Phase Transition from Clean Training to Adversarial Training

Yue Xing
Department of Statistics
Purdue University
xing49@purdue.edu

Qifan Song
Department of Statistics
Purdue University
qfsong@purdue.edu

Guang Cheng
Department of Statistics
University of California, Los Angeles
guangcheng@ucla.edu

Abstract

Adversarial training is one important algorithm to achieve robust machine learning models. However, numerous empirical results show a great performance degradation from clean training to adversarial training (e.g., 90+% vs 67% testing accuracy on CIFAR-10 dataset), which does not match the theoretical guarantee delivered by the existing studies. Such a gap inspires us to explore the existence of an **(asymptotic)** phase transition phenomenon with respect to the attack strength: adversarial training is as well behaved as clean training in the small-attack regime, but there is a sharp transition from clean training to adversarial training in the large-attack regime. We validate this conjecture in linear regression models, and conduct comprehensive experiments in deep neural networks.

1 Introduction

Among various algorithms towards adversarially robust models, adversarial training is a popular and simple way to improve the adversarial robustness of the model. Many studies improve adversarial training from either theoretical or empirical aspects.

To justify the theoretical properties, an abundant literature studies the performance of adversarially robust models under common statistical models, e.g., linear regression, Gaussian mixture model and etc. in Javanmard et al. (2020); Mehrabi et al. (2021); Javanmard and Soltanolkotabi (2020); Dan et al. (2020); Taheri et al. (2020); Xing et al. (2021c,a). Besides, some other studies work on deep learning models. For example, Gao et al. (2019); Zhang et al. (2020b); Allen-Zhu and Li (2020) show the convergence of adversarial training in neural networks when the attack strength is sufficiently small. Other literature, e.g., Zhang et al. (2019); Wang et al. (2019b,a), propose improvements in the training loss of adversarial training.

However, there are some gaps between the promising theoretical results and the actual performance of adversarial training in real practice. First, the aforementioned theoretical works consider either simple models or sufficiently small attack strength. For simple models, adversarial training is well-behaved because the adversarial loss is still convex. For complicated models with small attacks, they conduct a similar analysis for adversarial training as for clean training, in the sense that adversarial training only introduces some additional diminishing error. Such settings are different from the setups used in real practice. Second, some other studies also reveal several potential concerns of adversarial training in deep learning, e.g., the smoothness problem Lee and Chandrakasan (2020); Xie et al. (2020); Xing et al. (2021a), overfitting issue Rice et al. (2020); Wu et al. (2020). Based on RobustBench (`https://robustbench.github.io/`), the best robust accuracy for CIFAR-10 using state-of-the-art methods is 66.6% Rebuffi et al. (2021), much worse than the clean accuracy of above 90% in clean training.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

As our attempt to fill these gaps, this work focuses on the role of adversarial attack strength. For instance, we are wondering whether taking 8/255 $\mathcal{L}_\infty$ attack is too strong for CIFAR-10 adversarial training using the current state-of-the-art neural networks architectures and optimizer configurations. Although clean training generalizes well in this setup and adversarial training can be viewed as an adaptation of clean training, they may act differently.

This paper investigates how adversarial training deviates from clean training as the attack strength grows. An illustration of this can be found in Figure 1. Under small attack strength ϵ, the trajectory of adversarial training is close to that of clean training so that they converge to the same flat minima. As ϵ increases, the adversarial training process deviates from the clean training dramatically. Our aim is to determine a critical attack level, denoted as ϵ^*, such that (i) when $\epsilon \ll \epsilon^*$, (S)GD trajectories of clean training and adversarial training are similar in some sense so that they can be handled similarly; but (ii) when $\epsilon \gg \epsilon^*$, the adversarial training will significantly deviate from clean training.

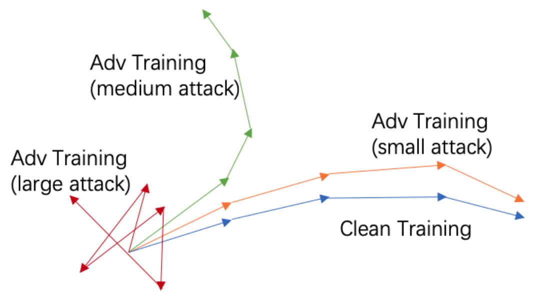

Figure 1: The trajectory of clean training, adversarial training with small/large attack.

Some other literature also supports our conjecture that adversarial training differs from clean training under large attack strength. Gong et al. (2017); Frederickson et al. (2018) show that, as the attack strength increases, the attacked samples are more detectable against clean samples. Such a difference in the training samples leads to the discrepancy between clean and adversarial training. In addition, the trained model from adversarial training is different from the clean model in terms of its Hessian Yao et al. (2018). Furthermore, the neural network capacity required by adversarial training is larger than clean training Xie and Yuille (2019); Bubeck and Sellke (2021).

Note that the "phase transition" phenomenon in this paper refers to an asymptotic behavior rather than a transition at some specific point. In thermodynamics, the phase transition means the state of a substance changes at an exact threshold temperature, e.g., the melting point. In our case, the threshold ϵ^* refers to an asymptotic order w.r.t. n and model dimension. The behavior of the training trajectory is different under $\epsilon \ll \epsilon^*$ and $\epsilon \gg \epsilon^*$ ($\epsilon/\epsilon^* \to 0$ and $\epsilon/\epsilon^* \to \infty$ in sample size and possibly in data dimension as well).

Contributions The main contributions of this paper can be summarized as follows:

- Directly defining the critical value ϵ^* requires comparing the whole optimization trajectories between clean training and adversarial training. The whole path comparison can be rather difficult. Thus instead, from theoretical intuitions obtained in simple models, we identify the critical ϵ^* via a surrogate measure that only relies on the training losses at trajectory destination. In particular, we propose to use clean testing loss in clean training (denote as $R^*(0)$) as a benchmark for adversarial training. If the attack strength ϵ is small enough so that the (proper) adversarial training loss is smaller than $R^*(0)$, then the adversarial training is similar to the clean training (and vice versa).
- To compute ϵ^*, a naive way is to repeatedly perform the adversarial training from scratch (i.e., initializing from a random start) for each possible ϵ until the corresponding adversarial training loss hits $R^*(0)$. This could be time-consuming. To speed up the computation, we also propose a method based on adversarial fine-tuning and extrapolation to approximate ϵ^*.

  Due to the page limit, we postpone many details of approximating ϵ^* to Appendix A.

Last but not least, we emphasize that ϵ^* severs as a diagnosis of the adversarial training, but not a rule of how to choose the attack level. That is, when $\epsilon \ll \epsilon^*$, one can safely assume adversarial training works as well as clean training; when $\epsilon \gg \epsilon^*$, it indicates that the attack strength is too large for the current data set using the current training configuration, and certain remedy is necessary such that the adversarial training performs as good as its clean counterpart. It is beyond our scope to provide a complete solution of how to adjust adversarial training, but some discussions are given in Section 7.

2 Other Related Works

Clean Training in Deep Learning Many existing literature studies deep neural networks in clean training. For the training performance, studies such as Du et al. (2019, 2018); Du and Lee (2018) work on the optimization convergence of the empirical training loss. In terms of the testing performance, there are many works showing the good performance of clean training in different aspects, e.g., double-descent phenomenon and benign over-fitting Belkin et al. (2018); Hastie et al. (2019); Ba et al. (2020); Bartlett et al. (2020), implicit regularization Neyshabur et al. (2017b); Neyshabur (2017); Baratin et al. (2020), generalization bounds using PAC-Bayes method or other methods Neyshabur et al. (2017a); Kohler and Langer (2019); Hu et al. (2020); Taheri et al. (2021).

Adversarial Training In the literature, there are several strands of theoretical studies related to adversarial training, e.g., the statistical properties or generalization performance of the global optimizer of adversarial training loss without any consideration of solving the optimization problem Javanmard et al. (2020); Yin et al. (2018); Raghunathan et al. (2019); Min et al. (2020); Zhai et al. (2019); Hendrycks et al. (2019); Chen et al. (2020a); Xing et al. (2021c); Dan et al. (2020), the (local) convergence of adversarial training algorithms under convexity assumptions Sinha et al. (2018); Wang et al. (2019a), the adversarial training loss in deep neural networks (Gao et al., 2019; Zhang et al., 2020b; Wu et al., 2020), and the generalization properties of deep neural networks, e.g. upper bound in Allen-Zhu and Li (2020) and lower bound in Bubeck and Sellke (2021).

3 Adversarial Training

To introduce adversarial training, let l denote the loss function and $f_\theta(x)$ be the model with parameter θ. The (population) adversarial loss is defined as $R(\theta,\epsilon) := \mathbb{E}\left[l\left(f_\theta[x + A_\epsilon(f_\theta, x, y)], y\right)\right]$, where A_ϵ is an attack of strength $\epsilon > 0$ and intends to deteriorate the loss in the following way

$$A_\epsilon(f_\theta, x, y) := \underset{z \in B_p(0,\epsilon)}{\operatorname{argmax}} \{l(f_\theta(x+z), y)\}, \tag{1}$$

where $B_p(x,r)$ is a $\mathcal{L}_p$ ball centering at x with radius r. Given n i.i.d. samples $S = \{(x_i, y_i)\}_{i=1}^n$, the adversarial training minimizes the sample version of $R(\theta,\epsilon)$ w.r.t. θ:

$$\widehat{R}(\theta,\epsilon) = \frac{1}{n}\sum_{i=1}^{n} l\left(f_\theta[x_i + A_\epsilon(f_\theta, x_i, y_i)], y_i\right), \tag{2}$$

and the estimator $\widehat{\theta}(\epsilon)$ aims to minimize $\widehat{R}(\theta,\epsilon)$, and the minimized adversarial training loss is $\widehat{R}_S(\epsilon)$. We rewrite $\widehat{R}(\theta,\epsilon)$ as $\widehat{R}(\theta)$ for simplicity when there is no confusion.

The minimization in (2) is often implemented through an iterative two-step (min-max) update. In the t-th iteration, we calculate the adversarial sample $\widetilde{x}_i^{(t)} = x_i + A_\epsilon(f_{\theta^{(t)}}, x_i, y_i)$ based on the current $\theta^{(t)}$, and then update $\theta^{(t+1)}$ based on the gradient of the adversarial training loss while fixing $\widetilde{x}_i^{(t)}$'s with learning rate η_t. The algorithm runs for T iterations. Note that for some loss function l or model f_θ (e.g. deep neural networks), there may not be an analytic form for A_ϵ, thus numerical methods, e.g. FGM and PGD, are utilized to approximate A_ϵ. Denote $\theta^{(t)}(\epsilon)$ as the adversarially trained model using attack strength ϵ.

4 Intuitions from Simple Models

To obtain insights into the critical strength ϵ^*, we consider the linear regression models. Briefly speaking, we justify that ϵ^* is in $\Theta(\sqrt{d/n})$ under $\mathcal{L}_2$ attack by comparing the training trajectories of clean and adversarial training, as well as establishing an explicit connection between the critical bound and the generalization error of clean training.

Simple Linear Regression For simple linear regression problem, to measure the difference between the trajectories of adversarial training and clean training, if $\epsilon \ll \sqrt{d/n}$[1], then asymptotically, there is

[1]With slightly modification of usual notations, we denote $\epsilon \ll \sqrt{d/n}$ as $\epsilon \log^k n/\sqrt{d/n} \to 0$ for any fixed $k > 0$ to accommodate with some tail probability bounds. The definition of $\epsilon \gg \sqrt{d/n}$ is modified similarly.

no difference between adversarial training and clean training during the training process. Throughout the training (with proper early stopping), we always have that the updating gradient of adversarial training is dominated by the one in clean training. To be specific, we have the following result:

Theorem 1. *Assume $Y = \theta_0^\top X + \varepsilon$ where $X \in \mathbb{R}^d$ follows multivariate normal distribution with zero mean and covariance I_d, and ε is a Gaussian noise with constant variance. The true coefficient θ_0 satisfies $\|\theta_0\| = \Theta(1)$ so that $Var(Y) = \Theta(1)$ as well. Consider $\mathcal{L}_2$-adversarial training.*

When $\epsilon \ll \sqrt{d/n}$, with zero initialization[2], proper learning rate η and number of steps T, the optimization converges to the global risk minimizer, i.e., $\theta^{(T)}(0) \to \widehat{\theta}(0)$ and $\theta^{(T)}(\epsilon) \to \widehat{\theta}(\epsilon)$. Besides, with probability tending to 1, for all $t \leq T$, the updates in clean and adversarial training have insignificant difference (small attack in Figure 1):

$$\|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\| / \|Std(\widehat{\theta}(0))\|_F \to 0.$$

When $\epsilon \gg \sqrt{d/n}$, with probability tending to 1,

- *If $\liminf d/n > 0$[3], both $\|\widehat{\theta}(\epsilon)\|$ and $\|\widehat{\theta}(\epsilon)\| / \|\widehat{\theta}(0)\|$ converge to 0 (implying a training trajectory wandering around 0, i.e., large attack in Figure 1). That is, the empirical adversarial risk minimizer is asymptotically zero, giving all training and testing adversarial predictions as zero.*
- *If $\lim d/n = 0$, $\|\widehat{\theta}(\epsilon) - \widehat{\theta}(0)\| / \|Std(\widehat{\theta}(0))\|_F \to \infty$, i.e., the adversarial training trajectory will be statistically different from the clean training process (medium attack in Figure 1).*
- *Such a difference between $\widehat{\theta}(\epsilon)$ and $\widehat{\theta}(0)$ implies the difference in the training trajectories.*

The proof (and details of η and T) of Theorem 1 is postponed to the appendix.

Theorem 1 implies that $\epsilon^* = \Theta(\sqrt{d/n})$, which gives an appropriate asymptotic order for the critical strength level under linear regression model. However, reproducing such an analysis to find ϵ^* for complex statistical models is more mathematically involved or even intractable. This motivates us to seek a simpler surrogate measure to identify ϵ^*.

We find that the difference between the clean training loss and clean testing loss can be a proper choice of such a surrogate measure, as illustrated in Theorem 2 below. Using the loss to do comparison helps avoid directly comparing model parameters (i.e., $\theta^{(t)}(\epsilon)$ vs $\theta^{(t)}(0)$), which eases the comparison for complex models such as neural networks in practice. Denote Δ as the difference between the clean training loss and clean testing loss of clean trained model, i.e., $\Delta = R(\widehat{\theta}(0), 0) - \widehat{R}(\widehat{\theta}(0), 0)$. In practice, one can use $R(\theta^{(T)}(0), 0) - \widehat{R}(\theta^{(T)}(0), 0)$ as an estimate of Δ.

Theorem 2. *Under the model setup as in Theorem 1, regardless of the growth of dimension d and sample size n, when $\epsilon \ll \sqrt{d/n}$, with high probability,*

$$(\widehat{R}(\widehat{\theta}(0), 0) - \widehat{R}(\widehat{\theta}(\epsilon), \epsilon)) / \Delta \to 0.$$

When $\epsilon \gg \sqrt{d/n}$, the above amount does not go to 0.

The proof of Theorem 2 is postponed to the appendix.

Both Theorems 1 and 2 describe a phase-transition under linear models with the same phase-transition boundary. This synchronicity justifies our idea of defining critical attack strength ϵ^* implicitly via the empirical loss and generalization gap Δ. Furthermore, since in real practice mostly we have $R(\widehat{\theta}(0), 0) = \widehat{R}(\widehat{\theta}(0), 0) + \Delta$, ϵ^* can be defined as

$$\epsilon^* := \sup\{\epsilon \mid \widehat{R}(\widehat{\theta}(\epsilon), \epsilon) \leq R(\widehat{\theta}(0), 0)\}.$$

It is worth mentioning that it is not coincident that both the gap of training losses and the difference of training trajectories occur at the same point. Under linear models, the loss function is always convex for both clean and adversarial training; hence the similarity of gradient descent trajectories highly depends on the similarity of the loss functions.

[2]Note that the adversarial loss is not differentiable at $\theta = \mathbf{0}$. But since ϵ is sufficiently small, it does not affect the convergence.

[3]We exclude the case that $d \in [n-1, n+1]$.

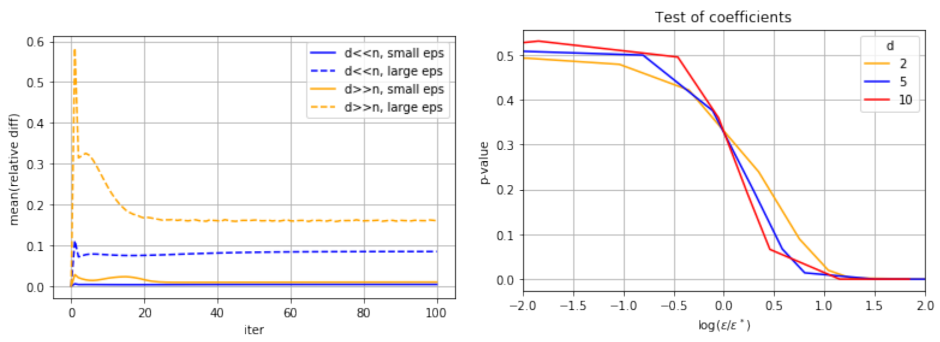

Figure 2: Left: the difference between clean training and adversarial training throughout the training process. The relative difference is calculated as $\|\theta^{(t)}(0) - \theta^{(t)}(\epsilon)\|/\|\theta^{(t)}(0)\|$. The difference gets larger when increasing ϵ. Right: The p-value for hypothesis testing to test whether $\theta^{(T)}(\epsilon)$ follows the same asymptotic distribution as $\theta^{(T)}(0)$. For all setups, the p-value starts to dramatically decrease when $\log(\epsilon/\epsilon^*) \approx 0$.

Remark 1. *Theorems 2 and 1 reveal a negative relationship between ϵ^* and n, i.e., larger n leads to a larger discrepancy between adversarial training and cleaning training. However, it does not imply that a larger n hurts the adversarial training. With larger n, the adversarial training may act different to clean training, but by the Law of Large Numbers, a large n can force the adversarial training to converge to the correct place.*

Remark 2. *The concept of ϵ^* is for adversarial training, rather than commonly used data augmentation, e.g., via adding Gaussian noise (Reed and MarksII, 1999). The noisy sample is randomly allocated around the original sample. As a result, it does not hurt the training too much when considering the average effect of Gaussian noise, which is not the case for the adversarial attack.*

Simulation Evidences We use a simulation study to demonstrate the above observations in the theorems. In particular, we would like to validate numerically (1) the adversarial training parameter is similar to the one in the clean training when $\epsilon \ll \epsilon^*$ and vice versa, and (2) the adversarial training loss is similar to the clean training loss when $\epsilon \ll \epsilon^*$ and vice versa.

To verify (1), we calculate $\|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\|/\|\theta^{(t)}(0)\|$ for each step $t = 1, ..., 100$, and repeat this experiment 100 times to obtain the average. Due to page limit, we postponed the detailed configurations to the appendix. As shown in Figure 2, when the attack strength is large, there is a great difference between the clean trained model parameters and adversarial trained model parameters. Note that in left panel in Figure 2, we only compare the curves for the same (d, n). The comparison among different (d, n) is not meaningful because the attack strengths are different.

Besides, we conduct hypothesis testing to check whether $\theta^{(T)}(\epsilon)$ is statistically significantly different from $\theta^{(T)}(0)$. We repeat clean training 300 times to obtain the mean and variance of $\theta^{(T)}(0)$, and then calculate the p-value of $\theta^{(T)}(\epsilon)$. The p-value represents "the probability of obtaining test results at least as extreme as the result actually observed, under the assumption that the null hypothesis is correct" (Wikipedia). A close-to-zero p-value means that $\theta^{(T)}(\epsilon)$ is significantly different from $\theta^{(T)}(0)$. We take $n = 100$ and $d \in \{2, 5, 10\}$ in this experiment. The results are shown in the right panel of Figure 2, and one can see that there is a dramatic change in the p-value when ϵ is around ϵ^*, i.e. $\log(\epsilon/\epsilon^*) \approx 0$.

To verify (2), we also repeat 300 times of clean training to get the distribution of $\widehat{R}(\theta^{(T)}, 0)$ and use this distribution to test whether the adversarial training result is from this distribution. As shown in Figure 3, similar to Figure 2, when the attack strength exceeds ϵ^*, there is a significant difference between the distributions of the losses, and the p-value is close to zero.

Two-Layer Neural Networks While the linear regression problem enjoys the above properties, following Ba et al. (2020); Xing et al. (2021b), the two-layer neural network with vanishing initialization can be proved to share similar properties as follows:

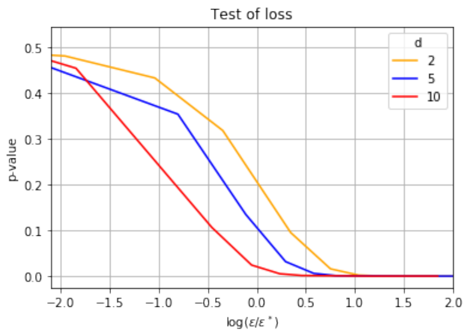

Figure 3: The p-value for hypothesis testing to test whether $\widehat{R}(\theta^{(T)}(\epsilon), \epsilon)$ follows the same asymptotic distribution as $\widehat{R}(\theta^{(T)}(0), 0)$. For all setups, the p-value is close to 0 when $\log(\epsilon/\epsilon^*) \approx 0$.

Proposition 1 (Two-layer neural networks)**.** *Consider a two-layer neural network with smooth activation function ϕ*

$$f(x) = \frac{1}{\sqrt{h}} \sum_{j=1}^{h} \phi(x^\top \theta_j) a_j \tag{3}$$

with h as the number of hidden nodes. When $h \to \infty$, with vanishing initialization of θ_j and a_j, if we fix the second layer (a_j) and only train the hidden layer (θ_j), the network parameter satisfies similar (but not the same[4]) property as Theorem 1, and the loss results are the same as Theorem 2.

The details of Proposition 1 and the main idea of the proof are postponed to the appendix.

5 Experiments in Deep Neural Networks

In this section, we study the performance of the proposed ϵ^* in terms of (1) whether our choice of ϵ^* is reasonable in deep learning, and (2) what implications the ϵ^* can bring towards commonly used configurations in deep learning.

5.1 General Configurations

Here we describe some general setups in the implementation for both this section and the next section.

For datasets CIFAR-10, CIFAR-100, SVHN, we modified the code of Rice et al. (2020) for our implementation. We keep all the existing configurations (optimizer, neural network architecture, transformer) from the original code. In particular, if there is no specification, we use an SGD optimizer with batch size 128 to train on the full training set for 200 epochs. The learning rate is initialized as 0.1 for CIFAR and 0.01 for SVHN, and is divided by 10 at the 100th and 150th epochs. We consider PreActResNet18 and WideResNet34 as those in Rice et al. (2020). After training 200 epochs, we find the epoch with the smallest adversarial testing loss as the final model for early stopping. For MNIST, we implement a CNN with two convolution layers and two fully connected layers.

Besides, we also use the loss TRADES and MART in Zhang et al. (2019) and Wang et al. (2019b).

5.2 Verifying the effectiveness of ϵ^*

In this section, we present some metrics of adversarial training to argue that our choice of ϵ^* is reasonable. We use CIFAR-10 with PreActResNet18 and $\mathcal{L}_\infty$ in this section.

To obtain ϵ^*, we first run a clean training to get the clean testing loss, then run adversarial training for a wide range of ϵ's and perform a linear interpolation to obtain ϵ^*. For both clean training and adversarial training, we train from scratch and summarize the results in Table 1. From Table 1, one

[4]We cannot directly compare the parameters of the neural networks trained from clean training and adversarial training, but the output predictive models are indeed similar when ϵ is small enough.

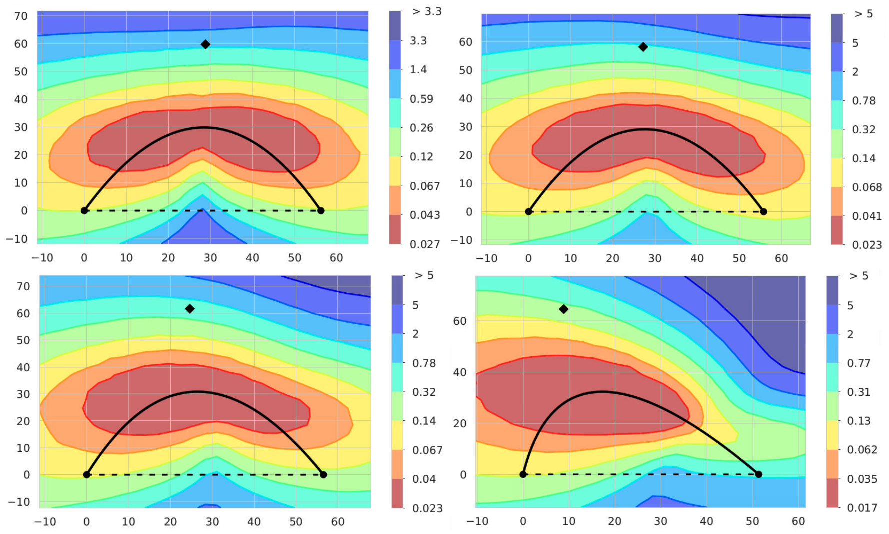
Figure 4: Connectivity between a clean trained model and adversarial robust models in terms of their clean training loss. Upper left: $\epsilon = 0$. Upper right: $\epsilon = 0.5$. Lower left: $\epsilon = 2$. Lower right: $\epsilon = 8$. When increasing ϵ, the right point (the checkpoint for adversarial training) is away from the basin.

can see that the clean testing loss for clean training is 0.1837. The threshold ϵ^* is the attack strength such that the adversarial training loss is 0.1837. From the table, when $\epsilon = 0.5$, the adversarial training loss is 0.1802, implying that ϵ^* is approximately 0.5.

Table 1: Adversarial training in PreActResNet18 using CIFAR-10.

ϵ	Epoch	Adv Training		Adv Testing	
		loss	acc	loss	acc
0	104	0.0761	0.9743	**0.1837**	0.9405
0.5/255	101	**0.1802**	0.9376	0.2937	0.8977
1/255	102	0.2148	0.9227	0.371	0.8723
2/255	102	0.3411	0.8722	0.5218	0.8101

We conduct comparisons in some aspects, e.g., connectivity, overfitting, and FGM catastrophic overfitting, to show that the adversarial training with $\epsilon \ll \epsilon^*$ is more similar to the clean training compared to stronger attacks.

Connectivity The connectivity of deep neural networks aims to answer whether there is a path in the parameter space between two deep neural networks (of the same architecture), such that all the neural networks along this path have good prediction performance. It is a useful tool to study the loss landscape of deep neural networks, e.g., Chao et al. (2020). Good connectivity implies that the two neural networks are in the same "basin" of the loss.

To numerically figure out a path, we use the method in Garipov et al. (2018). In order to have a graphical presentation, we take `num_bend` as 3, i.e., besides the start and the end neural networks, there is only one neural network to be trained in the path parameters. Figure 4 shows the connectivity between the clean trained model and the robust model. In each subfigure, the left point represents the clean model at the 101st epoch, the right point is the adversarially trained model at the 101st epoch, and the upper point is the extra checkpoint to be trained. We take the 101st epoch because the adversarial training generally achieves the best adversarial testing loss around the 101st epoch. These three points together determine the black curve in the figure, and they are in the same 2D plane. We calculate the clean training loss for other points in this plane to obtain the contour.

From Figure 4, one can observe that when $\epsilon = 0$ and $\epsilon = 0.5$, the low-loss region is almost symmetric to the lower two points. The middle part is also well connected, i.e., most of the connecting path belongs to the low-loss region (red area). When $\epsilon > 0.5$, the low-loss region gradually shifts towards the clean model. This indicates that the adversarial training gradually moves out of the "basin".

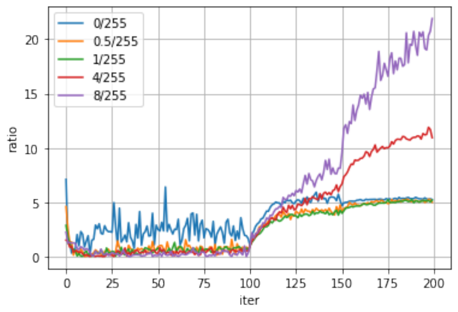

Figure 5: Overfitting in clean training and adversarial training. We use the ratio of generalization error over the one at the 101st epoch to examine the overfitting level, where generalization error is the difference between adversarial training and testing loss. When $\epsilon \leq 1$, this ratio is small. Results from other attack strengths, e.g. 2/255 and 16/255, are not included in this figure to keep the plot clear, and they do not alter the main observations.

Generalization and Overfitting In the literature, it is observed that in deep neural networks with clean training, overfitting training data does not affect the generalization performance. Some studies, e.g., Belkin et al. (2018, 2019a); Li et al. (2021); Chatterji et al. (2021), provide theoretical justifications towards this phenomenon.

Our numerical results show no harmful overfitting in clean training, and the overfitting in adversarial training with ϵ^* is also not severe. As shown in Figure 5, when ϵ gets larger, the ratio (generalization gap)/(generalization gap at the 101st epoch) increases after the 100th epoch.

FGSM and Catastrophic Overfitting Based on Andriushchenko and Flammarion (2020), FGSM (i.e., the FGM approximation of $\mathcal{L}_\infty$-PGD attack) leads to catastrophic overfitting when attack strength is large, yielding an almost-zero adversarial testing accuracy using $\mathcal{L}_\infty$-PGD attack. Since FGM is only an approximation of PGD, the FGM attack is not as strong as the PGD attack, and its adversarial testing performance under PGD attack is expected to be worse. However, for $\mathcal{L}_\infty$ attack, the performance drop in testing accuracy is far more severe, and Andriushchenko and Flammarion (2020) reveals some threats in FGSM which cause this phenomenon. This catastrophic overfitting is a second type of overfitting different from the one discussed in the previous paragraph.

Based on our intuition of ϵ^*, the above concern does not hurt the training process when $\epsilon < \epsilon^*$. First, since ϵ is small, the difference between FGM and PGD is small. Second, since adversarial training is similar to clean training and clean training does not suffer from severe over-fitting problems, those problems can be avoided. To verify this, we train CIFAR-10 using 10000/50000 samples with PGD/FGM adversarial training in different levels of $\mathcal{L}_\infty$ attacks. We take ϵ as 1/255, 2/255, 4/255, 8/255, 12/255, 16/255 to see whether it obtains a large adversarial testing loss with an almost-zero adversarial testing accuracy.

The results are summarized in Table 2. The column "ϵ^*" shows the critical ϵ^*, and the column "Unstable ϵ" is the minimal attack strength when FGSM starts the catastrophic over-fitting. One can see that ϵ^* is always smaller than the unstable threshold. This is intuitive because adversarial training is still similar to clean training with $\epsilon \approx \epsilon^*$. On the other hand, this also implies that one can use FSGM to search for ϵ^*.

Table 2: Catastrophic overfitting in FGSM adversarial training in CIFAR-10 using PreActResNet18.

n	norm	ϵ^*	Unstable ϵ
10K	$\mathcal{L}_\infty$	3.3/255	12/255
50K	$\mathcal{L}_\infty$	0.5/255	8/255

5.3 Observations in common settings

The previous section conducts numerical experiments to justify our choice of ϵ^*. In this section, we use this method in various datasets and neural network architectures to provide more insights and study how ϵ^* is affected by these factors.

Table 3: The value of ϵ^* in different datasets, neural network architectures, and training sample size. "PAResNet" and "WResNet" refer to "PreActResNet" and "WideResNet" respectively to save the margin.

Dataset	Architecture	n	$\epsilon^*(\mathcal{L}_\infty)$	Dataset	Architecture	n	$\epsilon^*(\mathcal{L}_\infty)$
CIFAR-10	PAResNet18	50K	0.5/255	MNIST	MLP(128)	50K	48.0/255
CIFAR-10	PAResNet18	10K	3.3/255	MNIST	MLP(16)	50K	26.1/255
CIFAR-10	WResNet34-1	50K	1.1/255	SVHN	PAResNet18	50K	4.9/255
CIFAR-10	WResNet34-10	50K	1.1/255				
CIFAR-100	PAResNet18	25K	4.0/255				
CIFAR-100	PAResNet18	50K	1.0/255				

5.3.1 General Observations

An important observation in Table 3 is that the commonly used attack strengths are greater than ϵ^*. In the literature, we usually use $\mathcal{L}_\infty$ attack with strength $8/255$ for CIFAR-10/100 and SVHN, and 0.3 for MNIST. From our experiments, we observe that using all the 50000 training samples, the corresponding ϵ^* for CIFAR-10/100, SVHN, and MNIST are $0.5/255$, $1.0/255$, $4.9/255$, $48.0/255$.

5.3.2 More Detailed Observations

Wide neural networks Different neural network architectures lead to different ϵ^*. For MNIST, using a simple neural network of two convolution layers and two fully connected layers, when there are 16 hidden nodes for the first FC layer, ϵ^* is 26.1/255. When there are 128 hidden nodes, it becomes 48/255. This observation verifies the argument in the existing literature that wider neural network structures are essential in adversarial training to enlarge the model capacity Xie and Yuille (2019); Author (2021). For CIFAR-10, ϵ^* changes little when expanding the width of the neural network from 1 to 10, implying that WideResNet34-10 may not be sufficient for adversarial training. Note that one need to be cautious when enlarging the network size, as wider neural networks tend to overfit the data even in clean training.

Improved adversarial loss Some literature tries to improve the adversarial training process via improving the loss. We would like to examine how ϵ^* changes along these loss functions as well. We run TRADES and MART in CIFAR-10 with PreActResNet18 for this experiment, and the ϵ^* are 0.7/255 and 0.8/255, respectively. Although 0.7/255 and 0.8/255 are larger than 0.5/255 (i.e., ϵ^* for vanilla adversarial training; See Table 3), they are still less than 8/255.

Training size From Table 3, for both CIFAR-10 and CIFAR-100, a larger training size indicates a smaller ϵ^*. There are two reasons for this phenomenon. First, when enlarging the sample size, the generalization gap gets smaller; thus, ϵ^* is smaller. Second, as observed in Author (2021), when increasing the training size, the neural network has a larger norm, implying that a larger model capacity is needed to fit the attacked samples. As a result, the neural network architecture cannot handle the adversarial training properly, so the corresponding ϵ^* is smaller.

6 Faster Approximation of ϵ^*

To compute ϵ^*, a naive way is to repeatedly perform the adversarial training for each possible ϵ until it hits the threshold. This could be very time-consuming when solving the adversarial training from scratch (i.e., initializing from a near-zero random start). For instance, it takes 20 minutes to train the CIFAR-10 dataset for clean training but takes 10 hours to complete adversarial training for a given ϵ. Hence we propose to speed up this process via a linear extrapolation approximation.

Due to the page limit, the intuition, potential difficulties, and final algorithm for approximating ϵ^* are postponed to Section A in the appendix. Briefly speaking, due to good connectivity between the clean and adversarial training when $\epsilon \ll \epsilon^*$ as in Figure 4, we consider using adversarial fine-tuning (Chen et al., 2020b) to replace the whole adversarial training (from scratch) process in the grid search for ϵ^*. To overcome the potential over-fitting and learning rate tuning problems, we conduct adversarial fine-tuning under similar ϵ's and use extrapolation to approximate ϵ^*. Since we are using extrapolation rather than interpolation, we also provide theoretical support to justify the correctness

Table 4: Performance of fine tuning. $\widehat{\epsilon}$(FGM,a) represents that the adversarial fine-tuning uses FGM attack with a runs of fine tuning.

n	norm	ϵ^*	$\widehat{\epsilon}$(PGD,1)	$\widehat{\epsilon}$(PGD,3)	$\widehat{\epsilon}$(FGM,1)	$\widehat{\epsilon}$(FGM,3)
10000	$\mathcal{L}_\infty$	3.3/255	4.7/255	2.5/255	1.7/255	2.1/255
50000	$\mathcal{L}_2$	24.8/255	20.1/255	17.9/255	18.4/255	17.4/255
50000	$\mathcal{L}_\infty$	0.5/255	0.58/255	0.52/255	0.53/255	0.59/255

of the algorithm. The numerical results are shown in Table 4. One can see that the proposed algorithm gives a good estimate of ϵ^*.

7 What Can We Do If $\epsilon \gg \epsilon^*$?

Our paper focuses on how to determine ϵ^* and the consequence of $\epsilon \gg \epsilon^*$, and it is beyond our scope to study how to adjust adversarial training when $\epsilon \gg \epsilon^*$. We provide some potential solutions.

In general, when $\epsilon \gg \epsilon^*$, one needs to overcome a series of potential problems in adversarial training. In the literature, methods such as MART Wang et al. (2019b), Dynamic Wang et al. (2019a), FAT Zhang et al. (2020a), HAT Rade and Moosavi-Dezfooli (2021), smoothing Xie et al. (2020); Xing et al. (2021b) can overcome some of the problems. However, they may not be sufficient to resolve every problem caused by the fundamental gap between adversarial training and cleaning training revealed by this work. Alternatively, we suggest two other ways: (1) adjusting the neural network architecture to enlarge ϵ^*, and (2) utilizing more information/data to force adversarial training to converge to the correct place.

For (1), in our numerical experiment, enlarging the size of the neural network can make ϵ^* larger. If controlling the over-fitting issue properly, this can be a solution to improve adversarial robustness. Similarly, empirical experiments in various literature (e.g., Xie and Yuille, 2019; Rice et al., 2020; Gowal et al., 2021) show that wider neural networks lead to better performance. Some other studies (e.g., Huang et al., 2021) also study how the neural network architecture affects robustness.

For (2), there are several ways to utilize more information from the data. For example, Gowal et al. (2021) trains a clean classifier and an unlabeled data generator to generate extra synthetic data, which facilitate the adversarial training. This framework does not introduce any new data source but improves adversarial robustness, implying that vanilla adversarial training overlooks some information from the data. Although ϵ^* may be smaller than the actual ϵ, using more information/data can force the training to converge to the correct place. Similar studies can be found in Carmon et al. (2019); Xing et al. (2021c).

8 Conclusion

Observing the great gap between empirical results in adversarially robust models and the theoretical studies in this area, we conjecture that adversarial training acts differently from our common understanding of clean training in deep learning. Through intuitions in simple statistical models, we design a metric of similarity to determine whether adversarial training is "similar" to clean training or not. Our results reveal that the commonly used adversarial training setups in literature take a large attack strength so that it is different from clean training given the current neural network architectures and data. We reveal some potential factors which affect ϵ^*. Besides, since adversarial training from scratch is time-consuming, we propose to use adversarial fine-tuning and extrapolation to do approximate ϵ^*. Such a method can reduce 80% running time compared to adversarial training from scratch while leading to reasonable estimates.

A future direction of this work is to theoretically understand how the neural network architecture affects ϵ^*, and based on which, to provide proper guidance on architecture selection that accommodates stronger attacks (e.g., the commonly used 8/255 attack strength in practice).

Acknowledgements

This project is partially supported by NSF-SCALE MoDL (2134209) and ONR N00014-22-1-2680.

References

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

Andriushchenko, M. and Flammarion, N. (2020), “Understanding and improving fast adversarial training,” *arXiv preprint arXiv:2007.02617*.

Author, A. (2021), “Adversarial Rademacher Complexity of Deep Neural Networks,” .

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*.

Baratin, A., George, T., Laurent, C., Devon Hjelm, R., Lajoie, G., Vincent, P., and Lacoste-Julien, S. (2020), “Implicit Regularization in Deep Learning: A View from Function Space,” *arXiv e-prints*, arXiv–2008.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020), “Benign overfitting in linear regression,” *Proceedings of the National Academy of Sciences*, 117, 30063–30070.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019a), “Reconciling modern machine learning and the bias-variance trade-off,” *Proceedings of the National Academy of Sciences*, 116, 15849–15854.

Belkin, M., Hsu, D., and Mitra, P. (2018), “Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate,” *arXiv preprint arXiv:1806.05161*.

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

Bubeck, S. and Sellke, M. (2021), “A Universal Law of Robustness via Isoperimetry,” *arXiv preprint arXiv:2105.12806*.

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.

Chao, S.-K., Wang, Z., Xing, Y., and Cheng, G. (2020), “Directional Pruning of Deep Neural Networks,” *arXiv preprint arXiv:2006.09358*.

Chatterji, N. S., Long, P. M., and Bartlett, P. L. (2021), “The Interplay Between Implicit Bias and Benign Overfitting in Two-Layer Linear Networks,” *arXiv preprint arXiv:2108.11489*.

Chen, L., Min, Y., Zhang, M., and Karbasi, A. (2020a), “More data can expand the generalization gap between adversarially robust and standard models,” *arXiv preprint arXiv:2002.04725*.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., and Wang, Z. (2020b), “Adversarial robustness: From self-supervised pre-training to fine-tuning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 699–708.

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.

Du, S. S. and Lee, J. D. (2018), “On the power of over-parametrization in neural networks with quadratic activation,” in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, vol. 80 of *Proceedings of Machine Learning Research*, pp. 1329–1338.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. (2019), “Gradient descent finds global minima of deep neural networks,” in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, vol. 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018), "Gradient descent provably optimizes over-parameterized neural networks," *arXiv preprint arXiv:1810.02054*.

Frederickson, C., Moore, M., Dawson, G., and Polikar, R. (2018), "Attack strength vs. detectability dilemma in adversarial machine learning," in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 1–8.

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*.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., and Wilson, A. G. (2018), "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8803–8812, https://github.com/timgaripov/dnn-mode-connectivity.

Gong, Z., Wang, W., and Ku, W.-S. (2017), "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*.

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*.

Hu, T., Shang, Z., and Cheng, G. (2020), "Sharp Rate of Convergence for Deep Neural Network Classifiers under the Teacher-Student Setting," *arXiv preprint arXiv:2001.06892*.

Huang, H., Wang, Y., Erfani, S., Gu, Q., Bailey, J., and Ma, X. (2021), "Exploring Architectural Ingredients of Adversarially Robust Deep Neural Networks," *Advances in Neural Information Processing Systems*, 34.

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," *arXiv preprint arXiv:2002.10477*.

Kohler, M. and Langer, S. (2019), "On the rate of convergence of fully connected very deep neural network regression estimates," *arXiv preprint arXiv:1908.11133*.

Lee, K. and Chandrakasan, A. P. (2020), "Rethinking Empirical Evaluation of Adversarial Robustness Using First-Order Attack Methods," *arXiv preprint arXiv:2006.01304*.

Li, Z., Zhou, Z.-H., and Gretton, A. (2021), "Towards an Understanding of Benign Overfitting in Neural Networks," *arXiv preprint arXiv:2106.03212*.

Mehrabi, M., Javanmard, A., Rossi, R. A., Rao, A., and Mai, T. (2021), "Fundamental Tradeoffs in Distributionally Adversarial Training," *arXiv preprint arXiv:2101.06309*.

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*.

Neyshabur, B. (2017), "Implicit regularization in deep learning," *arXiv preprint arXiv:1709.01953*.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2017a), "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks," *arXiv preprint arXiv:1707.09564*.

Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. (2017b), "Geometry of optimization and implicit regularization in deep learning," *arXiv preprint arXiv:1705.03071*.

Rade, R. and Moosavi-Dezfooli, S.-M. (2021), "Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off," in *ICML 2021 Workshop on Adversarial Machine Learning*.

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

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. (2021), "Fixing data augmentation to improve adversarial robustness," *arXiv preprint arXiv:2103.01946*.

Reed, R. and MarksII, R. J. (1999), *Neural smithing: supervised learning in feedforward artificial neural networks*, Mit Press.

Rice, L., Wong, E., and Kolter, J. Z. (2020), "Overfitting in adversarially robust deep learning," *arXiv preprint arXiv:2002.11569*, https://github.com/locuslab/robust_overfitting.

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

Taheri, H., Pedarsani, R., and Thrampoulidis, C. (2020), "Asymptotic Behavior of Adversarial Training in Binary Classification," *arXiv preprint arXiv:2010.13275*.

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

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*, https://github.com/YisenWang/MART.

Wu, B., Chen, J., Cai, D., He, X., and Gu, Q. (2020), "Does Network Width Really Help Adversarial Robustness?" *arXiv preprint arXiv:2010.01279*.

Xie, C., Tan, M., Gong, B., Yuille, A., and Le, Q. V. (2020), "Smooth adversarial training," *arXiv preprint arXiv:2006.14536*.

Xie, C. and Yuille, A. (2019), "Intriguing properties of adversarial training at scale," *arXiv preprint arXiv:1906.03787*.

Xing, Y., Song, Q., and Cheng, G. (2021a), "On the algorithmic stability of adversarial training," *Advances in Neural Information Processing Systems*, 34, 26523–26535.

— (2021b), "On the generalization properties of adversarial training," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 505–513.

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

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. (2018), "Hessian-based analysis of large batch training and robustness to adversaries," *arXiv preprint arXiv:1802.08241*.

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

Zhai, R., Cai, T., He, D., Dan, C., He, K., Hopcroft, J., and Wang, L. (2019), "Adversarially robust generalization just requires more unlabeled data," *arXiv preprint arXiv:1906.00555*.

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, https://github.com/yaodongyu/TRADES.

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, 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 authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
    (b) Did you describe the limitations of your work? [Yes]
    (c) Did you discuss any potential negative societal impacts of your work? [No] This is a theoretical study on existing methodologies. It does not introduce new methods in the real-world applications.
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? [Yes]
    (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the github links for the original code we modified from.
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We mention the time when introducing our speed-up algorithm.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
    (a) If your work uses existing assets, did you cite the creators? [Yes]
    (b) Did you mention the license of the assets? [N/A]
    (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

There are three sections in the appendix.

In Section A, we present how to approximate ϵ^*. Theorems and experiments are provided to verify the performance of the algorithm.

In Section B, we present the experiment details postponed from the main text. In particular, we present some details about the simulation in Section 4. Besides, we present the full-size connectivity figures, the training results using different ϵ for obtaining ϵ^*, fine-tuning detailed results as well as the learning rate schedule used in fine-tuning.

In Section C, we present the proofs of Theorem 1, Theorem 2, Proposition 1, and Proposition 2.

A Speed up the Approximation

In this section, we state the possibility (feasibility) to speed up the approximation using adversarial fine-tuning, and describe what potential problems we have in this method, as well as our solution and the numerical results.

Feasibility of Adversarial Fine-Tuning One may question whether fine-tuning (i.e., initializing from an estimated model trained under a similar setup) can achieve our goal since there is no guarantee that adversarial training from scratch converges to the same places as adversarial fine-tuning. We use connectivity to illustrate the feasibility of the fine-tuning method. Based on Figure 4 in Section 5.2, when taking a small ϵ, the connectivity between the clean model and adversarially robust model (trained from scratch) is good. This good connectivity implies that solving ϵ^* via adversarial training from scratch and adversarial fine-tuning lead to similar estimates.

A Practical Problem Following Chen et al. (2020b), one can obtain an adversarially robust model via fine-tuning. However, some simple trials reveal two potential problems:

1. if we fine-tune from a clean model, i.e., $\theta^{(0)}(\epsilon) = \theta^{(t)}(0)$, the learning rate schedule that works for small ϵ is not suitable for large ϵ;
2. if we sequentially fine-tune the model, i.e., $\theta^{(0)}(\epsilon_{i+1}) = \theta^{(t)}(\epsilon_i)$ for an increasing sequence of ϵ_i's, it can be easily trapped by a bad local minimum which overfits the training data and yields zero adversarial training loss for large ϵ.

Consequently, it is hard to provide a fully automatic algorithm to solve the exact ϵ^* without manually tuning the learning rate schedule.

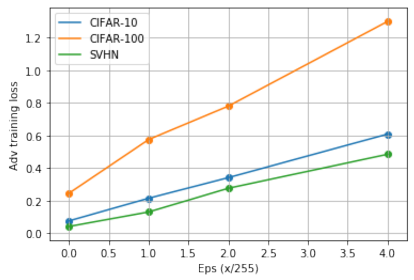

Figure A.1: The adversarial training loss (with early stopping) for three real datasets. The loss is approximately a linear function in the attack strength.

Solution To overcome the above practical problem, we propose to use adversarial fine-tuning for only a small number of ϵ's whose values are close, such that the same learning rate schedule can be adopted. After obtaining the adversarial fine-tuning results for these ϵ's, we then use regression extrapolation to approximate ϵ^*. This process can significantly reduce the computation and time cost of finding ϵ^*.

The success of the above approximation idea relies on how accurate the extrapolation is. By the definitions of adversarial attack and adversarially robust models, the adversarial risk is a monotone increasing function w.r.t. the attack strength ϵ. If we plot ϵ against the corresponding adversarial training loss $\widehat{R}(\widehat{\theta}(\epsilon),\epsilon)$ (trained from scratch), the curve is almost a linear function as in Figure A.1. Based on this observation, linear (or isotonic if necessary) extrapolation with adversarial fine-tuning gives a reasonable approximation.

The following Algorithm A.1 describes the above idea:

Algorithm A.1 Solve ϵ^*

```
Input: the trained clean model $\widehat{\theta}(0)$, data $\{(x_i,y_i)\}_{i=1}^n$, learning rate schedule $\{\eta_t\}$, attack strength $\{\epsilon_j\}$.
for $\epsilon = \epsilon_1, \epsilon_2, \epsilon_3, ...$ do
    Use adversarial fine tuning with $\epsilon$ to obtain the minimal adversarial training loss when minimizing adversarial testing loss. Denote the adversarial model as $\widehat{\theta}(\epsilon)$.
end for
Use $(\epsilon, \widehat{R}(\widehat{\theta}(\epsilon),\epsilon))$ for $\epsilon = 0, \epsilon_1, \epsilon_2, \epsilon_3, ...$ to find an approximation $\widehat{\epsilon}$ of $\epsilon^*$.
```

Algorithm A.1 illustrates the general idea, while we do not explicitly specify the exact number of fine-tunes and the type of approximation method used. One can use the clean model and one fine-tuned single ϵ_1 to conduct the linear extrapolation, or several models for linear regression or nonlinear extrapolation (e.g., monotone splines). For the experiment presented in this paper, we choose linear regression.

The following proposition illustrates the effectiveness of Algorithm A.1:

Proposition 2. *Consider the linear models in Theorem 2.*

Adversarial fine tuning satisfies that, if $\|\theta^{(t)}\| \leq r$ for all t for some $r > 0$, then

$$\mathbb{E}\left[\min_{t=1,\dots,T} \widehat{R}(\theta^{(t)}) - \widehat{R}(\widehat{\theta}) \middle| S\right] \leq \frac{\mathbb{E}\|\theta^{(0)} - \widehat{\theta}\|^2}{2\eta T} - \frac{\mathbb{E}[\|\theta^{(T)} - \widehat{\theta}\|^2 | S]}{2\eta T} + O\left(\frac{\eta (L_r)^2}{2}\right),$$

where L_r is the Lipschitz constant of $\widehat{R}$ for $\theta \in B_2(0,r)$.

Given sufficiently small η and large T, the above optimization error will be negligible, then the approximation error involved in $\widehat{\epsilon}$ is $o(\epsilon^*)$ if all ϵ and ϵ^* are $o(1)$.

Proposition 2 shows the convergence of the proposed algorithm, and the proof is in Section C.

Proposition 2 itself is applicable to any proper initialization of fine-tuning. Compared to training from scratch, since the trained clean model already has a small adversarial training loss, taking $\widehat{\theta}(0)$ as the initialization of Algorithm A.1, the number of iterations used in the fine-tuning is much less.

Besides, Algorithm A.1 only requires small-ϵ adversarial training, based on our observations in Section 5 for the catastrophic overfitting, using FGM is also a fast and effective approximation if we replace it with PGD attack in the adversarial fine-tuning when $\epsilon = O(\epsilon^*)$.

Numerical Illustration This section aims to verify that (1) the proposed method with extrapolation is still reasonable, and (2) using FGM leads to a good estimate of ϵ^*.

After obtaining the clean model, we use 40 epochs to do the adversarial fine-tuning for each ϵ, so there will be at least 80% reduction on the computation time compared to adversarial training from scratch. We use the same base learning rate as the one in clean training. We use an increasing and decreasing learning rate schedule in the fine-tuning to adjust the learning rate. The detailed learning rate schedule and the choices of ϵ we take (together with the loss values) are in Section B.

The results are summarized in Table 4. We consider both PGD and FGM in our experiments. In Table 4, we present the two adversarial fine-tuning algorithms using PGD and FGM, respectively, and the all one- and three-points approximation for ϵ^*. All the settings lead to good approximation.

B Experiment Details

B.1 Simulation in Section 4

Table B.1 shows the the detailed configurations for the experiment for the left panel of Figure 2.

Table B.1: Detailed parameter setups for Figure 2

Regime	d	n	ϵ^*	Small ϵ	Large ϵ
$d \gg n$	20	1000	0.02	0.01	0.2
$d \ll n$	2000	10	15	1	20

B.2 Details for ϵ^*

To obtain ϵ^* in Table 2, 4, and 3, we use adversarial training to train on different ϵs from scratch, then use linear interpolation to approximate ϵ^*. We paste the detailed training results here in Table B.2, B.3, B.4, B.5, and B.6. Table B.2 collects the results for CIFAR-10 dataset using different training sizes, batch sizes, and attack norms. Table B.3 is the result for other datasets. Table B.4 is the result for CIFAR-10 using wide neural networks, and Table B.5 is the result of CIFAR-10 using Adam optimizer, or TRADES/MART loss. Table B.6 summarizes the performance of FGM in adversarial training using CIFAR-10.

Table B.2: Adversarial training in CIFAR-10 (PreActResNet) using different training sample size (10000, 50000), batch size (128, 256), attack norm ($\mathcal{L}_\infty$, $\mathcal{L}_2$).

				Train		Test	
N	ϵ(/255)	Batch size	Norm	Adv loss	Adv acc	Adv loss	Adv acc
10000	0	128	$\mathcal{L}_\infty$	0.0403	0.9894	0.441	0.8717
10000	1	128	$\mathcal{L}_\infty$	0.1586	0.9462	0.879	0.734
10000	2	128	$\mathcal{L}_\infty$	0.2199	0.9269	1.1723	0.6555
10000	8	128	$\mathcal{L}_\infty$	1.2186	0.5114	1.7621	0.3723
10000	32	128	$\mathcal{L}_2$	0.0777	0.9778	0.9131	0.7396
10000	64	128	$\mathcal{L}_2$	0.2001	0.9339	1.1616	0.6621
10000	96	128	$\mathcal{L}_2$	0.2681	0.908	1.4464	0.57
10000	128	128	$\mathcal{L}_2$	0.3494	0.8743	1.5258	0.5263
10000	256	128	$\mathcal{L}_2$	0.8098	0.6826	1.8491	0.3773
10000	512	128	$\mathcal{L}_2$	1.6061	0.3789	2.0156	0.2782
50000	0	128	$\mathcal{L}_\infty$	0.0761	0.9743	0.1837	0.9405
50000	0.5	128	$\mathcal{L}_\infty$	0.1802	0.9376	0.2937	0.8977
50000	1	128	$\mathcal{L}_\infty$	0.2148	0.9227	0.371	0.8723
50000	2	128	$\mathcal{L}_\infty$	0.3411	0.8722	0.5218	0.8101
50000	4	128	$\mathcal{L}_\infty$	0.6082	0.7631	0.8079	0.6972
50000	8	128	$\mathcal{L}_\infty$	1.1102	0.5645	1.238	0.5291
50000	0	256	$\mathcal{L}_\infty$	0.0263	0.9924	0.1823	0.9469
50000	1	256	$\mathcal{L}_\infty$	0.1769	0.9389	0.4179	0.8582
50000	2	256	$\mathcal{L}_\infty$	0.2779	0.8972	0.5858	0.8022
50000	4	256	$\mathcal{L}_\infty$	0.5181	0.7991	0.8656	0.6924
50000	32	128	$\mathcal{L}_2$	0.2149	0.9212	0.3739	0.8675
50000	64	128	$\mathcal{L}_2$	0.34	0.8702	0.5362	0.8038
50000	96	128	$\mathcal{L}_2$	0.4698	0.8159	0.6775	0.7484
50000	128	128	$\mathcal{L}_2$	0.5963	0.7644	0.8038	0.7012

B.3 Adversarial Fine-tuning

Similar to solving ϵ^*, when using adversarial fine-tuning, we also runs the fine-tuning for some different ϵs. The detailed results for each ϵ are summarized in Table B.7.

Table B.3: Adversarial training using other datasets (CIFAR-100, SVHN) (PreActResNet18).

Dataset	N	ϵ(/255)	Adv train loss	Adv train acc	Adv test loss	Adv test acc
CIFAR-100	25000	0	0.2459	0.935	1.2872	0.667
CIFAR-100	25000	1	0.5751	0.8322	2.1848	0.4938
CIFAR-100	25000	2	0.7798	0.7737	2.5653	0.4029
CIFAR-100	25000	4	1.2994	0.634	3.0408	0.312
CIFAR-100	25000	8	2.7874	0.2804	3.5278	0.1899
CIFAR-100	50000	0	0.4647	0.8617	0.9047	0.7448
CIFAR-100	50000	1	0.8917	0.7335	1.4699	0.5941
CIFAR-100	50000	2	1.383	0.6041	1.7852	0.5126
CIFAR-100	50000	4	1.7344	0.517	2.2901	0.4032
CIFAR-100	50000	8	2.6004	0.3356	2.9453	0.2834
SVHN	50000	0	0.0006	1	0.2303	0.9438
SVHN	50000	2	0.0016	0.9999	0.7909	0.8177
SVHN	50000	4	0.0931	0.9648	1.4166	0.6563
SVHN	50000	8	0.7251	0.7419	1.6793	0.4806

Table B.4: The performance of adversarial training in CIFAR-10 (WideResNet34, $N = 50000$) using different width factor.

Width	ϵ(/255)	Adv train loss	Adv train acc	Adv test loss	Adv test acc
1	0	0.0476	0.986	0.2261	0.934
1	1	0.2061	0.9266	0.4645	0.8402
1	2	0.3782	0.8594	0.6365	0.7723
10	0	0.0416	0.9872	0.1505	0.954
10	1	0.1307	0.9546	0.3311	0.8892
10	2	0.2759	0.8997	0.4603	0.8329

Table B.5: The performance of adversarial training in CIFAR-10 (PreActResNet18, $N = 50000$) using other optimizer (Adam) or loss function (TRADES, MART).

Name	ϵ(/255)	Adv train loss	Adv train acc	Adv test loss	Adv test acc
Adam	0	0.0307	0.9896	0.3554	0.9204
Adam	1	0.0767	0.972	0.6983	0.8224
Adam	2	0.2832	0.8883	0.9108	0.7257
Adam	4	0.9711	0.6289	1.1258	0.5972
MART	0	0.432	0.9195	0.3411	0.8856
MART	1	0.5989	0.8381	0.5627	0.8032
MART	2	0.8447	0.7687	0.7675	0.727
TRADES	0	0.1681	0.9613	0.2857	0.9044
TRADES	1	0.3212	0.825	0.5536	0.8111
TRADES	2	0.3007	0.7767	0.8246	0.7094

B.4 Fine-tuning learning rate

Figure B.1 shows the learning rate schedule used for adversarial fine-tuning. The values in Figure B.1 are multiplied by `lr_max` in training.

Table B.6: FGM adversarial training for CIFAR-10.

N	ϵ (/255)	Norm	Adv train loss	Adv train acc	Adv test loss	Adv test acc
10000	1	$\mathcal{L}_\infty$	0.1418	0.958	0.8212	0.7483
10000	2	$\mathcal{L}_\infty$	0.2066	0.9297	1.1251	0.6587
10000	4	$\mathcal{L}_\infty$	0.5863	0.7665	1.6388	0.4856
10000	8	$\mathcal{L}_\infty$	0.9963	0.5905	2.0131	0.3419
10000	12	$\mathcal{L}_\infty$	0.265	0.9151	31.7608	0
10000	16	$\mathcal{L}_\infty$	0.3133	0.8996	36.0938	0
10000	20	$\mathcal{L}_\infty$	0.1067	0.9666	36.4307	0
10000	32	$\mathcal{L}_\infty$	0.1253	0.9585	33.5073	0
50000	1	$\mathcal{L}_\infty$	0.1785	0.9381	0.3863	0.8682
50000	2	$\mathcal{L}_\infty$	0.2727	0.9005	0.5578	0.8022
50000	4	$\mathcal{L}_\infty$	0.4683	0.8195	0.8726	0.6839
50000	8	$\mathcal{L}_\infty$	0.3184	0.8981	29.4849	0
50000	12	$\mathcal{L}_\infty$	0.2812	0.9098	29.5383	0
50000	16	$\mathcal{L}_\infty$	0.2475	0.9205	39.2974	0
50000	20	$\mathcal{L}_\infty$	0.2216	0.9293	39.3613	0
50000	32	$\mathcal{L}_\infty$	0.2144	0.9315	33.7759	0

Table B.7: The adversarial training loss using adversarial fine-tuning.

Attack	N	ϵ(/255)	Norm	Adv loss	Attack	N	ϵ(/255)	Norm	Adv loss
None	10000	0	-	0.0403					
PGD	10000	0.25	$\mathcal{L}_\infty$	0.0426	FGM	10000	0.25	$\mathcal{L}_\infty$	0.0893
PGD	10000	0.5	$\mathcal{L}_\infty$	0.0829	FGM	10000	0.5	$\mathcal{L}_\infty$	0.1562
PGD	10000	1	$\mathcal{L}_\infty$	0.2008	FGM	10000	1	$\mathcal{L}_\infty$	0.2289
None	50000	0	-	0.0761					
PGD	50000	0.25	$\mathcal{L}_\infty$	0.1163	FGM	50000	0.25	$\mathcal{L}_\infty$	0.1183
PGD	50000	0.5	$\mathcal{L}_\infty$	0.1642	FGM	50000	0.5	$\mathcal{L}_\infty$	0.1739
PGD	50000	1	$\mathcal{L}_\infty$	0.288	FGM	50000	1	$\mathcal{L}_\infty$	0.2425
PGD	50000	5	$\mathcal{L}_2$	0.0999	FGM	50000	5	$\mathcal{L}_2$	0.0958
PGD	50000	10	$\mathcal{L}_2$	0.1273	FGM	50000	10	$\mathcal{L}_2$	0.1321
PGD	50000	15	$\mathcal{L}_2$	0.1655	FGM	50000	15	$\mathcal{L}_2$	0.1673

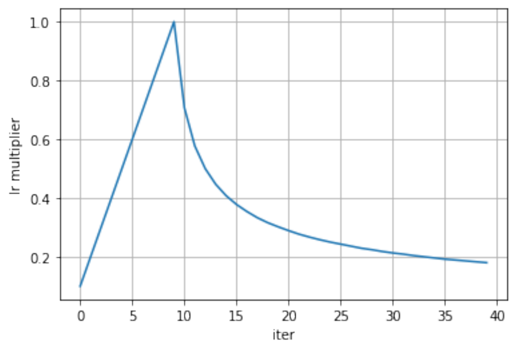

Figure B.1: Learning rate multiplier used in adversarial fine-tuning.

C Proofs

C.1 Theorem 1

To prove Theorem 1, we first figure out $\|Std(\widehat{\theta}(0))\|_F$ in the following lemma, then present the main proof.

Lemma 1. *Under the data generation model in Theorem 1, when $n-d-1>0$,*

$$Var(\widehat{\theta}(0)) = \frac{\sigma^2 I_d}{n-d-1},$$

and when $d-n-1>0$,

$$tr(Var(\widehat{\theta}(0))) = \frac{\sigma^2 n}{d-n-1}.$$

Consequently,

$$\|Std(\widehat{\theta}(0))\|_F = \begin{cases} \Theta(\sqrt{d/n}) & d \ll n \\ \Theta(1) & d \asymp n \\ \Theta(\sqrt{n/d}) & d \gg n \end{cases}.$$

Proof of Lemma 1. From the definition of Std and Var, we know that

$$\|Std(\widehat{\theta}(0))\|_F = \sqrt{tr(Std(\widehat{\theta}(0))Std(\widehat{\theta}(0)))} = \sqrt{tr(Var(\widehat{\theta}(0)))}.$$

Since we use linear regression with Gaussian design, the closed-form solution of $\widehat{\theta}(0)$ exists and its variance can be directly computed.

Denote X_n as the data matrix and Y_n as the corresponding response vector. Also define ε_n as the noise vector.

Case 1, $n-d-1>0$***:*** we know that

$$\widehat{\theta}(0) = (X_n^\top X_n)^{-1}X_n^\top Y_n = \theta_0 + (X_n^\top X_n)^{-1}X_n^\top \varepsilon_n.$$

As a result, the variance of $\widehat{\theta}(0)$ becomes

$$\begin{aligned} & Var(\widehat{\theta}(0)) \\ = \;& Var((X_n^\top X_n)^{-1}X_n\varepsilon_n) \\ = \;& \mathbb{E}(X_n^\top X_n)^{-1}X_n^\top \varepsilon_n\varepsilon_n^\top X_n(X_n^\top X_n)^{-1} - \mathbb{E}\left[(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n\right]\mathbb{E}\left[\varepsilon_n^\top X_n(X_n^\top X_n)^{-1}\right] \\ = \;& \mathbb{E}\left[(X_n^\top X_n)^{-1}X_n^\top \mathbb{E}[\varepsilon_n\varepsilon_n^\top|X_n]X_n(X_n^\top X_n)^{-1}\right] - 0 \\ = \;& \sigma^2\mathbb{E}(X_n^\top X_n)^{-1}. \end{aligned}$$

Since $x\sim N(0,I_d)$, $(X_n^\top X_n)^{-1}$ follows inverse Wishart distribution associated with (I_d,n), and we obtain

$$Var(\widehat{\theta}(0)) = \frac{\sigma^2 I_d}{n-d-1}.$$

Case 2, $d-n-1>0$***:*** when $n<d-n-1$, following Belkin et al. (2019b), we know that

$$\widehat{\theta}(0) = X_n^\top(X_nX_n^\top)^{-1}Y_n = X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0 + X_n^\top(X_nX_n^\top)^{-1}\varepsilon_n.$$

As a result, the variance matrix of $\widehat{\theta}(0)$ becomes

$$\begin{aligned} & Var(\widehat{\theta}(0)) \\ = \;& \mathbb{E}\widehat{\theta}(0)\widehat{\theta}(0)^\top - \mathbb{E}\widehat{\theta}(0)\mathbb{E}\widehat{\theta}(0)^\top \\ = \;& \mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\theta_0X_n^\top(X_nX_n^\top)^{-1}X_n + \mathbb{E}X_n^\top(X_nX_n^\top)^{-1}\varepsilon_n\varepsilon_n^\top(X_nX_n^\top)^{-1}X_n \\ & + \underbrace{\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\varepsilon_n^\top(X_nX_n^\top)^{-1}X_n}_{=0} \\ & + \underbrace{\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}\varepsilon_n\theta_0X_n^\top(X_nX_n^\top)^{-1}X_n}_{=0} \\ & - \left[\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\right]\theta_0\theta_0^\top\left[\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\right] \\ = \;& \mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\theta_0X_n^\top(X_nX_n^\top)^{-1}X_n + \sigma^2\mathbb{E}X_n^\top(X_nX_n^\top)^{-2}X_n \\ & - \left[\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\right]\theta_0\theta_0^\top\left[\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\right]. \end{aligned}$$

Since each dimension of x is independent $N(0,1)$, we have

$$tr(\mathbb{E}X_n^\top(X_nX_n^\top)^{-2}X_n) = \mathbb{E}tr((X_nX_n^\top)^{-1}) = \frac{n}{d-n-1}.$$

Using the symmetric property of $X_nX_n^\top$, we have

$$\|\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\|^2 = \frac{1}{d}tr(\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n) = 1,$$

and all the eigenvalues of $\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n$ are the same, so

$$tr\left(\left[\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\right]\theta_0\theta_0^\top\left[\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\right]\right) = \|\theta_0\|^2.$$

For the trace of $\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\theta_0^\top X_n^\top(X_nX_n^\top)^{-1}X_n$, we have

$$\begin{aligned}
& tr\left(\mathbb{E}X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\theta_0X_n^\top(X_nX_n^\top)^{-1}X_n\right)\\
= \;& \mathbb{E}\theta_0^\top X_n^\top(X_nX_n^\top)^{-1}X_nX_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\\
= \;& \mathbb{E}\theta_0^\top X_n^\top(X_nX_n^\top)^{-1}X_n\theta_0\\
= \;& \|\theta_0\|^2.
\end{aligned}$$

As a result,

$$tr(Var(\widehat{\theta}(0))) = \frac{\sigma^2 n}{d-n-1}.$$

□

Proof of Theorem 1. We first show that the training trajectory for clean training dominates adversarial training when $\epsilon = o(\sqrt{d/n})$.

Instead of considering the two cases where $\liminf d/n > 0$ or not, we consider three different scenarios w.r.t. d: (1) high-dimension regime ($d \gg n$), (2) large-sample regime ($d \ll n$), and (3) moderate dimension regime ($d \asymp n$). These three scenarios are more general in statistical literature.

Denote $X_n \in \mathbb{R}^{n\times d}$ as the data matrix and $Y_n \in \mathbb{R}^n$ as the response vector. Then the updating gradient of square loss in clean training is

$$\frac{\partial\widehat{R}(\theta,0)}{\partial\theta} = \frac{2}{n}(X_n^\top X_n\theta - X_n^\top Y_n),$$

while the gradient for adversarial training is

$$\frac{\partial\widehat{R}(\theta,\epsilon)}{\partial\theta} = \underbrace{\frac{2}{n}(X_n^\top X_n\theta - X_n^\top Y_n)}_{:=A} + \underbrace{2\epsilon^2\theta}_{:=B} + \underbrace{\frac{2\epsilon}{n}\frac{\theta}{\|\theta\|}\|X_n\theta - Y_n\|_1}_{:=C} + \underbrace{\frac{2\epsilon\|\theta\|}{n}X_n^\top\operatorname{sgn}(X_n\theta - Y_n)}_{:=D}.$$

- In high-dimension regime, taking η small enough so that it is smaller than $1/\lambda_{\max}(X_n^\top X_n)$ ($\lambda_{\max}$ refers to the largest eigenvalue), from the proof of Theorem 5 of Xing et al. (2021b), $\epsilon = o(\sqrt{d/n})$ is a sufficient condition to ensure that $\theta^{(t)}(\epsilon)$ is dominated by $\theta^{(0)}(0)$ for all $t \le T$ with some properly chosen T.

- In the large sample regime, one can show that

$$\|\theta^{(t+1)}(\epsilon) - \theta^{(t+1)}(0)\| \le \|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\| + \eta\left\|\frac{\partial\widehat{R}(\theta^{(t)}(0),0)}{\partial\theta} - \frac{\partial\widehat{R}(\theta^{(t)}(\epsilon),\epsilon)}{\partial\theta}\right\|,$$

where

$$\begin{aligned}
& \left\|\frac{\partial\widehat{R}(\theta^{(t)}(0),0)}{\partial\theta} - \frac{\partial\widehat{R}(\theta^{(t)}(\epsilon),\epsilon)}{\partial\theta}\right\|\\
= \;& \left\|\frac{2}{n}X_n^\top X_n(\theta^{(t)}(0) - \theta^{(t)}(\epsilon))\right\| + \|B\| + \|C\| + \|D\|.
\end{aligned}$$

With probability tending to 1, we have for some positive constant C_1,

$$\lambda_{\min}\left(\frac{1}{n}X_n^\top X_n - I_d\right) \geq C_1\sqrt{\frac{d}{n}}\frac{1}{\log n},\ \lambda_{\max}\left(\frac{1}{n}X_n^\top X_n - I_d\right) \leq C_1\sqrt{\frac{d}{n}}\log n,$$

therefore,

$$\left\|\frac{2}{n}X_n^\top X_n(\theta^{(t)}(0) - \theta^{(t)}(\epsilon))\right\| = \Theta(\|\theta^{(t)}(0) - \theta^{(t)}(\epsilon)\|).$$

On the other hand,

$$\begin{aligned}
\|A\| &= \left\|\frac{2}{n}(X_n^\top X_n\theta - X_n^\top Y_n)\right\| = \frac{2}{n}\sqrt{(X_n\theta - Y_n)^\top(X_n^\top X_n)(X_n\theta - Y_n)},\\
\|B\| &= 2\epsilon^2\|\theta\|,\\
\|C\| &\leq \frac{2\epsilon}{n}\sqrt{n}\|X_n\theta - Y_n\|,\\
\|D\| &\leq 2\epsilon\|\theta\|\frac{1}{n}\|X_n\|\|\operatorname{sgn}(X_n\theta - Y_n)\|_2 \leq 2\epsilon\|\theta\|\|X_n\|/\sqrt{n}.
\end{aligned}$$

With probability tending to 1, we have $\|A\| \leq C_3\|X_n\theta - Y_n\|/\sqrt{n}$, $\|A\| \geq C_4\|X_n\theta - Y_n\|/\sqrt{n}$, and $\|X_n\| \leq C_5\sqrt{n}$ for some positive constants C_3, C_4, C_5. As a result, when $\|\theta-\theta_0\| = \Omega(\sqrt{d/n})$ and $\|X_n\theta-Y_n\|_2^2/n = \Omega(d/n)$, the adversarial training is dominated by A.

Rewriting $\theta^{(t)}(\epsilon) = \theta^{(t)}(0) + \theta^{(t)}(\epsilon) - \theta^{(t)}(0)$, we have $\|\theta^{(t)}(\epsilon)\| \leq \|\theta^{(t)}(0)\| + \|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\|$. Similarly we have $\|X_n\theta^{(t)}(\epsilon) - Y_n\| \leq \|X_n\theta^{(t)}(0) - Y_n\| + \|X_n(\theta^{(t)}(\epsilon) - \theta^{(t)}(0))\|$, and for some positive constants C_6 to C_{10}

$$\begin{aligned}
\|\theta^{(t+1)}(\epsilon) - \theta^{(t+1)}(0)\| &\leq (1 + C_6\eta)\|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\| + \eta\|B\| + \eta\|C\| + \eta\|D\|\\
&\leq (1 + C_6\eta + C_7\epsilon\eta)\|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\| + C_8\epsilon\eta\|\theta^{(t)}(0)\|\\
&\quad + C_9\epsilon\eta\|X_n\theta^{(t)}(0) - Y_n\|/\sqrt{n}.
\end{aligned}$$

Based on Ba et al. (2020), we have with probability tending to 1, $\|\theta^{(t)}(0)\|$ and $\|X_n\theta^{(t)}(0) - Y_n\|/\sqrt{n}$ are both bounded, thus we have

$$\|\theta^{(t+1)}(\epsilon) - \theta^{(t+1)}(0)\| \leq (1 + C_{10}\eta)\|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\| + C_{11}\epsilon\eta.$$

Taking (η, T) such that $\eta \to 0$, $T = O(\text{poly}(d, n))$, $\eta T \to \infty$, $\eta T/(\log n) \to 0$, we have for all $t \leq T$,

$$\|\theta^{(t)}(\epsilon) - \theta^{(t)}(0)\| = (1 + \eta t)\epsilon = o(\sqrt{d/n}).$$

In addition, based on Xing et al. (2021b), we also have $\theta^{(T)}(\epsilon) \to \widehat{\theta}(\epsilon)$ in this case.

- For moderate dimension case, the proof is similar.

When $\epsilon \gg \sqrt{d/n}$,

- When $d/n < \infty$, the optimum solutions for $\widehat{R}(\theta, \epsilon)$ and $\widehat{R}(\theta, 0)$ are already different, so their training trajectory are different as well.
- When $d/n \to \infty$, we have both $\|\widehat{\theta}(\epsilon)\| \to 0$ and $\|\widehat{\theta}(0)\| \to 0$. However, in clean training, there is a stage that $\|\theta^{(t)}(0)\| = O(\sqrt{n/d})$ while $\|X_n\theta^{(t)}(0) - Y_n\|^2/n \to 0$, so B will finally dominates $\partial\widehat{R}(\theta, \epsilon)/\partial\theta$.

□

C.2 Theorem 2

Lemma 2. *Under the conditions in Theorem 1,*

- *When $d/n < \infty$,*
$$\Delta = O_p(\sqrt{d/n}).$$
- *When $d/n \to \infty$,*
$$\Delta = O_p(1).$$

Proof of Lemma 2. Since $\epsilon = 0$, we can obtain the closed-form solution of $\widehat{\theta}(0)$.

Case 1, $n - d - 1 > 0$, we know that

$$\widehat{\theta}(0) = \theta_0 + (X_n^\top X_n)^{-1} X_n^\top \varepsilon_n,$$

therefore,

$$\begin{aligned} \Delta &= \mathbb{E}_{(X,\varepsilon)}(X(X_n^\top X_n)^{-1}X_n^\top \varepsilon_n - \varepsilon)^2 - \|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n \\ &= \varepsilon_n^\top X_n (X_n^\top X_n)^{-2} X_n^\top \varepsilon_n + \sigma^2 - \|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n, \end{aligned}$$

where

$$\mathbb{E}\|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n = \frac{\sigma^2}{n}\mathbb{E}tr\left(I_n - X_n(X_n^\top X_n)^{-1}X_n\right) = \frac{\sigma^2(n-d)}{n},$$

and

$$\begin{aligned} & \mathbb{E}\|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^4/n^2 \\ =\ & 2\sigma^4\mathbb{E}tr(I_n - X_n(X_n^\top X_n)^{-1}X_n) + \mathbb{E}^2\|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n \\ =\ & 2\sigma^4\frac{n-d}{n} + \sigma^4\frac{(n-d)^2}{n^2}. \end{aligned}$$

As a result, the mean and variance of Δ become

$$\begin{aligned} \mathbb{E}\Delta &= \sigma^2\left(tr(\mathbb{E}(X_n^\top X_n)^{-1}) + 1\right) - \mathbb{E}\|(X_n^\top X_n)^{-1}X_n\varepsilon_n - \varepsilon_n\|^2/n \\ &= \sigma^2\frac{d}{n-d-1} + \sigma^2 - \sigma^2\frac{n-d}{n} \\ &= \sigma^2 d\left(\frac{1}{n-d-1} - \frac{1}{n}\right), \end{aligned}$$

and

$$\begin{aligned} \mathbb{E}\Delta^2 &= \mathbb{E}\left(\varepsilon_n^\top X_n(X_n^\top X_n)^{-2}X_n^\top\varepsilon_n + \sigma^2 - \|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n\right)^2 \\ &= \mathbb{E}\left(\varepsilon_n^\top X_n(X_n^\top X_n)^{-2}X_n^\top\varepsilon_n + \sigma^2\right)^2 + \mathbb{E}\|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^4/n^2 \\ &\quad -2\mathbb{E}\left(\varepsilon_n^\top X_n(X_n^\top X_n)^{-2}X_n^\top\varepsilon_n + \sigma^2\right)\|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n, \end{aligned}$$

where

$$\begin{aligned} & \mathbb{E}\left(\varepsilon_n^\top X_n(X_n^\top X_n)^{-2}X_n^\top\varepsilon_n + \sigma^2\right)^2 \\ =\ & 2\sigma^4\mathbb{E}tr\left(X_n(X_n^\top X_n)^{-3}X_n^\top\right) + \sigma^4\frac{d^2}{(n-d-1)^2} + \sigma^4 + 2\sigma^4\frac{d}{n-d-1}, \end{aligned}$$

$$\begin{aligned} & \mathbb{E}\left(\varepsilon_n^\top X_n(X_n^\top X_n)^{-2}X_n^\top\varepsilon_n + \sigma^2\right)\|X_n(X_n^\top X_n)^{-1}X_n^\top\varepsilon_n - \varepsilon_n\|^2/n \\ =\ & 2\sigma^4\frac{1}{n}\mathbb{E}tr\left(X_n(X_n^\top X_n)^{-2}X_n^\top\right) + \sigma^4\frac{d}{n-d-1}\frac{n-d}{n} + \sigma^4\frac{n-d}{n} \\ =\ & 2\sigma^4\frac{d}{n-d-1} + \sigma^4\frac{d}{n-d-1}\frac{n-d}{n} + \sigma^4\frac{n-d}{n}. \end{aligned}$$

One can conclude that $\mathbb{E}\Delta^2 = \Theta(d/n)$ and thus $\Delta = O_p(\sqrt{d/n})$.

Case 2, $n/d \to 0$, we know that $\widehat{R}(\widehat{\theta}(0), 0) = 0$ due to data interpolation, and $\mathbb{E}\|\widehat{\theta}(0)\|^2 = \Theta(n/d)$, thus one can directly obtain that $\Delta = O_p(1)$.

Case 3, $d - n - 1 > 0$ *while* $n/d > 0$, the analysis is similar to Case 1. □

Proof of Theorem 2. When $d/n < \infty$, we have with probability tending to 1,

$$\begin{aligned} & \widehat{R}(\widehat{\theta}(0),0)+O(\epsilon+\epsilon^2) \\ \leq\ & \widehat{R}(\widehat{\theta}(\epsilon),0)+O(\epsilon+\epsilon^2) \\ \leq\ & \widehat{R}(\widehat{\theta}(\epsilon),\epsilon) \\ \leq\ & \widehat{R}(\widehat{\theta}(0),\epsilon) \\ =\ & \widehat{R}(\widehat{\theta}(0),0)+O(\epsilon+\epsilon^2), \end{aligned}$$

which means that $\epsilon = \Theta(\sqrt{d/n})$ is the threshold of $(\widehat{R}(\widehat{\theta}(\epsilon),\epsilon)-\widehat{R}(\widehat{\theta}(0),0))/\Delta$.

When $d/n \to \infty$, if $\epsilon = o(\sqrt{d/n})$, we have

$$\widehat{R}(\widehat{\theta}(0),\epsilon) = O(\epsilon^2\|\widehat{\theta}(0)\|^2) = o(1).$$

Thus

$$\widehat{R}(\widehat{\theta}(\epsilon),\epsilon) \leq \widehat{R}(\widehat{\theta}(0),\epsilon) = o(1),$$

and

$$(\widehat{R}(\widehat{\theta}(\epsilon),\epsilon)-\widehat{R}(\widehat{\theta}(0),0))/\Delta \to 0.$$

When $d/n \to \infty$ and $\epsilon \gg \sqrt{d/n}$, $\widehat{R}(\theta,\epsilon)$ is dominated by $\epsilon^2\|\theta\|^2$, so $\|\widehat{\theta}(\epsilon)\|$ is small enough so that $\epsilon^2\|\widehat{\theta}(\epsilon)\|^2 = O(1)$. When $\|\theta\|$ is small enough, we have $\widehat{R}(\theta,0)=\Theta(1)$ in probability. These observations imply that $\widehat{R}(\widehat{\theta}(\epsilon),\epsilon)=\Theta(1)$ in probability. Consequently, $(\widehat{R}(\widehat{\theta}(\epsilon),\epsilon)-\widehat{R}(\widehat{\theta}(0),0))/\Delta$ does not converge to zero.

□

C.3 Proposition 1

The proof idea is the same as Ba et al. (2020) and Xing et al. (2021b). When $h \to 0$, for smooth activation functions, one can use first-order Taylor expansion to approximate them as linear functions. As a result, with vanishing initialization, the two-layer neural network is approximately a linear network (a linear function), thus the results follows Theorem 1 and 2.

In particular, we assume

- The activation function ϕ in model (3) is twice continuously differentiable, $\phi'(0) \neq 0$, and $\phi(0)=0$.
- The initialization satisfies $\theta_{\xi,j}^{(0)} \sim N(0, I_d/dh^{1+\delta})$ for some $\delta > 0$.
- The parameter a satisfy $\|a\|_\infty = O(1)$, $\max|a_j|/(\min|a_j|)=\Theta(1)$.

Then when h is sufficiently large and δ is large enough, taking $\eta = \eta_{linear} h/(\|a\|^2\phi'(0)^2)$ and suitable $T \to \infty$, we have with probability tending to 1 over the generation of x,

$$\left| \frac{1}{\sqrt{h}} \sum_{j=1}^{h} \phi(x^\top \theta_j^{(T)}) a_j - x^\top \theta_{linear}^{(T)} \right| = o(\sqrt{d/n} \wedge \sqrt{n/d}),$$

where η_{linear} and $\theta_{linear}^{(T)}$ are the learning rate and model parameters corresponding to the linear model.

C.4 Proposition 2

Proof of Proposition 2. Denote $g_t = \partial \widehat{R}(\theta^{(t)},\epsilon)/\partial\theta^{(t)}$. The updating rule leads to

$$\|\theta^{(t)}-\widehat{\theta}\|^2 \leq \|\theta^{(t-1)}-\widehat{\theta}-\eta_t g_t\|^2 \leq \|\theta^{(t-1)}-\widehat{\theta}\|^2 - 2\eta_t g_t^\top(\theta^{(t-1)}-\widehat{\theta}) + \eta_t^2 L^2.$$

Taking expectation and move some terms, it becomes

$$g_t^\top(\theta^{(t-1)} - \widehat{\theta}) \le \frac{1}{2\eta_t}\|\theta^{(t-1)} - \widehat{\theta}\|^2 - \frac{1}{2\eta_t}\|\theta^{(t)} - \widehat{\theta}\|^2 + \frac{1}{2}\eta_t L_r^2.$$

Taking average over $t = 1$ to T, we have

$$\begin{aligned}
\frac{1}{T}\left[\sum_{t=1}^{T} g_t^\top(\theta^{(t-1)} - \widehat{\theta})\right] &\le \frac{1}{2T}\left[\sum_{t=1}^{T}\frac{1}{\eta_t}\|\theta^{(t-1)} - \widehat{\theta}\|^2 - \frac{1}{\eta_t}\|\theta^{(t)} - \widehat{\theta}\|^2\right] + \frac{L^2}{2T}\sum_{t=1}^{T}\eta_t \\
&= \frac{\|\theta^{(0)} - \widehat{\theta}\|^2}{2\eta_1 T} - \frac{\|\theta^{(T)} - \widehat{\theta}\|^2}{2\eta_T T} \\
&\quad + \frac{1}{2T}\left[\sum_{t=1}^{T-1}\left(\frac{1}{\theta_{t+1}} - \frac{1}{\theta_t}\right)\|\theta^{(t)} - \widehat{\theta}\|^2\right] + \frac{L^2}{2T}\sum_{t=1}^{T}\eta_t.
\end{aligned}$$

Finally, since $\widehat{R}$ is a convex function, we have

$$\frac{1}{T}\left[\sum_{t=1}^{T} g_t^\top(\theta^{(t-1)} - \widehat{\theta})\right] \ge \frac{1}{T}\sum_{t=1}^{T}\widehat{R}(\theta^{(t)}) - \widehat{R}(\widehat{\theta}) \ge \left[\min_{t=1,\ldots,T}\widehat{R}(\theta^{(t)}) - \widehat{R}(\widehat{\theta})\right].$$

Further, to show that $\widehat{\epsilon} - \epsilon^* = o(\epsilon^*)$, it suffices to show that $\widehat{R}(\widehat{\theta}(\epsilon), \epsilon)$ is differentiable and has finite second order derivative when $\epsilon = o(1)$.

Recall the first order optimality condition for $\widehat{R}(\theta, \epsilon)$ is

$$\frac{\partial \widehat{R}(\widehat{\theta}(\epsilon), \epsilon)}{\partial \theta} = \mathbf{0}.$$

Therefore, take $\Delta\epsilon \to 0$,

$$\begin{aligned}
\mathbf{0} &= \frac{\partial \widehat{R}(\widehat{\theta}(\epsilon), \epsilon)}{\partial \theta} - \frac{\partial \widehat{R}(\widehat{\theta}(\epsilon + \Delta\epsilon), \epsilon + \Delta\epsilon)}{\partial \theta} \\
&= \frac{2}{n}(X_n^\top X_n \widehat{\theta}(\epsilon) - X_n^\top Y_n) + 2\epsilon^2\widehat{\theta}(\epsilon) + \frac{2\epsilon}{n}\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon) - Y_n\|_1 \\
&\quad + \frac{2\epsilon\|\widehat{\theta}(\epsilon)\|}{n}X_n^\top \operatorname{sgn}(X_n\widehat{\theta}(\epsilon) - Y_n) \\
&\quad - \frac{2}{n}(X_n^\top X_n\widehat{\theta}(\epsilon + \Delta\epsilon) - X_n^\top Y_n) - 2(\epsilon + \Delta\epsilon)^2\widehat{\theta}(\epsilon + \Delta\epsilon) \\
&\quad - \frac{2(\epsilon + \Delta\epsilon)}{n}\frac{\widehat{\theta}(\epsilon + \Delta\epsilon)}{\|\widehat{\theta}(\epsilon + \Delta\epsilon)\|}\|X_n\widehat{\theta}(\epsilon + \Delta\epsilon) - Y_n\|_1 \\
&\quad - \frac{2(\epsilon + \Delta\epsilon)\|\widehat{\theta}(\epsilon + \Delta\epsilon)\|}{n}X_n^\top \operatorname{sgn}(X_n\widehat{\theta}(\epsilon + \Delta\epsilon) - Y_n) \\
&= \frac{2}{n}X_n^\top X_n(\widehat{\theta}(\epsilon) - \widehat{\theta}(\epsilon + \Delta\epsilon)) + 2\epsilon^2(\widehat{\theta}(\epsilon) - \widehat{\theta}(\epsilon + \Delta\epsilon)) + 2\epsilon\Delta\epsilon\widehat{\theta}(\epsilon) \\
&\quad + \frac{2\Delta\epsilon}{n}\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon) - Y_n\|_1 \\
&\quad + \frac{2\epsilon}{n}\left[\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon) - Y_n\|_1 - \frac{\widehat{\theta}(\epsilon + \Delta\epsilon)}{\|\widehat{\theta}(\epsilon + \Delta\epsilon)\|}\|X_n\widehat{\theta}(\epsilon + \Delta\epsilon) - Y_n\|_1\right] \\
&\quad + \frac{2\Delta\epsilon}{n}\|\widehat{\theta}(\epsilon)\|X_n^\top \operatorname{sgn}(X_n\widehat{\theta}(\epsilon) - Y_n) \\
&\quad + \frac{2\epsilon}{n}\left[\|\widehat{\theta}(\epsilon)\|X_n^\top \operatorname{sgn}(X_n\widehat{\theta}(\epsilon) - Y_n) - \|\widehat{\theta}(\epsilon + \Delta\epsilon)\|X_n^\top \operatorname{sgn}(X_n\widehat{\theta}(\epsilon + \Delta\epsilon) - Y_n)\right] + o.
\end{aligned}$$

As a result, putting $(\widehat{\theta}(\epsilon)-\widehat{\theta}(\epsilon+\Delta\epsilon))$ on one side and $\Delta\epsilon$ on the other side of the above equation, we have

$$\begin{aligned}
&\left[\frac{1}{n}X_n^\top X_n+\epsilon^2+\frac{\epsilon}{n}\frac{\partial}{\partial\theta}\left(\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon)-Y_n\|_1\right)+\frac{\epsilon}{n}\frac{\partial\|\widehat{\theta}(\epsilon)\|X_n^\top\operatorname{sgn}(X_n\widehat{\theta}(\epsilon)-Y_n)}{\partial\theta}\right]\\
&\times(\widehat{\theta}(\epsilon)-\widehat{\theta}(\epsilon+\Delta\epsilon))\\
=\;&\Delta\epsilon\left[\epsilon\widehat{\theta}(\epsilon)+\frac{1}{n}\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon)-Y_n\|_1+\frac{1}{n}\|\widehat{\theta}(\epsilon)\|X_n^\top\operatorname{sgn}(X_n\widehat{\theta}(\epsilon)-Y_n)+o\right],
\end{aligned}$$

which implies that

$$\begin{aligned}
&\frac{\widehat{\theta}(\epsilon)-\widehat{\theta}(\epsilon+\Delta\epsilon)}{\Delta\epsilon}\\
=\;&\left[\frac{1}{n}X_n^\top X_n+\epsilon^2+\frac{\epsilon}{n}\frac{\partial}{\partial\theta}\left(\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon)-Y_n\|_1\right)\right.\\
&\qquad\qquad\left.+\frac{\epsilon}{n}\frac{\partial\|\widehat{\theta}(\epsilon)\|X_n^\top\operatorname{sgn}(X_n\widehat{\theta}(\epsilon)-Y_n)}{\partial\theta}\right]^{-1}\\
&\qquad\times\left[\epsilon\widehat{\theta}(\epsilon)+\frac{1}{n}\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon)-Y_n\|_1+\frac{1}{n}\|\widehat{\theta}(\epsilon)\|X_n^\top\operatorname{sgn}(X_n\widehat{\theta}(\epsilon)-Y_n)+o\right].
\end{aligned}$$

Note that due to the distribution of X_n and Y_n, $\|X_n\widehat{\theta}(\epsilon)-Y_n\|_1/n$ and $\operatorname{sgn}(X_n\widehat{\theta}(\epsilon)-Y_n)/n$ are approximately smooth. In addition, since both ϵ^* and ϵ are $o(1)$, this corresponds to the large-sample and moderate-dimension regimes, thus $\|\widehat{\theta}(\epsilon)\|$ diverges from zero, and $\frac{\partial}{\partial\theta}\left(\frac{\widehat{\theta}(\epsilon)}{\|\widehat{\theta}(\epsilon)\|}\|X_n\widehat{\theta}(\epsilon)-Y_n\|_1\right)$ exists as well.

Using similar technique, one can obtain the second-order derivative of $\widehat{R}(\widehat{\theta}(\epsilon),\epsilon)$ w.r.t. ϵ. $\square$