w}_h(0)\|^2),$$

  which indicates that the prediction made by NN is similar to the one by NTK.

- Attack: Similarly, for the adversarial attack, we also have

  $$\frac{\partial}{\partial\boldsymbol{x}}\left(f_{\boldsymbol{W}}^{NN}(\boldsymbol{x})-f_{\boldsymbol{W}}^{NTK}(\boldsymbol{x})\right)$$
  $$= \frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h\boldsymbol{w}_h\underbrace{[\phi'(\boldsymbol{x}^\top\boldsymbol{w}_h)-\phi'(\boldsymbol{x}^\top\boldsymbol{w}_h(0))]}_{=O_p(\sqrt{d}\|\boldsymbol{w}_h-\boldsymbol{w}_h(0)\|)}-\underbrace{\frac{1}{\sqrt{H}}\sum_{h=1}^{H}a_h\boldsymbol{w}_h(0)(\boldsymbol{w}_h-\boldsymbol{w}_h(0))^\top\boldsymbol{x}\phi''(\boldsymbol{x}^\top\boldsymbol{w}_h(0))}_{\text{NTK result shows this is negligible}},$$

  which indicates that the attack of NN and NTK are also similar.

- Updating gradient: Recall that given attacked sample $\widetilde{\boldsymbol{X}}$, the NTK updating rule satisfies

$$\begin{aligned}\boldsymbol{w}_h^{NTK}(t)-\boldsymbol{w}_h(t-1) &= \frac{2\eta a_h}{n\sqrt{H}}\widetilde{\boldsymbol{X}}(t-1)\mathrm{diag}\left(\phi'(\widetilde{\boldsymbol{X}}(t-1)^\top\boldsymbol{w}_h(0))\right)(\boldsymbol{y}-\widetilde{\boldsymbol{y}}(t-1))\\ &\quad -2\eta\lambda(\boldsymbol{w}_h(t-1)-\boldsymbol{w}_h(0)),\end{aligned}$$

  and the NN updating rule is

$$\begin{aligned}\boldsymbol{w}_h(t)-\boldsymbol{w}_h(t-1) &= \frac{2\eta a_h}{n\sqrt{H}}\widetilde{\boldsymbol{X}}(t-1)\mathrm{diag}\left(\phi'(\widetilde{\boldsymbol{X}}(t-1)^\top\boldsymbol{w}_h(t-1))\right)(\boldsymbol{y}-\widetilde{\boldsymbol{y}}(t-1))\\ &\quad -2\eta\lambda(\boldsymbol{w}_h(t-1)-\boldsymbol{w}_h(0)),\end{aligned}$$

  and the only difference is on $\phi'(\widetilde{\boldsymbol{X}}(t-1)^\top\boldsymbol{w}_h(t-1)))$ and $\phi'(\widetilde{\boldsymbol{X}}(t-1)^\top\boldsymbol{w}_h(0))$, whose difference is also a negligible term.

The above comparisons indicate that NN is closed to NTK when $\|\boldsymbol{w}_h-\boldsymbol{w}_h(0)\|$ is small enough. Meanwhile, from optimal solution of lNTK and the connection between lNTK and NTK, we know that $\|\boldsymbol{w}_h^{NTK}(t)-\boldsymbol{w}_h(0)\|=O(1/\sqrt{H})$, which is much smaller than $1/\sqrt{d}$. Therefore, one can use induction to show that NN also converges to the solution of lNTK, i.e., the ridge regression solution in linear model.

### D.4 Proof of Theorem 2 for Two-Stage Method

The key difference between Theorem 1 and Theorem 2 is the sample size ($n_1 \asymp d$ vs $n_2=\infty$). It is essential to validate the lNTK solution for the two-stage method. For the NTK and NN result, as long as we figure out lNTK solution, they still converges to this solution.

For simplicity of the analysis, we take $\|\boldsymbol{w}_h(0)\|_2=1$ for all $\boldsymbol{w}_h(0)$.

For lNTK, since $n_2=\infty$, we know that the optimal solution is in the form of

$$\boldsymbol{W}^{lNTK}-\boldsymbol{W}(0)=\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}.$$

We first assume that the attack for all samples are almost parallel with each other to figure out $\boldsymbol{W}^{lNTK}$, and finally show that the obtained $\boldsymbol{W}^{lNTK}$ gives the correct attack.

The matrix $\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top\right)$ can be viewed as a $d\times d$ matrix of block matrices of size $H\times H$. Denote $\boldsymbol{V}_{i,j}$ $(i,j=1,\ldots,d)$ as the index of the block matrices. Observe that

$$\boldsymbol{V}_{i,j}\xrightarrow{P}\frac{\mathbb{E}\widetilde{X}_i\widetilde{X}_j}{H}\left(b_0^2\mathbf{1}\mathbf{1}^\top+b_1^2\boldsymbol{I}_H\right),$$

and

$$\frac{a_h}{\sqrt{H}}\mathbb{E}\widetilde{X}_i\phi'(X^\top\boldsymbol{w}_h(0))\widehat{y}\xrightarrow{P}\frac{a_h}{\sqrt{H}}b_0\mathbb{E}\left(\widetilde{X}_i\widehat{y}\right).$$

As a result,

$$\begin{aligned}&\boldsymbol{V}(\boldsymbol{x})^\top\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}\\ \xrightarrow{P}\;&\boldsymbol{V}(\boldsymbol{x})^\top\left\{\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}-\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\right\}\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}\\ &+\boldsymbol{V}(\boldsymbol{x})^\top\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}\\ =\;&\boldsymbol{V}(\boldsymbol{x})^\top\left\{\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}-\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\right\}\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}\\ &+\boldsymbol{x}^\top\left[\mathbb{E}\left(\widetilde{X}\widetilde{X}^\top\right)+\lambda\boldsymbol{I}_{H\times d}\right]^{-1}\mathbb{E}\left(\widetilde{X}\widehat{y}\right)+o,\end{aligned}$$

where

$$\begin{aligned}
&\left|\boldsymbol{V}(\boldsymbol{x})^\top\left\{\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}-\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\right\}\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}\right| \\
\leq\ & \|\boldsymbol{V}(\boldsymbol{x})\|\,\|\mathbb{E}\boldsymbol{V}_{adv}(X)\widehat{y}\|\left\|\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}-\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\right\| \\
=\ & O\left(\sqrt{d}\left\|\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}-\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\right\|\right).
\end{aligned}$$

When $\lambda$ is a positive constant, we have

$$\begin{aligned}
&\left\|\left[\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top+\lambda\boldsymbol{I}_{H\times d}\right)\right]^{-1}-\left(\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)+\lambda\boldsymbol{I}_{H\times d}\right)^{-1}\right\| \\
=\ & O\left(\left\|\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top\right)-\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)\right\|\right),
\end{aligned}$$

where

$$\begin{aligned}
&\left\|\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top\right)-\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)\right\| \\
\leq\ & \left\|\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top\right)-\frac{1}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes\mathbb{E}(\phi'(X^\top\boldsymbol{W}(0))\phi'(X^\top\boldsymbol{W}(0))^\top)\right\| \\
&+\left\|\frac{1}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes\mathbb{E}(\phi'(X^\top\boldsymbol{W}(0))\phi'(X^\top\boldsymbol{W}(0))^\top)-\frac{b_0^2}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes(\mathbf{1}\mathbf{1}^\top)\right\|.
\end{aligned}\tag{D.5}$$

For the first term of (D.5), it can be decomposed as

$$\begin{aligned}
&\mathbb{E}\widetilde{X}(\widetilde{X}-X)^\top(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0)))-\mathbb{E}\widetilde{X}(\widetilde{X}-X)^\top\mathbb{E}(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0))) \\
&+\mathbb{E}(\widetilde{X}-X)\widetilde{X}^\top(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0)))-\mathbb{E}(\widetilde{X}-X)\widetilde{X}^\top\mathbb{E}(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0))) \\
&+\mathbb{E}XX^\top(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0)))-\mathbb{E}XX^\top\mathbb{E}(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0))).
\end{aligned}\tag{D.6}$$

We study the conditional distribution of $X$ given $X^\top\boldsymbol{w}_h(0)$, $X^\top\boldsymbol{w}_{h'}(0)$, $X^\top\boldsymbol{\theta}$ and $X^\top\widehat{\boldsymbol{\theta}}_0$, where $\boldsymbol{\theta}$ is the linear form we assumed as the final trained model. Since $X\sim N(\mathbf{0},\boldsymbol{I}_d)$, we have for any $h$ and $h'$, the non-zero eigenvalues of the matrix

$$\mathbb{E}XX^\top(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0)))-\mathbb{E}XX^\top\mathbb{E}(\phi'(X^\top\boldsymbol{w}_h(0))\phi'(X^\top\boldsymbol{w}_{h'}(0)))$$

are in $O(1)$ and are associated with the vectors $\boldsymbol{w}_h(0)$ and $\boldsymbol{w}_{h'}(0)$, i.e., the eigenvalues are different when changing $h$ and $h'$, thus with high probability,

$$\left\|\mathbb{E}\left(\boldsymbol{V}(X)\boldsymbol{V}(X)^\top\right)-\frac{1}{H}\mathbb{E}XX^\top\otimes\mathbb{E}(\phi'(X^\top\boldsymbol{W}(0))\phi'(X^\top\boldsymbol{W}(0))^\top)\right\|=O(1/d).$$

On the other hand, in terms of the attack term $(\widetilde{X}-X)X^\top$, we have $\mathbb{E}_{X|X^\top\boldsymbol{w}_h(0)}(\widetilde{X}-X)X^\top=O_p(1/\sqrt{d})$ and its expectation w.r.t. the randomness of $\boldsymbol{w}_h(0)$ is in $O(1/d)$, thus we get

$$\left\|\mathbb{E}\left(\boldsymbol{V}_{adv}(X)\boldsymbol{V}_{adv}(X)^\top\right)-\frac{1}{H}\mathbb{E}\widetilde{X}\widetilde{X}^\top\otimes\mathbb{E}(\phi'(X^\top\boldsymbol{W}(0))\phi'(X^\top\boldsymbol{W}(0))^\top)\right\|=O(1/d),$$

with high probability.

For the second term of (D.5), we have with high probability,

$$\left\|\mathbb{E}(\phi'(X^\top\boldsymbol{W}(0))\phi'(X^\top\boldsymbol{W}(0))^\top)-b_0^2(\mathbf{1}\mathbf{1}^\top)-b_1^2\boldsymbol{I}_H\right\|=O(1/d),\tag{D.7}$$

thus

$$\left\| \frac{1}{H} \mathbb{E} \widetilde{X} \widetilde{X}^\top \otimes \mathbb{E}(\phi'(X^\top \boldsymbol{W}(0)) \phi'(X^\top \boldsymbol{W}(0))^\top) - \frac{b_0^2}{H} \mathbb{E} \widetilde{X} \widetilde{X}^\top \otimes (\mathbf{1}\mathbf{1}^\top) \right\| = O(1/d)$$

as well.

Finally, we have with large probability,

$$\boldsymbol{V}(\boldsymbol{x})^\top \left[ \mathbb{E} \left( \boldsymbol{V}_{adv}(X) \boldsymbol{V}_{adv}(X)^\top + \lambda \boldsymbol{I}_{H \times d} \right) \right]^{-1} \mathbb{E} \boldsymbol{V}_{adv}(X) \widehat{y} = \boldsymbol{x}^\top \left[ \mathbb{E} \left( \widetilde{X} \widetilde{X}^\top \right) + \lambda \boldsymbol{I}_{H \times d} \right]^{-1} \mathbb{E} \left( \widetilde{X} \widehat{y} \right) + o.$$

### D.5 Additional Lemmas

**Lemma 3.** *Denote $g \sim N(0,1)$, and*

$$b_0 = \mathbb{E}\phi'(g), \quad b_1^2 = \mathbb{E}\phi'(g)^2 - b_0^2.$$

*There exists some positive function $c_\phi$ such that,*

$$\frac{1}{H} \sum_{h=1}^{H} a_h^2 \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \phi'(\boldsymbol{x}_j^\top \boldsymbol{w}_h(0)) = c_\phi \left( \frac{\|\boldsymbol{x}_i\|}{\sqrt{d}}, \frac{\|\boldsymbol{x}_j\|}{\sqrt{d}}, \frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{d} \right) + O_p(1/\sqrt{D}).$$

*The function $c_\phi$ satisfies that*

$$c_\phi(1,1,1) = b_0^2 + b_1^2, \qquad c_\phi(1,1,0) = b_0^2.$$

*Proof of Lemma 3.* To show Lemma 3, we have

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{w}_h(0)} a_h^2 \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \phi'(\boldsymbol{x}_j^\top \boldsymbol{w}_h(0)) \\ = \; & \mathbb{E}_g \left\{ \mathbb{E} \left[ a_h^2 \phi'(\|\boldsymbol{x}_i\| g) \phi'(\boldsymbol{x}_j^\top \boldsymbol{w}_h(0)) \mid \boldsymbol{x}_i^\top \boldsymbol{w}_h(0) = \|\boldsymbol{x}_i\| g \right] \right\} \\ = \; & \mathbb{E}_g \left\{ \mathbb{E} \left[ a_h^2 \phi'(\|\boldsymbol{x}_i\| g) \phi' \left( g \frac{\boldsymbol{x}_j^\top \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} + \boldsymbol{x}_j^\top \left( \boldsymbol{w}_h(0) - \frac{g \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} \right) \right) \mid \boldsymbol{x}_i^\top \boldsymbol{w}_h(0) = \|\boldsymbol{x}_i\| g \right] \right\}. \end{aligned}$$

One can see that, given $\|\boldsymbol{x}_i\|$ and $\boldsymbol{x}_i^\top \boldsymbol{x}_j$,

$$\mathbb{E} \left[ a_h^2 \phi'(\|\boldsymbol{x}_i\| g) \phi' \left( g \frac{\boldsymbol{x}_j^\top \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} + \boldsymbol{x}_j^\top \left( \boldsymbol{w}_h(0) - \frac{g \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} \right) \right) \mid \boldsymbol{x}_i^\top \boldsymbol{w}_h(0) = \|\boldsymbol{x}_i\| g \right]$$

is only a function of $\|\boldsymbol{x}_j\|$, but not depends on the particular direction of $\boldsymbol{x}_j$. As a result, $\mathbb{E}_{\boldsymbol{w}_h(0)} a_h^2 \phi'(\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)) \phi'(\boldsymbol{x}_j^\top \boldsymbol{w}_h(0))$ is a function of $\|\boldsymbol{x}_i\|$, $\|\boldsymbol{x}_j\|$, and $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ only.

Given $H$ hidden nodes, the empirical average converges to the above expectation.

When $\|\boldsymbol{x}_i\| = \|\boldsymbol{x}_j\| = \sqrt{d}$ and $\boldsymbol{x}_i^\top \boldsymbol{x}_j = 0$, we have $\boldsymbol{x}_i^\top \boldsymbol{w}_h(0)$ and $\boldsymbol{x}_j^\top \boldsymbol{w}_h(0)$ are two independent $N(0,1)$ variables considering the randomness of $\boldsymbol{w}_h(0)$, which indicates that

$$c_\phi(1,1,0) = b_0^2.$$

When $\|\boldsymbol{x}_i\| = \sqrt{d}$, it is easy to see that

$$c_\phi(1,1,1) = b_0^2 + b_1^2.$$

□

## E Discussions in ReLU Networks

Although Ba et al. (2020) provides justifications on ReLU network to connect it with linear neural networks, its mechanism differs from other non-linear networks. As in Equation (172) of Ba et al. (2020), they separate ReLU as a positive part and a negative part, and thus the sum of them is still a linear network. They also assume symmetric data to ensure that the positive and negative parts always behave symmetrically. In contrast, for other smooth non-linear neural networks, the proof leverages the neural tangent kernel and the fact that the trained parameters are close to their initialization. When taking the above to adversarial training, such a difference happens again.

We conducted some simulation trials to check how ReLU works. In Table 3, we check change in $\|\boldsymbol{W}(t)-\boldsymbol{W}(0)\|$ compared to $\|\boldsymbol{W}(0)\|$, and we also monitor the difference of the updating gradient of NN and lNTK $\|g_1(t)-g_2(t)\|$. For $\|\boldsymbol{W}(t)-\boldsymbol{W}(0)\|$, the ReLU version is still small. However, for $\|g_1(t)-g_2(t)\|$, ReLU gets a larger value than Sigmoid. We conjecture that ReLU may still converge to the linear regression solution, but the detailed proofs need to be changed accordingly.

| $d=20, n_1=1000$ | ReLU | | | Sigmoid | | |
|---|---|---|---|---|---|---|
| H | $\|\boldsymbol{W}(t)-\boldsymbol{W}(0)\|$ | $\|\boldsymbol{W}(0)\|$ | $\|g_1(t)-g_2(t)\|$ | $\|\boldsymbol{W}(t)-\boldsymbol{W}(0)\|$ | $\|\boldsymbol{W}(0)\|$ | $\|g_1(t)-g_2(t)\|$ |
| 10 | 2.7158 | 9.98 | 0.029903 | 2.4550 | 9.98 | 2.29E-03 |
| 20 | 2.8362 | 19.93 | 0.021782 | 2.5388 | 19.93 | 1.22E-03 |
| 100 | 2.8419 | 99.89 | 0.003365 | 2.5705 | 99.89 | 5.37E-05 |
| 200 | 2.9390 | 200.59 | 0.001232 | 2.5795 | 200.59 | 2.25E-07 |
| 1000 | 3.0019 | 1000.05 | 0.000129 | 2.5801 | 1000.05 | 2.48E-09 |
| 2000 | 3.0203 | 1997.07 | 0.000047 | 2.5805 | 1997.07 | 4.13E-10 |

Table 3: Comparison between ReLU and Sigmoid

## F Additional Experiments

We take different values of $H$ and $n_1$ and check the difference between the adversarial training loss of the two methods and the adversarial testing loss of the two methods. The results are summarized in Table 4. For the best training result, the Sigmoid network has a stronger expressive power than a linear function, so the smallest training loss is smaller than its linear counterpart. In addition, the smallest training loss is always achieved at the last epoch. In terms of the best testing result, it appears in some intermediate steps during the training, and the the performance of the Sigmoid network is similar to the linear counterpart (the mean difference is not significantly away from zero considering the standard error when $n_1 > 10$).

| $d=20$ | | | | | |
|---|---|---|---|---|---|
| $H$ | $n_1$ | diff(adv train loss) | std | diff(adv test loss) | std |
| 100 | 10 | -0.0462 | 0.1201 | -1.7120 | 0.8290 |
| 1000 | 10 | -0.0360 | 0.0851 | -1.7122 | 0.8289 |
| 10000 | 10 | -0.0347 | 0.0904 | -1.7121 | 0.8286 |
| 100 | 100 | -0.2451 | 0.0396 | -0.0790 | 0.0682 |
| 1000 | 100 | -0.2147 | 0.0286 | -0.0756 | 0.0694 |
| 10000 | 100 | -0.2084 | 0.0273 | -0.0750 | 0.0694 |
| 100 | 1000 | -0.0131 | 0.0037 | -0.0008 | 0.0029 |
| 1000 | 1000 | -0.0112 | 0.0028 | 0.0002 | 0.0025 |
| 10000 | 1000 | -0.0111 | 0.0030 | 0.0007 | 0.0023 |

Table 4: Chaning $H$ and $n_1$.