Quantifying the Benefit of Using Differentiable Learning over Tangent Kernels

Eran Malach [1] **Pritish Kamath** [2] **Emmanuel Abbe** [3] **Nathan Srebro** [2]

Abstract

We study the relative power of learning with gradient descent on differentiable models, such as neural networks, versus using the corresponding tangent kernels. We show that under certain conditions, gradient descent achieves small error only if a related tangent kernel method achieves a non-trivial advantage over random guessing (a.k.a. weak learning), though this advantage might be very small even when gradient descent can achieve arbitrarily high accuracy. Complementing this, we show that without these conditions, gradient descent can in fact learn with small error even when no kernel method, in particular using the tangent kernel, can achieve a non-trivial advantage over random guessing.[1]

1. Introduction

A recent line of research seeks to understand Neural Networks through their kernel approximation, as given by the Neural Tangent Kernel (NTK, Jacot et al., 2018). The premise of the approach is that in certain regimes, the dynamics of training neural networks are essentially the same as those of its first order Taylor expansion at initialization, which in turn is captured by the Tangent Kernel at initialization. It is then possible to obtain convergence, global optimality and even generalization guarantees by studying the behaviour of training using the Tangent Kernel (e.g. Li & Liang, 2018; Du et al., 2019b; Chizat et al., 2019; Zou et al., 2020; Allen-Zhu et al., 2019; Arora et al., 2019b; Du et al., 2019a, and many others). Some have also suggested using Tangent Kernel directly for training (e.g. Arora et al., 2019a), even suggesting it can sometimes outperform training by GD on the actual network (Geiger et al., 2020).

[*]Equal contribution [1]Hebrew University of Jerusalem, Israel [2]Toyota Technological Institute at Chicago, USA [3]EPFL, Switzerland. Correspondence to: Eran Malach <eran.malach@mail.huji.ac.il>, Pritish Kamath <pritish@ttic.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1]The full (better formatted and potentially more up-to-date) version of this paper is available at arxiv.org/abs/2103.01210.

Can all the success of deep learning be explained using the NTK? This would imply that we can replace training by gradient descent on a non-convex model with a (potentially simpler to train, and certainly better understood) kernel method. Is anything learnable using gradient descent on a neural network or other differentiable model also learnable using a kernel method (i.e. using a kernelized linear model)?

This question was directly addressed by multiple authors, who showed examples where neural networks trained with gradient descent (or some variant thereof) provably outperform the best that can possibly be done using the tangent kernel or any linear or kernel method, under different settings and assumptions (Yehudai & Shamir, 2019; Allen-Zhu & Li, 2019; 2020; Li et al., 2020; Daniely & Malach, 2020; Ghorbani et al., 2019b; 2020). However in these examples, while training the model with gradient descent performs better than using the NTK, the error of the NTK is still much better then baseline, with a significant "edge" over random guessing or using a constant, or null, predictor. That is, the NTK at least allows for "weak learning". In fact, in some of the constructions, the process of leveraging the "edge" of a linear or kernel learner and amplifying it is fairly explicit (e.g. Allen-Zhu & Li, 2019; 2020). The question we ask in this paper is:

> *Can gradient descent training on the actual deep (non-convex) model only amplify (or "boost") the edge of the NTK, with the NTK **required** to have an edge in order to allow for learning in the first place? Or can differentiable learning succeed even when the NTK —or any other kernel— does not have a non-trivial edge?*

Can gradient descent succeed at "strong learning" only when the NTK achieves "weak learning"? Or is it possible for gradient descent to succeed even when the NTK is not able to achieve any significant edge?

Our Contributions. The answer turns out to be subtle, and as we shall see, relies crucially on two important considerations: the *unbiasedness* of the initialization, and whether the initialization can depend on the *input distribution*.

We start, in Section 3, by providing our own example of a learning problem where Gradient Descent outperforms the NTK, and indeed any kernel method. But unlike the previous recent separating examples, where the NTK enjoys

a considerable edge (constant edge, or even near zero error, but with a slower rate than GD), we show that the edge of the NTK, and indeed of any (poly-sized) kernel, can be arbitrarily close to zero while the edge of GD can be arbitrarily large. The edge of the NTK in this example is nonetheless vanishing at low rate, i.e., polynomially, and this leads us to ask whether the NTK must have at least a polynomial edge when GD succeeds.

In Theorem 4 of Section 4.1 we show that when using an *unbiased* initialization, that is, where the output of the model is 0 at initialization, then indeed the NTK must have a non-trivial edge (polynomial in the accuracy and scale of the model) in order for gradient descent to succeed.

The requirement that the initialization is unbiased turns out to be essential: In Section 5 we show an example where gradient descent on a model succeeds, from a biased initialization, even though *no* (reasonably sized) kernel method (let alone the NTK) can ensure a non-trivial edge.

But all is not lost with biased initialization. In Theorem 5 of Section 4.2 we show that at least for the square loss, if gradient descent succeeds from any (possibly biased) initialization, then we can construct some alternate random "initialization" such that the Tangent Kernel at this random initialization (i.e. in expectation over the randomness) *does* ensure a non-trivial edge. Importantly, this random initialization must depend on knowledge of the input distribution. Again, our example in Section 5 shows that this distribution-dependence is essential. This also implies a separation between problems learnable by gradient descent using an unbiased vs arbitrary initialization, emphasizing that the initialization being unbiased should not be taken for granted.

This subtle answer is mapped in Table 1 (Appendix A).

2. Differentiable Learning and Tangent Kernels

We consider learning a predictor $f : \mathcal{X} \to \mathbb{R}$ over an *input space* $\mathcal{X}$, so as to minimize its *population loss* $\mathcal{L}_{\mathcal{D}}(f) := \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(f(x), y)\right]$ with respect to a *source distribution* $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a loss function. We denote by ℓ' and ℓ'' the derivatives of ℓ w.r.t. its first argument. Some of our guarantees apply to any smooth loss, i.e., $|\ell''| := \sup_{\widehat{y},y} |\ell''(\widehat{y}, y)| < \infty$, while others are specific to the square loss $\ell_{\mathrm{sq}}(\widehat{y}, y) = \frac{1}{2}(\widehat{y} - y)^2$ with binary labels $\mathcal{Y} = \{-1, 1\}$. Our lower bounds and separation results are all stated for the square loss with binary labels (they could be adapted to other loss functions, but in order to demonstrate a separation, it is sufficient to provide a specific example). To understand whether a predictor gives any benefit, we discuss the error of a predictor compared to that of "null prediction", i.e. predicting 0 on all inputs. For the square loss with binary labels, the error of null prediction is $\mathcal{L}_{\mathcal{D}}(0) = 0.5$, and so if $\mathcal{L}_{\mathcal{D}}(f) = 0.5 - \gamma$, we say that f has an "edge" of γ (over null prediction).

Differentiable Learning. We study learning by (approximate) gradient descent on a differentiable model, such as a neural network or other parametric model. More formally, a *differentiable model* of size p is a mapping $f : \mathbb{R}^p \times \mathcal{X} \to \mathbb{R}$, denoted $f_\theta(x)$, where $\theta \in \mathbb{R}^p$ are the model parameters, or "weights", of the model, and the mapping is differentiable w.r.t. θ. We will control the scale of the model through a bound[2] $C_f^2 := \sup_{\theta,x}(\|\nabla_\theta f_\theta(x)\|_2^2 + f_\theta^2(x))$ on the norm of the gradients and function values[3]

For a differentiable model $f_\theta(x)$ and an *initialization* $\theta_0 \in \mathbb{R}^p$, gradient accuracy τ, and stepsize[4] η, **τ-approximate gradient descent training** is given by iterates, starting with $\theta^{(0)} \leftarrow \theta_0$:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta g_t \tag{1}$$

$$\text{for } \|g_t - \nabla_\theta \mathcal{L}_{\mathcal{D}}(f_{\theta^{(t)}})\|_2 \leq \tau \tag{2}$$

The gradient estimates g_t can be obtained by computing the gradient on an empirical sample of m training examples drawn from $\mathcal{D}$, in which case we can ensure accuracy $\tau \propto C_f/\sqrt{m}$. And so, we can think of C_f^2/τ^2 as capturing the sample size, and when we refer to the relative accuracy τ/C_f being polynomial, one might think of the sample size used to estimate the gradients being polynomial. But here we only assume the gradient estimates g_t are good approximations of the true gradients of the population loss, and do not specifically refer to sampling.[5] We say that τ-approximate gradient descent with model f_θ, initialization θ_0, gradient accuracy τ and T steps ensures error ε on a distribution $\mathcal{D}$ if with any gradient estimates satisfying (2), the gradient descent iterates (1) lead to an iterate $\theta^{(T)}$ with population loss $\mathcal{L}_{\mathcal{D}}(f_{\theta^{(T)}}) \leq \varepsilon$.

The Tangent Kernel. Consider the first order Taylor expansion of a differentiable model about some $\theta_* \in \mathbb{R}^p$:

$$f_\theta(x) \approx f_{\theta_*}(x) + (\theta - \theta_*)\nabla_\theta f_{\theta_*}(x) \tag{3}$$

$$= \langle [1, \theta - \theta_*], \phi_{\theta_*}(x) \rangle \tag{4}$$

$$\text{where } \phi_{\theta_*}(x) = [f_{\theta_*}(x), \nabla_\theta f_{\theta_*}(x)] \in \mathbb{R}^{p+1}, \tag{5}$$

which corresponds to a linear model with the feature map as in (5), and thus can also be represented using the kernel:

$$\mathsf{NTK}^f_{\theta_*}(x, x') = f_{\theta_*}(x) f_{\theta_*}(x') + \langle \nabla_\theta f_{\theta_*}(x), \nabla_\theta f_{\theta_*}(x') \rangle .$$

[2] The quantity C_f mixes the scale of the gradients and of the function values. We use a single quantity in order to minimize notation. Separating these would allow for more consistent scaling.

[3] ReLU networks do not formally fit this framework; see Remark 9 in Appendix C on how this can be incorporated.

[4] We can also allow variable or adaptive stepsize sequences—for simplicity of presentation we stick with a fixed stepsize.

[5] In some regimes, one should in fact distinguish the setting of large sample sets and approximate population gradients in view of the universality result proved for SGD in (Abbe & Sandon, 2020a;b). We focus here on the approximate setting that better reflects the noisier regime; see discussion in (Abbe & Sandon, 2020a) for parities.

In some regimes or model limits (e.g. Daniely et al., 2016; Jacot et al., 2018; Chizat et al., 2019; Lee et al., 2019), the approximation (3) about the initialization remains valid throughout training, and so gradient descent on the actual model $f_\theta(x)$ can be approximated by gradient descent on the linear model (4), i.e. a kernalized linear model specified by the tangent kernel $\mathsf{NTK}^f_{\theta_*}$ at initialization. One can then consider analyzing differentiable learning with the model f using this NTK approximation, or even replacing gradient descent on f with just using the NTK. How powerful can such an approximation be relative to the full power of differentiable learning? Can we expect that anything learnable with differentiable learning is also learnable under the NTK approximation?

To understand the power of a kernalized linear model with some kernel K, we should consider predictors realizable with bounded norm in the corresponding feature map (i.e. norm balls in the corresponding Reproducing Kernel Hilbert Space):

$$\mathcal{F}(K,B) := \{x \mapsto \langle w, \phi(x)\rangle : \|w\|_2 \cdot R \le B\} \quad (6)$$

$$= \{f : \mathcal{X} \to \mathbb{R} : \|f\|_K \cdot R \le B\} \quad (7)$$

$$\text{where } R := \sup_x \|\phi(x)\|_2 = \sup_x \sqrt{K(x,x)}$$

where $K(x,x') = \langle \phi(x), \phi(x')\rangle$ and $\|f\|_K$ is the K-RKHS-norm[6] of f. Predictors in $\mathcal{F}(K,B)$ can be learned to within error ε with $O(B^2/\varepsilon^2)$ samples using $O(B^2/\varepsilon^2)$ steps of gradient descent on the kernalized linear model. And since any predictor learned in this way will be in $\mathcal{F}(K,B)$, showing that there is no low-error predictor in $\mathcal{F}(K,B)$ establishes the limits of what can be learned using K. We thus study the norm ball of the Tangent Kernel, which, slightly overloading notations, we denote $\mathsf{NTK}^f_\theta(B) := \mathcal{F}(\mathsf{NTK}^f_\theta, B)$.

Learning Problems. For a fixed source distribution $\mathcal{D}$, there is always a trivial procedure for "learning" it, where a good predictor specific to $\mathcal{D}$ is hard-coded into the "procedure". E.g., we can use a kernel with a single feature corresponding to this hard coded predictor. Instead, in order to capture learning as inferring from data, and be able to state when this is *not* possible using some class of methods, e.g. using kernel methods, we refer to a "learning problem" as a family $\mathcal{P}$ of source distributions over $\mathcal{X} \times \mathcal{Y}$, and say that a learning procedure learns the problem to within error ε if for any distribution $\mathcal{D} \in \mathcal{P}$ the method learns a predictor with population error at most ε. [7]

We consider learning both when the *input distribution*, i.e. the marginal distribution $\mathcal{D}_\mathcal{X}$ over $\mathcal{X}$, is known (or fixed) and when it is unknown. **Distribution dependent learning** refers to learning when the input distribution is either fixed (i.e. all distributions $\mathcal{D} \in \mathcal{P}$ have the same marginal $\mathcal{D}_\mathcal{X}$), or when the model or kernel is allowed to be chosen based on the input distribution $\mathcal{D}_\mathcal{X}$, as might be the case when using unlabeled examples to construct a kernel. In **distribution independent learning**, we seek a single model, or kernel, that ensures small error for all source distributions in $\mathcal{P}$, even though they might have different marginals $\mathcal{D}_\mathcal{X}$.

[6] The direct definition (6) is sufficient for our purposes, and so we do not get into the definition of an RKHS (and RKHS norm). See, e.g. Schölkopf & Smola (2002), for a definition and discussion. In (7) we take the norm of f to be infinite if f is not in the RKHS.

[7] See Remark 10 in Appendix C for how this relates to a probabilistic quantifier over the training set.

3. Gradient Descent Outperforms the NTK

In this section, we exhibit a simple example of a learning problem for which (i) approximate gradient descent on a differentiable model of size p ensures arbitrarily small (in fact zero) error, whereas (ii) the tangent kernel for this model, or in fact any reasonably sized kernel, cannot ensure error better than $0.5 - \gamma$, for arbitrarily small γ, where recall 0.5 is the error of the null prediction and γ depends polynomially on the parameters of the model and of gradient descent.

Several authors have already demonstrated a range of examples where gradient descent ensures smaller error than can be ensured by any reasonably sized kernel, including the tangent kernel (Yehudai & Shamir, 2019; Ghorbani et al., 2019a;b; Allen-Zhu & Li, 2019; 2020; Li et al., 2020; Daniely & Malach, 2020). In Appendix B and Table 2 we survey these papers in detail and summarize the separations they establish. Here, we provide a concrete, self-contained and simple example for completeness, so that we can explicitly quantify the edge a kernel method might have. Our emphasis is on showing that the error for kernel methods is not just worse than gradient descent, but in fact not much better than null prediction—this is in contrast to prior separations, where the error possible using a kernel is either some constant between the error of null prediction and zero error, or more frequently, close to zero error, but not as close as the error attained by gradient descent (see Appendix B for details). In our example, we also pay attention to whether the model's predictions at initialization are zero—a property that, as we will later see, plays a crucial role in our analysis.

The Learning Problem. We consider the problem of learning k-sparse parities over n biased bits, i.e. where the marginal distribution of each bit (i.e. coordinate) is non-uniform. In order to easily obtain lower bounds for any kernel (not just the tangent kernel), we let the input distribution be a mixture of independent biased bits and a uniform distribution, so that we can argue that no kernel can do well on the uniform component, and hence bound its error also on the mixture. The key is that when k is moderate, up to $k = O(\log n)$, due to the bits being biased, a linear predictor based on all the bits in the parity has enough of an edge to be detected. This does give the tangent kernel a small edge over a trivial predictor, but this is the best that can be done with the tangent kernel. However, in a differ-

entiable model, once this linear predictor is picked up, its support can be leveraged to output a parity of these bits, instead of their sum, thereby obtaining zero error.

Formally, for integers $2 \le k \le n$ and for $\alpha \in (0,1)$, we consider the "biased sparse parities" problem $\mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$ over $\mathcal{X} = \{-1,1\}^n$ and $\mathcal{Y} = \{-1,1\}$, and learning w.r.t. the square loss. The problem consists of $\binom{n}{k}$ distributions over $\mathcal{X} \times \mathcal{Y}$, each corresponding to a subset $I \subseteq [n]$ with $|I| = k$. The distribution $\mathcal{D}_I$ is defined by the following sampling procedure:

- ▷ Let $\mathcal{D}_0$ be the uniform distribution over $\{-1,1\}^n$ and let $\mathcal{D}_1$ be the product distribution over $\{-1,1\}^n$ with $\mathbb{E}\, x_i = \frac{1}{2}$. Sample$x \sim \mathcal{D}_{\mathcal{X}} := (1-\alpha)\mathcal{D}_0 + \alpha\mathcal{D}_1$.
- ▷ Set $y \leftarrow \chi_I(x) := \prod_{i \in I} x_i$, the parity over the subset I.

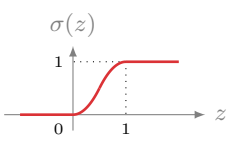
The differentiable model. To learn $\mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$, we construct a differentiable model that is a combination of a linear predictor $\langle \theta, x \rangle$ and a "selected-parity", which outputs the parity of the subset of bits indicated by the (essential) support of θ, that is, $\prod_{|\theta_i| \ge \nu} x_i$ (for a suitable threshold ν). Importantly, both use the same weights θ (see Figure 1). The weak edge of the linear predictor allows gradient descent to learn a parameter vector θ such that $\{i \mid |\theta_i| \ge \nu\} = I$, and once this is learned, the "selected-parity" kicks in and outputs a perfect predictor.

Formally, we construct a differentiable model $f_\theta(x)$ with $\theta \in \mathbb{R}^n$ (i.e. size $p = n$) that behaves as follows for all $\theta \in \left(\left[-\frac{2}{n}, \frac{2}{n}\right] \cup \left[\frac{3}{n}, \frac{5}{n}\right]\right)^n$:

$$f_\theta(x) \begin{cases} \approx 2\langle \theta, x \rangle & \text{if } \theta \approx 0 \\ = \prod_{i\,:\,\theta_i \ge \frac{3}{n}} x_i & \text{if } \exists i \,:\, \theta_i \ge \frac{3}{n} \end{cases}, \tag{8}$$

where $\approx$ stands for the first order approximation of f_θ about $\theta = 0$. The behaviour (8) is the *only* property we need to show that approximate gradient descent learns $\mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$: as formalized in Claim 1, a single step of gradient descent starting at $\theta^{(0)} = 0$ will leave $\theta_i^{(1)} \in \left[-\frac{2}{n}, \frac{2}{n}\right]$ for $i \notin I$, while increasing $\theta_i^{(1)} \ge \frac{3}{n}$ for $i \in I$, thus yielding the correct parity. We show how to implement f_θ as a continuous differentiable model with scale $C_f = O(n)$, and furthermore how this can implemented as a feed-forward neural network with piece-wise quadratic sigmoidal activations:

$$\sigma(z) = \begin{cases} 0 & z < 0 \\ 2z^2 & z \in [0, \frac{1}{2}] \\ 4z - 2z^2 - 1 & z \in [\frac{1}{2}, 1] \\ 1 & z > 1 \end{cases}$$



The model f_θ, illustrated in Figure 1, is defined below, where $\theta \circ x := (\theta_1 x_1, \ldots, \theta_n x_n)$ and $\sigma_{a,b}^{c,d}(z) := c + \sigma\left(\frac{z-a}{b-a}\right)(d-c)$, defined for all $a < b$ and c, d; it is defined such that (i) $\sigma_{a,b}^{c,d}(z) = c$ for every $z \le a$, (ii) $\sigma_{a,b}^{c,d}(z) = d$ for every $z \ge b$ and (iii) $\left|\frac{d}{dz}\sigma_{a,b}^{c,d}(z)\right| \le \frac{2|d-c|}{b-a}$.

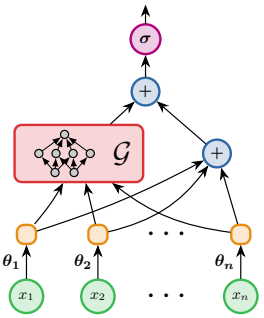


Figure 1. Schematic diagram for the construction of f_θ, as used in Claim 1, with only trainable parameters being $\theta_1, \theta_2, \ldots, \theta_n$. The sub-network $\mathcal{G}$ is a fixed module implementing the "selected-parity" function.

$$f_\theta(x) := \sigma_{-1,1}^{-1,1}\left(\langle \theta, x \rangle + \mathcal{G}(\theta \circ x)\right) \tag{9}$$
$$\mathcal{G}(z) := S(\xi(z)) \cdot (H(\xi(z)) - \textstyle\sum_i z_i) \tag{10}$$
$$S(s) := 1 - \textstyle\prod_{i=1}^n (1 - s_i^2) \tag{11}$$
$$H(s) := \textstyle\prod_{i=1}^n (1 + s_i - s_i^2) \tag{12}$$
$$\xi(z)_i := \sigma(nz_i - 2) - \sigma(-nz_i - 2) \tag{13}$$

The intuition for $\mathcal{G}$ is as follows. In the relevant regime of $\theta \in \left(\left[-\frac{2}{n}, \frac{2}{n}\right] \cup \left[\frac{3}{n}, \frac{5}{n}\right]\right)^n$ and any $x \in \{-1,1\}^n$ we have that $s = \xi(\theta \circ x) \in \{-1, 0, 1\}^n$, with $s_i = x_i$ if $\theta_i \in [\frac{3}{n}, \frac{5}{n}]$ and $s_i = 0$ if $\theta_i \in [-\frac{2}{n}, \frac{2}{n}]$. In this regime of θ, we have $S(s) = \mathbf{1}\{s \ne 0\}$ and $H(s) = \prod_{i:s_i \ne 0} s_i$. Thus, when $\theta_i \in [-\frac{2}{n}, \frac{2}{n}]$ for all i, we have $S(\xi(\theta \circ x)) = 0$ and hence $\mathcal{G}(\theta \circ x) = 0$ for all $x \in \{-1,1\}^n$. On the other hand, when $\theta_i \in [\frac{3}{n}, \frac{5}{n}]$ for some i, we have $S(\xi(\theta \circ x)) = 1$ and hence $\mathcal{G}(\theta \circ x) = \prod_{i\,:\,\theta_i \ge \frac{3}{n}} x_i - \langle \theta, x \rangle$ for all $x \in \{-1,1\}^n$. This gives us (8), by noting that $\sigma_{-1,1}^{-1,1}(z) \approx 2z$ at $z \approx 0$ (first order approximation) and $\sigma_{-1,1}^{-1,1}(z) = \mathrm{sign}(z)$ when $|z| \ge 1$. While we presented f_θ as a differentiable model using the gadget $\mathcal{G}$, it is also possible to implement f_θ as a neural network (with some untrainable weights); see Remark 11 in Appendix D.

We can also calculate that for any $i \in [n]$, θ and $x \in \{-1,1\}^n$, it holds that $|\frac{\partial}{\partial \theta_i} f_\theta(x)| \le O(n)$ and $|f_\theta(x)| \le 1$. Thus, we get that $C_f^2 = \sup_{\theta,x} \|\nabla_\theta f_\theta(x)\|^2 + f_\theta(x)^2 \le O(n^2)$.

The following Claim (proved in Appendix D.1) formalizes how a single step of gradient descent is sufficient for θ to be away from zero on I, and hence for the network to output the correct labels.

Claim 1. *For any n, k and $\alpha \in (0,1)$, and any $\mathcal{D}_I \in \mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$, τ-approximate gradient descent on the model f of size $p = n$ and $C_f = O(n)$ described above, with initialization $\theta_0 = 0$ (at which $\forall_x f_{\theta_0}(x) = 0$), accuracy $\tau \le \alpha/2^k$, step size $\eta = 2^k/(\alpha n)$ and $T = 1$ step en-*

sures $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(T)}}) = 0$. *In particular, if* $k \leq \log n$ *then an accuracy of* $\tau \leq \alpha/n$ *is sufficient.*

On the other hand, the next claim (proof in Appendix F.1.1), establishes that the tangent kernel at initialization only has a small edge over the null prediction.

Claim 2. *The tangent kernel of the model* f *at* $\theta_0 = 0$ *is the scaled linear kernel* $\mathsf{NTK}^f_{\theta_0}(x, x') = 4\langle x, x'\rangle$, *and for any* $2 \leq k \leq n$, $\alpha \in (0,1)$ *and* $\mathcal{D}_I \in \mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$ *the error with this kernel is* $\inf_{h \in \mathsf{NTK}^f_{\theta_0}(B)} \mathcal{L}_{\mathcal{D}_I}(h) \geq \frac{1}{2} - \frac{\alpha}{2} = \frac{1}{2} - O\left(\frac{n^2\tau}{C_f}\right)$ *for all* $B \geq 0$, *where* $\tau = \alpha/2^k$ *as in Claim 1. On the other hand,* $\exists h \in \mathsf{NTK}^f_{\theta_0}(n)$ *s.t.* $\mathcal{L}_{\mathcal{D}_I}(h) \leq \frac{1}{2} - \frac{\alpha^2}{2^{2k}} = \frac{1}{2} - \Omega\left(\frac{n^2\tau^2}{C_f^2}\right)$.

We already see that gradient descent can succeed where the tangent kernel at initialization can only ensure an arbitrarily small edge over the error achieved by null prediction, $\mathcal{L}(0) = 1/2$:

Separation 1. *For any* $\gamma > 0$, *there exists a source distribution (with binary labels and w.r.t. the square loss), s.t.*

- ▷ *[Gradient Descent] Using a differentiable model with* $p = 2$ *parameters,* $T = 1$ *steps, and accuracy* $\frac{\tau}{C_f} = \Theta(\gamma)$, τ*-approximate gradient descent can ensure zero squared-loss, but*
- ▷ *[Tangent Kernel] The tangent kernel of the model at initialization does not ensure square loss lower than* $\frac{1}{2} - \gamma = \frac{1}{2} - \Theta(\tau/C_f)$ *(compare to the null prediction that always has square loss* $\mathcal{L}(0) = \frac{1}{2}$*).*

Furthermore, the gradient descent algorithm has an initialization θ_0 *s.t.* $\forall_x f_{\theta_0}(x) = 0$.

Proof. Apply Claims 1 and 2 to the sole source distribution in $\mathcal{P}^{\mathsf{bsp}}[n = 2, k = 2, \alpha = 2\gamma]$, (corresponding to $I = \{1, 2\}$). □

Even though the tangent kernel at the initialization used by gradient descent might have an arbitrarily small edge, one might ask whether better error can be ensured by the tangent kernel at some other "initialization" θ, or perhaps by the tangent kernel of some other model, or perhaps even some other kind of kernel[8]. The following claim shows that this is not the case (proof in Appendix F.2.1).

Claim 3. *For all* $\alpha < 1/2, k, p, B, n$, *and any kernel* K *corresponding to a* p*-dimensional feature map, there exists* $\mathcal{D}_I \in \mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$ *such that* $\inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \geq \mathcal{L}_{\mathcal{D}_I}(0) - \frac{\alpha}{2} - \min\left\{\frac{p}{2|\mathcal{P}^{\mathsf{bsp}}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}^{\mathsf{bsp}}|^{1/3}}\right)\right\}$, *where* $\mathcal{L}_{\mathcal{D}_I}$ *is w.r.t. the square loss, and note that* $|\mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]| = \binom{n}{k}$.

And so, setting $k = \Theta(\log n)$ and $\alpha = 1/\operatorname{poly}(n)$, we see that gradient descent with polynomial parameters can learn $\mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$, while no tangent kernel of a polynomial sized model (i.e. with $p = \operatorname{poly}(n)$) can ensure better than an arbitrarily small polynomial edge; see Appendix G for full proof.

Separation 2. *For any sequence* $\gamma_n = 1/\operatorname{poly}(n)$, *there exists a sequence of learning problems* $\mathcal{P}_n$ *with fixed input distributions (with binary labels and w.r.t. the square loss,), such that*

- ▷ *[Gradient Descent] for each* n, *using a differentiable model with* $p = n$ *parameters, realizable by a neural network of depth* $O(\log n)$ *with* $O(n)$ *edges (where some edges have trainable weights and some do not),* $T = 1$ *steps, polynomial accuracy* $\frac{\tau}{C_f} = O\left(\frac{\gamma_n}{n^2}\right)$, *and initialization* θ_0 *s.t.* $\forall_x f_{\theta_0}(x) = 0$, τ*-approximate gradient descent learns* $\mathcal{P}_n$ *to zero loss; but*
- ▷ *[Poly-Sized Kernels] no sequence of kernels* K_n *corresponding to feature maps of dimension* $\operatorname{poly}(n)$ *(and hence no tangent kernel of a poly-sized differentiable models) can allow learning* $\mathcal{P}_n$ *to square loss better than* $\frac{1}{2} - \gamma_n$ *for all* n*; and*
- ▷ *[Arbitrary Kernel, Poly Norm] no sequence of kernels* K_n *of any dimension can allow learning* $\mathcal{P}_n$ *to square loss better than* $\frac{1}{2} - \gamma_n$ *for all* n *using predictors in* $\mathcal{F}(K_n, B_n)$ *of norm* $B_n = \operatorname{poly}(n)$.

Empirical Demonstration in Two-Layer Networks. While for ease of analysis we presented a fairly specific model, with many fixed (non-trainable) weights and only few trainable weights, we expect the same behaviour occurs also in more natural, but harder to analyze models. To verify this, we trained a two-layer fully-connected ReLU network on the source distribution $\mathcal{D}_\alpha$ analyzed above, for $n = 128$ and $k = 7$. We observe that indeed when $\alpha > 0$, and thus a linear predictor has at least some edge, gradient descent training succeeds in learning the sparse parity, while the best predictor in the Tangent Kernel cannot get error much better than 0.5. See Figure 2 for details.

4. Ensuring the Tangent Kernel has an Edge

In the example of Section 3 we saw that the Tangent Kernel at initialization cannot ensure small error, and even though Gradient Descent finds a perfect predictor, the tanget kernel only has an arbitarily small edge over the null prediction. But this edge isn't zero: the second part of Claim 2 tells us that at least in the example of Section 3, the tangent kernel *will* have an edge, and this edge is polynomial (even linear) in the accuracy required of gradient descent. Is this always the case? We now turn to asking whether whenever gradient

[8] For each instance $\mathcal{D}_I \in \mathcal{P}^{\mathsf{bsp}}[n,k,\alpha]$, there is of course always a point θ at which the tangent kernel allows for prediction matching that of Gradient Descent, namely the iterate $\theta^{(T)}$ reached by Gradient Descent. But to *learn* using a kernel, the kernel should be chosen without knowing the instance, i.e. without knowing I.

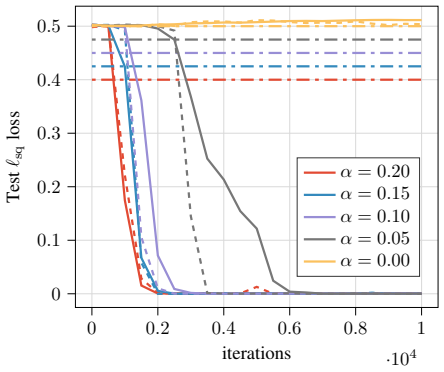


Figure 2. Two-layer fully-connected ReLU network (128 hidden units) training on data sampled from $\mathcal{D}_I$ with $n = 128$, $k = 7$ for different values of α, trained using Adam optimizer with learning-rate of 0.01. Solid lines show the test performance at each iteration averaged over 20 runs. Dashed lines show the test performance of the model with best test performance of the last iterate between the 20 runs. The horizontal dot-dash lines indicate the lower bound (from Claim 3) on best accuracy attainable by any kernel method with feature map with as many number of parameters as the NTK.

descent succeeds in ensuring small error, then the Tangent Kernel must indeed have an edge that is at least polynomial in the gradient descent parameters.

4.1. With Unbiased Initialization

We begin by considering situations where gradient descent succeeds starting at an initialization yielding zero predictions on all inputs, i.e. such that:

$$\forall x \; : \; f_{\theta_0}(x) \; = \; 0. \tag{14}$$

Following Chizat et al. (2019), we refer to θ_0 satisfying (14) as *unbiased* initializations. With an unbiased initialization, the Taylor expansion (3) does not have the zero-order, or "bias" term, there is no need for the first coordinate in the feature vector ϕ_{θ_0}, and the Tangent Kernel becomes $\mathsf{NTK}^f_{\theta_0}(x, x') = \langle \nabla_\theta f_{\theta_0}(x), \nabla_\theta f_{\theta_0}(x') \rangle$, simplifying the Neural Tangent Kernel analysis. Chizat et al. (2019), Woodworth et al. (2020) and others discuss how to construct generic unbiased initializations, namely by including pairs of units that cancel each other out at initialization, but quickly diverge during training and allow for meaningful learning. The initialization used in Claim 1 of Section 3 also satisfies (14).

In Theorem 4 below, we show that if gradient descent improves over the loss at initialization, then the Tangent Kernel must also have an edge over initialization, which for an unbiased initialization means an edge over the null prediction.

We can also considered a relaxation of the unbiasedness assumption (14), and instead require only that the output of the network at initialization is very close to zero. This would be the case with some random initialization schemes which initialize the network randomly, but with small magnitude. As long as the deviation from unbiasedness is smaller than the edge guaranteed, Theorem 4 would still ensure that the tangent kernel has an edge over null prediction.

Theorem 4. *For any smooth loss function ℓ, differentiable model f, initialization θ_0, and source distribution $\mathcal{D}$, if τ-accurate gradient descent for T steps ensures error $\mathcal{L}_\mathcal{D}(f_{\theta^{(T)}}) \leq \varepsilon$, then for $B = C_f + \frac{\tau}{|\ell''|C_f}$, there exists $h \in \mathsf{NTK}^f_{\theta_0}(B)$ with*

$$\mathcal{L}_\mathcal{D}(h) \leq \max\left(\varepsilon, \; \mathcal{L}_\mathcal{D}(f_{\theta_0}) - \frac{\tau^2}{2|\ell''|C_f^2}\right). \tag{15}$$

In particular, if θ_0 is an **unbiased initialization** *and $\varepsilon < \mathcal{L}_\mathcal{D}(0)$, then $\mathcal{L}_\mathcal{D}(h) \leq \mathcal{L}_\mathcal{D}(0) - \frac{\tau^2}{2|\ell''|C_f^2}$.*

At a high level, we argue that if gradient descent is going to move at all, the gradient at initialization must be substantially non-zero, and hence move the model in some direction that improves over initialization. But since the tangent kernel approximates the true model close to the initialization, this improvement translates also to an improvement over initialization also in the Tangent Kernel. If the initialization is unbiased, initialization is at the null predictor, and this is an improvement over null predictions. The details of the proof are presented in Appendix E.1.

Separation 1 establishes that Theorem 4 is polynomially tight: it shows that despite gradient descent succeeding with an unbiased initialization, the tangent kernel at initialization has an edge of at most $O(\tau/C_f)$. This leaves a quadratic gap versus the guarantee of $\Omega(\tau^2/C_f^2)$ in the Theorem (recalling $|\ell''| = 1$ for the square loss), but establishes the polynomial relationship.

Theorem 4 is a strong guarantee, in that it holds for each source distribution separately. This immediately implies that if τ-approximate gradient descent on a differentiable model f with unbiased initialization θ_0 learns some family of distribution $\mathcal{P}$ to within small error, or even just to with some edge over the null prediction, then the Tangent Kernel at θ_0 also has an edge of at least $\frac{\tau^2}{2|\ell''|C_f^2}$ over the null prediction. Note that this edge is polynomial in the algorithm parameters τ and C_f, as is the required norm B. And so, if, for increasing problem sizes n, gradient descent allows for "polynomial learnability", in the sense of learning with polynomially increasing complexity, then the Tangent Kernel at least allows for "weak learning" with a polynomial edge (in all the parameters, and hence in the problem size n). Separation 2 shows that this relationship is tight, in that there are indeed problems where strong learning is possible with gradient descent with polynomial complexity, but the tanget kernel, or indeed any kernel of polynomial complexity, can only ensure polynomially weak learning.

Theorem 4 relies on the initialization being unbiased, or at the very least the predictions at initialization not being worse than the null predictions. Can we always ensure the initialization satisfies such a condition? And what happens

if it does not? Might it then be possible for Gradient Descent to succeed even though the Tangent Kernel does not have an edge over the null prediction?

The example in Section 5 shows that, indeed, some learning problems might be learnable by gradient descent, but only using an initilization that is *not* unbiased. That is, we should not expect initializations to always be unbiased, nor can we expect to always be able to "fix" the initialization to be unbiased. And furthermore, the example shows that with an initilization that is not unbiased, the Tangent Kernel at initilization might not have a significant edge over the null predictor.

But before seeing this example, let us ask: What can we ensure when the initialization is not unbiased?

4.2. With Distribution Dependence

In Theorem 5 we show that, at least for the square loss, for an arbitrary initialization, even though the Tangent Kernel at initialization might not have an edge, we can construct a distribution $\mathcal{W}$, which we can think of as an alternate "random initilization", such that the Tangent Kernel of the model at a random point $\theta \sim \mathcal{W}$ drawn from this distribution, *does* have an edge over null prediction. But the catch is that this distribution depends not only on the true initialization θ_0, but also on the input distribution $\mathcal{D}_{\mathcal{X}}$. That is, the Tangent Kernel for which we are establishing an edge is *distribution dependent*. In Section 5 we will see that this distribution-dependence is unavoidable.

Theorem 5. *For a differentiable model f, initialization θ_0, stepsize η, number of iterations T, and the square loss, given an input distribution $\mathcal{D}_{\mathcal{X}}$, there exists a distribution $\mathcal{W}$ (that depends on $\mathcal{D}_{\mathcal{X}}$), supported on $T+1$ parameter vectors in $\mathbb{R}^p$, such that for any source distribution $\mathcal{D}$ that agrees with $\mathcal{D}_{\mathcal{X}}$ and for which τ-approximate gradient descent ensures error $\mathcal{L}_{\mathcal{D}}(f_{\theta^{(T)}}) < \mathcal{L}_{\mathcal{D}}(0) - \gamma$, we have*

$$\mathop{\mathbb{E}}_{\theta\sim\mathcal{W}} \inf_{h\in \mathsf{NTK}^f_\theta(B)} \mathcal{L}_{\mathcal{D}}(h) < \mathcal{L}_{\mathcal{D}}(0) - \gamma' \qquad (16)$$

where $\gamma' = \min\left\{\frac{\gamma}{T+1}, \frac{\tau^2}{2C_f^2}\right\}$ and $B = \sqrt{2\gamma' C_f}$. And so, the Average Tangent Kernel $K = \mathbb{E}_{\theta\sim\mathcal{W}} \mathsf{NTK}^f_\theta$ has rank at most $(T+1)(p+1)$ and $\inf_{h\in\mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}}(h) \le \mathcal{L}_{\mathcal{D}}(0) - \gamma'$.

The proof details are presented in Appendix E.2.

5. No Edge with the Tangent Kernel

As promised, we will now present an example establishing:

1. Some learning problems are learnable by gradient descent, but only with a biased initialization
2. Theorem 4 cannot be extended to biased initializations: Even if gradient descent, with a *biased* initialization, ensures small error, the Tangent Kernel at initialization might not have a significant edge.
3. Theorem 5 cannot be made distribution-independent: Even if gradient descent, with a biased initialization, ensures small error on a learning problem, there might not be any *distribution-independent* random "initialization" that yields a significant edge in a Tangent Kernel at a random draw from this "initialization".

The Learning Problem. This time we will consider learning parities of any cardinality (not necessarily sparse), but we will "leak" the support of the parity through the input distribution $\mathcal{D}_{\mathcal{X}}$. For an integer n and $\alpha \in (0,1)$, we consider the "leaky parities" problem $\mathcal{P}^{\mathsf{lp}}[n,\alpha]$ over $\mathcal{X} = \{-1,1\}^n$ and $\mathcal{Y} = \{-1,1\}$ with $2^n - n - 1$ distributions, each corresponding to $I \subseteq [n]$ with $|I| \ge 2$. Each source distribution $\mathcal{D}_I$ is a mixture of two components $\mathcal{D}_I = (1-\alpha)\mathcal{D}_I^{(0)} + \alpha\mathcal{D}_I^{(1)}$: the first component $\mathcal{D}_I^{(0)}$ has labels $y = \chi_I(x)$ corresponding to parity of bits in I, with a uniform marginal distribution over x. The second component has uniform random labels y independent of the inputs (i.e. the labels in this component are entirely uninformative), but the input is deterministic and set to $x := x^I$ where $x_i^I = 1$ for $i \in I$ and -1 for $i \notin I$. Learning this problem is very easy: all a learner has to do is examine the marginals of each input coordinate of x_i. Those in the support I will have $\mathbb{E}_{\mathcal{D}_I}[x_i] = \alpha$ while when $i \notin I$ we have $\mathbb{E}_{\mathcal{D}_I}[x_i] = -\alpha$. The marginals over x_i thus completely leak the optimal predictor. But as we shall see, while gradient descent can leverage the marginals in this way, kernel methods (where the kernel is pre-determined and does not depend on the input distribution) fundamentally cannot.

More formally, $\mathcal{D}_I^{(0)}$ is the distribution obtained by sampling $x \sim \mathcal{U}(\{\pm 1\}^n)$ and setting $y = \chi_I(x) := \prod_{i\in I} x_i$, while $\mathcal{D}_I^{(1)}$ is the distribution obtained by (deterministically) setting $x := x^I$ and sampling $y \sim \mathcal{U}(\{-1,1\})$, and recall $\mathcal{D}_I := (1-\alpha)\mathcal{D}_I^{(0)} + \alpha\mathcal{D}_I^{(1)}$. In other words, $\mathcal{D}_I$ corresponds to first choosing $b \in \{0,1\}$ with $\Pr[b=1] = \alpha$ and sampling $(x,y) \sim \mathcal{D}_I^{(b)}$.

The differentiable model. To learn $\mathcal{P}^{\mathsf{lp}}[n,\alpha]$, we construct a differentiable model that is very similar to the one in Section 3, being a combination of a linear predictor and a "selected-parity", but where the linear component has a bias term, and is thus equal to -1 (rather then zero) at initialization. Formally, we construct f_θ with $\theta \in \mathbb{R}^n$ (i.e. size $p = n$) that behaves as follows for all $\theta \in \left(\left[0, \frac{2}{n}\right] \cup \left[\frac{3}{n}, \frac{5}{n}\right]\right)^n$:

$$f_\theta(x) \begin{cases} \approx -1 + 2\left\langle \theta, x + \frac{5}{3}\alpha\mathbf{1}\right\rangle & \text{if } \theta \approx 0 \\ = \prod_{i\,:\,\theta_i \ge \frac{3}{n}} x_i & \text{if } \exists i \,:\, \theta_i \ge \frac{3}{n} \end{cases}, \qquad (17)$$

where $\approx$ stands for the first order approximation of f_θ about $\theta = 0$, and $\mathbf{1} \in \mathbb{R}^n$ is the all-1s vector. Again, (17) is the *only* property we need, to show that approximate gradient descent learns $\mathcal{P}^{\mathsf{lp}}[n,\alpha]$: at initialization, we output -1 for all examples, meaning the derivative of the loss is non-negative, and we generally want to try to increase the output of the model. How much we increase each coordinate θ_i

will depend on $\mathbb{E}[x_i + \frac{5}{3}\alpha]$, which is $8/3$ for $i \in I$ but only $2/3$ for $i \notin I$, and the first step of gradient descent will separate between the coordinates inside and outside the support I, thus effectively identifying I. With an appropriate stepsize, the step will push θ into the regime covered by the second option in (17), and the model will output the desired parity[9]. The key ingredient here is that even though all the coordinates are uncorrelated with the labels, and thus also with residuals at an *unbiased* initialization, and the model of Section 3 will have zero gradient at initialization, we artificially make all the residuals at initialization non-positive, thus creating a correlation between each coordinate and the residual, which moves θ in a way that depends on the marginal $\mathcal{D}_X$ (as in the proof of Theorem 5).

As with the model of Section 3, we can implement (17) as a continuous differentiable model with scale $C_f = O(n)$, through a slight modification of the architecture described in Equations (9)-(13):

$$f_\theta(x) := \sigma^{-1,0}_{0,\frac{2}{n}}\left(\langle\theta,\mathbf{1}\rangle\right) + \sigma^{-1,1}_{-1,1}\left(\left\langle\theta, x+\tfrac{5}{3}\alpha\mathbf{1}\right\rangle + \mathcal{G}(\theta,x)\right) \quad (18)$$

$$\mathcal{G}(\theta,x) := S(\xi(\theta\circ x)) \cdot \left(H(\xi(\theta\circ x)) - \left\langle\theta, x+\tfrac{5}{3}\alpha\mathbf{1}\right\rangle\right) \quad (19)$$

where S, H and ξ are as defined in (11), (12) and (13). The first term in (18) is -1 at $\theta^{(0)} = 0$ and satisfies $\nabla_\theta\, \sigma^{-1,0}_{0,\frac{2}{n}}\left(\left\langle\theta^{(0)},\mathbf{1}\right\rangle\right) = 0$. The second term is essentially the same as (9), with only difference being the linear term $\langle\theta, x\rangle$ is replaced with $\left\langle\theta, x+\frac{5}{3}\alpha\mathbf{1}\right\rangle$. Similar to the construction in Section 3, we get that (17) holds and that $C_f^2 \le O(n^2)$.

We prove the following claims in Appendix D.2 and Appendix F.1.2 respectively (cf. Claims 1 and 2).

Claim 6. *For any n and $\alpha \in (0,1)$, for any $\mathcal{D}_I \in \mathcal{P}^{\mathsf{lp}}[n,\alpha]$, gradient descent on the model f of size $p = n$ with $C_f^2 = O(n^2)$ described above, with initialization $\theta_0 = 0$, accuracy $\tau = \frac{4}{3}\alpha$, step size $\eta = 1$ and $T = 1$ step ensures error $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(T)}}) \le \alpha$.*

Claim 7. *The tangent kernel of the model f at $\theta_0 = 0$ is the scaled linear kernel endowed with a bias term, $\mathsf{NTK}^f_{\theta_0}(x,x') = (1+\frac{100}{9}\alpha^2) + 4\langle x, x'\rangle$, and for any $n \ge 2$, $\alpha \in (0,1)$, and any $\mathcal{D}_I \in \mathcal{P}^{\mathsf{lp}}[n,\alpha]$, the error with this kernel is $\inf_{h\in\mathsf{NTK}^f_{\theta_0}(B)} \mathcal{L}_{\mathcal{D}_I}(h) = \mathcal{L}_{\mathcal{D}_I}(0) = \frac{1}{2}$ for any $B \ge 0$.*

We already see that in contrast to a situation where the initialization is unbiased, and so Theorem 4 ensures the tangent kernel has at least a polynomial edge, even if gradient descent succeeds, but with an initialization which is not unbiased, the tangent kernel at initialization might not have any edge at all, and so we cannot hope to remove the requirement of the initialization being unbiased from Theorem 4:

Separation 3. *For any $\varepsilon > 0$, there exists a source distribution (with binary labels and w.r.t. the square loss), s.t.*

- ▷ *[Gradient Descent] Using a differentiable model with $p = 2$ parameters, $T = 1$ steps, and accuracy $\frac{\tau}{C_f} = \Theta(\varepsilon)$, τ-approximate gradient descent ensures square loss of ε, but*
- ▷ *[Tangent Kernel] The tangent kernel of the model at initialization cannot ensure square loss lower than $\frac{1}{2}$ (which is the loss of null prediction).*

Proof. Apply Claims 6 and 7 to the sole source distribution in $\mathcal{P}^{\mathsf{lp}}[n=2,\alpha=2\varepsilon]$. □

But what about the tangent kernel at a different "initialization", or of some other model, or even some other kernel altogether? The following Claim (proved in Appendix F.2.2) shows that no kernel can get a significant edge over the zero predictor unless the number of features and the norm are both exponentially large in n. In order to contrast with guarantee (16) of Theorem 5, we state the Claim also for learning using a random kernel, i.e. in expectation over an arbitrary distribution of kernels.

Claim 8. *For all $\alpha \in (0,1), p, B, n$, and any randomized kernel K with rank$(K) = p$ almost surely (i.e. a distribution over kernels, each of which corresponding to a p-dimensional feature map), there exists $\mathcal{D}_I \in \mathcal{P}^{\mathsf{lp}}[n,\alpha]$ such that $\mathbb{E}_K \inf_{h\in\mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \ge \mathcal{L}_{\mathcal{D}_I}(0) - \min\left\{\frac{p}{2|\mathcal{P}^{\mathsf{lp}}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}^{\mathsf{lp}}|^{1/3}}\right)\right\}$. where $\mathcal{L}_{\mathcal{D}_I}$ is w.r.t. the square loss, and note that $|\mathcal{P}^{\mathsf{lp}}[n,\alpha]| = 2^n - n - 1$.*

In particular, we see that in contrast to the distribution-dependent situation of Theorem 5, where we could ensure a polynomial edge for the tangent kernel at a specified randomized "initialization", this is not possible in general, and we cannot hope to remove the dependence on the input distribution in Theorem 5.

Separation 4. *For any sequence $\varepsilon_n = 1/\operatorname{poly}(n)$, there exists a sequence of learning problem $\mathcal{P}_n$ (with binary labels and w.r.t. the square loss), such that*

- ▷ *[Gradient Descent] for each n, using a differentiable model with $p = n$ parameters, realizable by a neural network of depth $O(\log n)$ and $O(n)$ edges (where some edges have trainable weights and some do not), $T = 1$ steps, and accuracy $\frac{\tau}{C_f} = O(\frac{\varepsilon_n}{n})$, τ-approximate gradient descent learns $\mathcal{P}_n$ to square loss of ε_n, but*
- ▷ *[Poly-Sized Kernels] no sequence of (randomized) kernels K_n corresponding to feature maps of dimension $\operatorname{poly}(n)$ (and hence no tangent kernel of a poly-sized differentiable models, even with randomized "initialization") can allow*

[9] The reason for the offset $\frac{5}{3}$ is to ensure all coordinates will be non-negative after the first step, which simplifies identification of the two regimes in (17) when implementing the model

learning $\mathcal{P}_n$ (in expectation) to square loss smaller than $\frac{1}{2} - 2^{-\Omega(n)}$ for all n; and

- ▷ *[Arbitrary Kernel, Poly Norm] no sequence of (randomized) kernels K_n of any dimension can allow learning $\mathcal{P}_n$ (in expectation) to square loss smaller than $\frac{1}{2} - 2^{-\Omega(n)}$ for all n using predictors in $\mathcal{F}(K_n, B_n)$ of norm $B_n = \mathrm{poly}(n)$.*

Proof. Apply Claims 6 and 8 to $\mathcal{P}_n = \mathcal{P}^{\mathsf{lp}}[n, \alpha = \varepsilon_n]$. □

Since we are working over $\mathcal{X} = \{-1, +1\}^n$ of cardinality 2^n, we can always ensure an edge of $(1-\alpha)p/2^{n+1}$ using indicator features on p of the 2^n inputs, thus allowing memorization of these tiny fraction of inputs. An edge of $2^{-\Theta(n)}$ is thus a "trivial" edge attained by this kind of memorization, and Claim 8 and Separation 4 establish the tangent kernel, or even any other kernel, cannot be significantly better.

We can also conclude that even though we saw that our learning problem is learnable with gradient descent, we cannot hope to learn it with an unbiased initialization, since this would imply existence of a kernel with a polynomial edge. We thus get the following separation between learnability with gradient descent using biased versus unbiased initialization (importantly, only for distribution *independent* learning); proof in Appendix G.

Separation 5. *For any sequence $\varepsilon_n = 1/\mathrm{poly}(n)$, there exists a sequence of learning problem $\mathcal{P}_n$ (with binary labels and w.r.t. the square loss), such that*

- ▷ *[GD with **biased** initialization] for each n, using a differentiable model with $p = n$ parameters, realizable by a neural network of depth $O(\log n)$ and $O(n)$ edges (where some edges have trainable weights and some do not), $T = 1$ step, and accuracy $\frac{\tau}{C_f} = O\left(\frac{\varepsilon_n}{n}\right)$, and some (**not unbiased**) initialization, τ-approximate gradient descent learns $\mathcal{P}_n$ to square loss ε_n, but*
- ▷ *[GD with **unbiased** initialization] It is not possible to ensure error better than $\frac{1}{2}$ on $\mathcal{P}_n$, for all n, using τ-approximate gradient descent starting with an **unbiased** initialization (as in Eq. (14)), and any differentiable models of size $p = \mathrm{poly}(n)$ and accuracy $\tau/C_f = 1/\mathrm{poly}(n)$, and any number of steps T.*

6. Conclusion and Discussion

With the study of Neural Tangent Kernels increasing in popularity, both as an analysis and methodological approach, it important to understand the limits of the relationship between the Tangent Kernel approximation and the true differentiable model. Furthermore, the notion of "gradual" learning of deep models, where we learn progressively more complex models, or more layers, and so the success of deep learning rests on being able to make progress even with simpler, e.g. linear models, is an appealing approach to understanding deep learning. Indeed, when we first asked ourselves whether the tangent kernel must always have an edge in order for gradient descent to succeed, and we sought to quantify how large this edge must be, we were guided also by understanding the "gradual" nature of deep learning. We were surprised to discover that in fact, with biased initialization, deep learning can succeed even without the tangent kernel having a significant edge.

Our results also highlight the importance of the distinction between distribution dependent and independent learning, and between biased and unbiased initialization. The gap between distribution dependent and independent learning relates to kernel (i.e. linear) methods inherently not being able to leverage information in $\mathcal{D}_{\mathcal{X}}$: success or failure is based on whether $y|x$ is well represented by a predictor in $\mathcal{F}(K, B)$, and has little to do with the marginal over x. In contrast, gradient descent on a non-linear model, could behave very differently depending on $\mathcal{D}_{\mathcal{X}}$, as we also see empirically in the experiment in Figure 2. Perhaps even more surprising is the role of the bias of the initialization. It might seem like a benign property, and that we should always be able to initialize with zero, or nearly-zero predictions, or at least at θ_0 that is not much worse than null, or perhaps correct for the bias as in Chizat et al. (2019). But we show that at least for distribution-independent learning, this is not a benign property at all: for some problems we *must* use biased initialization (Separation 5). This observation may be of independent interest, beyond the role it plays in understanding the Neural Tangent Kernel.

The learning problems and models we used to demonstrate the separation results are artificially extreme, so as to push the separation to the limit, and allow easy analytical study. But we believe they do capture ways in which gradient descent is more powerful than kernel methods. In the example of Section 3, gradient descent starts by selecting a few "simple features" (the $k \ll n$ relevant coordinates), based on simple correlations. But unlike kernel methods, gradient descent is then able to use these features in more complex ways. We see this happening also empirically in Figure 2 with a straightforward two-layer ReLU network, where gradient descent is able to succeed in learning the complex parity function, once there is enough signal to easily identify the few relevant coordinates. In the example of Section 5, we see how gradient descent is also able to pick up on structure in the input (unlabeled data) distribution, in a way that kernel methods are fundamentally unable to.

Acknowledgements

We thank Gilad Yehudai for clarifying our questions about Yehudai & Shamir (2019), Jascha Sohl-dickstein for pointing us to references on NTK and anonymous reviewers for helpful comments. This work was done while NS was visiting EPFL. This research is part of the NSF/Simons funded *Collaboration on the Theoretical Foundations of Deep Learning* (deepfoundations.ai). PK was partially supported by NSF BIGDATA award 1546500.

References

Abbe, E. and Sandon, C. Poly-time universality and limitations of deep learning. *arXiv*, abs/2001.02992, 2020a. URL http://arxiv.org/abs/2001.02992.

Abbe, E. and Sandon, C. On the universality of deep learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/e7e8f8e5982b3298c8addedf6811d500-Abstract.html.

Allen-Zhu, Z. and Li, Y. What can resnet learn efficiently, going beyond kernels? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9015–9025, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5857d68cd9280bc98d079fa912fd6740-Abstract.html.

Allen-Zhu, Z. and Li, Y. Backward feature correction: How deep learning performs deep learning. *arXiv*, abs/2001.04413, 2020. URL https://arxiv.org/abs/2001.04413.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 6155–6166, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/62dad6e273d32235ae02b7d321578ee8-Abstract.html.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8139–8148, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/dbc4d84bfcfe2284ba11beffb853a8c4-Abstract.html.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019b.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 2933–2943, 2019.

Daniely, A. and Malach, E. Learning parities with neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/eaae5e04a259d09af85c108fe4d7dd0c-Abstract.html.

Daniely, A., Frostig, R., and Singer, Y. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2253–2261, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/abea47ba24142ed16b7d8fbf2c740e0d-Abstract.html.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685. PMLR, 09–15 Jun 2019a. URL http://proceedings.mlr.press/v97/du19c.html.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019b.

Geiger, M., Spigler, S., Jacot, A., and Wyart, M. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, nov 2020. doi: 10.1088/1742-5468/abc4de. URL https://doi.org/10.1088/1742-5468/abc4de.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9108–9118, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/

c133fb1bb634af68c5088f3438848bfd-Abstract.html.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. Linearized two-layers neural networks in high dimension. *arXiv*, abs/1904.12191, 2019b. URL http://arxiv.org/abs/1904.12191.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. When do neural networks outperform kernel methods? In *Advances in Neural Information Processing Systems*, volume 33, pp. 14820–14830. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/a9df2255ad642b923d95503b9a7958d8-Paper.pdf.

Jacot, A., Hongler, C., and Gabriel, F. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8580–8589, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html.

Kamath, P., Montasser, O., and Srebro, N. Approximate is good enough: Probabilistic variants of dimensional and margin complexity. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2236–2262. PMLR, 2020. URL http://proceedings.mlr.press/v125/kamath20b.html.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8570–8581, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0d1a9651497a38d8b1c3871c84528bd4-Abstract.html.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/54fe976ba170c19ebae453679b362263-Paper.pdf.

Li, Y., Ma, T., and Zhang, H. R. Learning over-parametrized two-layer neural networks beyond NTK. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2613–2682. PMLR, 2020. URL http://proceedings.mlr.press/v125/li20a.html.

Mei, S., Bai, Y., and Montanari, A. The landscape of empirical risk for nonconvex losses. *The Annals of Statistics*, 46(6A):2747 – 2774, 2018. doi: 10.1214/17-AOS1637. URL https://doi.org/10.1214/17-AOS1637.

Schölkopf, B. and Smola, A. J. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002. ISBN 9780262194754. URL https://www.worldcat.org/oclc/48970254.

Soltanolkotabi, M. Learning relus via gradient descent. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2007–2017, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/e034fb6b66aacc1d48f445ddfb08da98-Abstract.html.

Vardi, G., Yehudai, G., and Shamir, O. Learning a single neuron with bias using gradient descent, 2021. URL https://arxiv.org/abs/2106.01101.

Woodworth, B. E., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., and Srebro, N. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pp. 3635–3673. PMLR, 2020. URL http://proceedings.mlr.press/v125/woodworth20a.html.

Yehudai, G. and Shamir, O. On the power and limitations of random features for understanding neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 6594–6604, 2019.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Gradient descent optimizes over-parameterized deep relu networks. *Mach. Learn.*, 109(3):467–492, 2020. doi: 10.1007/s10994-019-05839-6. URL https://doi.org/10.1007/s10994-019-05839-6.