Quantifying the Benefit of Using Differentiable Learning over Tangent Kernels

Eran Malach Hebrew University of Jerusalem eran.malach@mail.huji.ac.il
Pritish Kamath Toyota Technological Institute at Chicago pritish@ttic.edu
Emmanuel Abbe EPFL emmanuel.abbe@epfl.ch
Nathan Srebro Toyota Technological Institute at Chicago nati@ttic.edu

Collaboration on the Theoretical Foundations of Deep Learning (deepfoundations.ai)

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 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 and Liang, 2018; Du et al., 2019b; Chizat et al., 2019b; 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).

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 and Shamir, 2019; Allen-Zhu and Li, 2019, 2020; Li et al., 2020; Daniely and 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 and 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?

<table>
<tr><th colspan="2"></th><th>NTK at same Initialization</th><th>NTK at alternate randomized Initialization</th><th>NTK of arbitrary model or even an arbitrary Kernel</th></tr>
<tr><td colspan="2">GD with unbiased initialization ($\forall_x f_{\theta_0}(x) = 0$) ensures small error</td><td>▸ NTK edge $\geq \text{poly}^{-1}$ (Thm. 1)
▸ NTK edge can be $< \text{poly}^{-1}$ while GD reaches 0 loss (Separation 1)</td><td colspan="2">Edge with any kernel can be $< \text{poly}^{-1}$ while GD reaches 0 loss (Separation 2)</td></tr>
<tr><td rowspan="2">GD with arbitrary init. ensures small error</td><td>Kernel (or alt init) can depend on input dist. $\mathcal{D}_\mathcal{X}$</td><td rowspan="2">NTK edge can be $= 0$ while GD reaches arb. low loss (Separation 3)</td><td>▸ NTK edge $\geq \text{poly}^{-1}$ (Thm. 2)
▸ NTK edge can be $< \text{poly}^{-1}$ while GD reaches 0 loss (Separation 2)</td><td>Edge can be $< \text{poly}^{-1}$ while GD reaches 0 loss (Separation 2)</td></tr>
<tr><td>Dist-indep kernels</td><td colspan="2">edge with any kernel can be $< \exp^{-1}$ while GD reaches arb. low loss (Separation 4)</td></tr>
</table>

Table 1: Our Results at a glance: What does learnability with Gradient Descent imply, and does not imply, on the edge over null prediction that the Neural Tangent Kernel (NTK), or some other kernel, must have? The results provide a complete picture (up to polynomial differences) of how large an edge is ensured in each scenario.

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 we can rely on *input distribution knowledge* for the initialization.

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 1 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 2 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 that we uncover is mapped in Table 1.

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 we refer to the improvement $0.5 - \mathcal{L}_{\mathcal{D}}(f)$ as the "edge" of the predictor f. That is, 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[1] $C_f^2 := \sup_{\theta,x}(\|\nabla_\theta f_\theta(x)\|_2^2 + f_\theta^2(x))$ on the norm of the gradients and function values[2].

For a differentiable model $f_\theta(x)$ and an *initialization* $\theta_0 \in \mathbb{R}^p$, gradient accuracy τ, and stepsize[3] η, **τ-approximate gradient descent training** is given by iterates of the form:

$$\theta^{(0)} \leftarrow \theta_0$$
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta g_t \tag{1}$$
$$\text{for } \|g_t - \nabla_\theta \mathcal{L}_{\mathcal{D}}(f_{\theta^{(t)}})\|_2 \le \tau \tag{2}$$

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.[4] 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 $\mathcal{L}_{\mathcal{D}}(f_{\theta^{(T)}}) \le \varepsilon$.

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

$$f_\theta(x) \approx f_{\theta_*}(x) + (\theta - \theta_*)\nabla_\theta f_{\theta_*}(x) \tag{3}$$
$$= \langle [1, \theta - \theta_*], \phi_{\theta_*}(x) \rangle \tag{4}$$
$$\text{where } \phi_{\theta_*}(x) = [f_{\theta_*}(x), \nabla_\theta f_{\theta_*}(x)] \in \mathbb{R}^{p+1}, \tag{5}$$

[1] 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.

[2] ReLU networks do not fit in this framework because (1) they are not differentiable; and (2) their gradients are not bounded. The first issue is not significant, since we never rely on a bound on the second derivatives of the model (only the loss), and so we can either approximate the ReLU with an arbitrarily sharp approximation, or use some relaxed notion of a "gradient", as is actually done informally when discussing gradients of ReLU networks. Our results do rely on a bound on the gradient norms, which for ReLU networks depends on the scale of the weights θ (formally, it is sufficient to consider the model restricted to some large enough norm ball of weights which we will never exit). By truncating the model outside the scale of weights we would need to consider (in the analysis only), it is sufficient to take the supremum in the definition of C_f only over weight settings explored by gradient descent, and so for ReLU networks C_f would scale with this scale of θ.

[3] The stepsize doesn't play an important role in our analysis. We can also allow variable or adaptive stepsize sequences—for simplicity of presentation we stick with a fixed stepsize.

[4] 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 and Sandon (2020a,b). We focus here on the approximate setting that better reflects the noisier regime; see discussion in Abbe and Sandon (2020a) for parities.

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

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

In some regimes or model limits (e.g. Daniely et al., 2016; Jacot et al., 2018; Chizat et al., 2019a), 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):

$$\begin{aligned} \mathcal{F}(K,B) :=& \quad \{x \mapsto \langle w, \phi(x)\rangle : \|w\|_2 \cdot R \leq B\} && (6)\\ =& \quad \{f : \mathcal{X} \to \mathbb{R} : \|f\|_K \cdot R \leq B\} && (7)\\ \text{where } R :=& \sup_x \|\phi(x)\|_2 \;=\; \sup_x \sqrt{K(x,x)} \end{aligned}$$

where $K(x,x') = \langle \phi(x), \phi(x')\rangle$ and $\|f\|_K$ is the K-RKHS-norm[5] 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.

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 $\mathcal{D}_\mathcal{X}$.

Remark. Typically, we would have a probabilistic quantifier (e.g. in expectation, or with high probability) over sampling the training set, but we define differentiable learning, and learning using a kernel, without any probabilistic quantifier: For differentiable learning, we required that for any $\mathcal{D} \in \mathcal{P}$, gradient descent yields error at most ε using any sequence of gradient estimates satisfying (2) (this happens with high probability if we use $m = O(B^2/\tau^2)$ to obtain the gradient estimates, but we do not get into this in our results). For kernel learning, we require that for any $\mathcal{D} \in \mathcal{P}$ there exists a predictor in $\mathcal{F}(K,B)$ with error at most ε, as these are the predictors that can be learned (with high probability) using the kernel method (but we do not get into the exact learning procedure).

[5]The direct definition (6) is sufficient for our purposes, and so we refrain from getting into the definition of an RKHS and the RKHS norm. See, e.g. Schölkopf and 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.

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 and Shamir, 2019; Ghorbani et al., 2019a,b; Allen-Zhu and Li, 2019, 2020; Li et al., 2020; Daniely and Malach, 2020). In Section 6 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 Section 6 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 differentiable 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[6] $x \sim \mathcal{D}_{\mathcal{X}} := (1-\alpha)\mathcal{D}_0 + \alpha\mathcal{D}_1$.
- ▷ 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 2). 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.

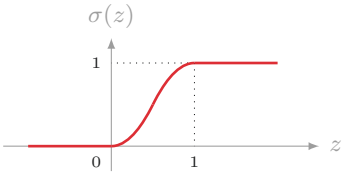
Formally, we construct a differentiable model $f_\theta(x)$ with $\theta \in \mathbb{R}^n$ (i.e. size $p = n$) that behaves as follows

$$\forall \theta \in \left(\left[-\tfrac{2}{n}, \tfrac{2}{n}\right] \cup \left[\tfrac{3}{n}, \tfrac{5}{n}\right]\right)^n \quad : \quad f_\theta(x) \begin{cases} \approx 2\langle \theta, x \rangle & \text{if } \theta \approx 0 \\ = \prod_{i\,:\,\theta_i \ge \frac{3}{n}} x_i & \text{if } \exists i \,:\, \theta_i \ge \frac{3}{n} \end{cases}, \tag{8}$$

[6] The uniform component in the mixture is used to facilitate the establishment of a lower bound for kernels by directly relying on Lemma 5 in Kamath et al. (2020). One could also directly lower bound the kernel error of a biased input distribution without the uniform component, by going beyond a standard application of Kamath et al. (2020); see Remark 4.

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 3.1, 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 $\theta_i^{(1)} \geq \frac{3}{n}$ for $i \in I$, thus yielding the correct parity. In what follows 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 2, is defined below (where $\theta \circ x := (\theta_1 x_1, \ldots, \theta_n x_n)$).

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

where $\sigma_{a,b}^{c,d}(z) := c + \sigma\left(\frac{z-a}{b-a}\right)(d - c)$, defined for all $a < b$ and c, d, satisfies (i) $\sigma_{a,b}^{c,d}(z) = c$ for every $z \leq a$, (ii) $\sigma_{a,b}^{c,d}(z) = d$ for every $z \geq b$ and (iii) $\left|\frac{d}{dz}\sigma_{a,b}^{c,d}(z)\right| \leq \frac{2|d-c|}{b-a}$. We visualize ξ, which is a (coordinate-wise) soft implementation of the $\mathrm{sign}(\cdot)$ function, in Figure 1.

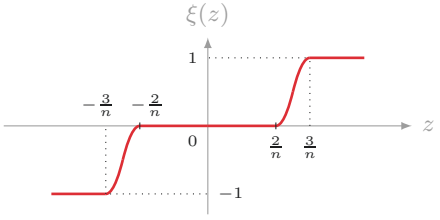


Figure 1: Visualization of $\xi(z)_i$.

The intuition for $\mathcal{G}$ is as follows. In the relevant regime of $\theta \in \left([-\frac{2}{n}, \frac{2}{n}] \cup [\frac{3}{n}, \frac{5}{n}]\right)^n$ 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 $S(s) = \mathbb{1}\{s \neq 0\}$ and $H(s) = \prod_{i:s_i \neq 0} s_i$. 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 $\mathcal{G}(\theta \circ x) = \prod_{i:\theta_i \geq \frac{3}{n}} x_i - \langle \theta, x \rangle$ 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 $|z| \geq 1$.

Furthermore, we show how to implement S, H (and hence $\mathcal{G}$) as a neural network using the activation σ. This is based on observing that we can compute squares in a bounded range by exploiting the quadratic part of the nonlinearity, i.e., $z^2 = 2r^2\left(\sigma(z/2r) + \sigma(-z/2r)\right)$ for $z \in [-r, +r]$. The n-term products in (11) and (12) can be implemented with subnetworks of depth $O(\log n)$ and width $O(n)$ that recursively computes products of pairs, using $uv = \frac{1}{2}((u + v)^2 - u^2 - v^2)$; if $u, v \in [-1, 1]$, we only need to compute squares in the range $[-2, 2]$. There is a slight issue in computing the final product because $H(\xi(\theta \circ x)) - \langle \theta, x \rangle$ can be unbounded. However, $\langle \theta, x \rangle \leq 5$ in the relevant regime of θ and so we can correctly compute the final product in this regime; the product is not implemented correctly outside the relevant regime, but that doesn't affect the learning algorithm. The overall network thus has depth $O(\log n)$ and $O(n)$ edges.

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

The following Claim (proved in Appendix B.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 3.1. *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*

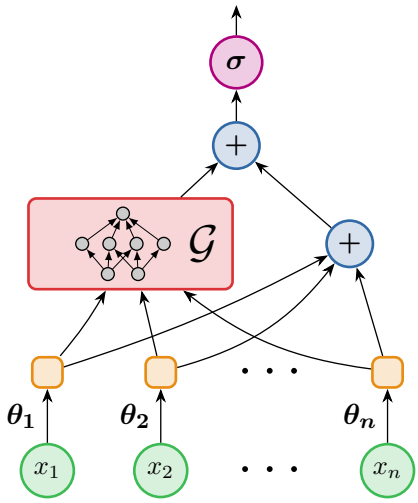


Figure 2: Schematic diagram for the construction of f_θ, as used in Claim 3.1, with only trainable parameters being $\theta_1, \theta_2, \ldots, \theta_n$. The sub-network $\mathcal{G}$ is a fixed module implementing the "selected-parity" function.

$\tau \leq \alpha/2^k$, *step size* $\eta = 2^k/(\alpha n)$ *and* $T = 1$ *step ensures* $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(T)}}) = 0$. *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 C.1.1), establishes that the tangent kernel at initialization only has a small edge over the null prediction.

Claim 3.2. *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 3.1. 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), such that*

- ▷ *[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 3.1 and 3.2 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[7]. The following claim shows that this is not the case (proof in Appendix C.2.1).

[7]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.

Claim 3.3. *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:

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 $B_n = \operatorname{poly}(n)$.*

Proof. Consider $\mathcal{P}_n = \mathcal{P}^{\mathsf{bsp}}[n, k = \log_2 n, \alpha = \gamma_n = 1/\operatorname{poly}(n)]$. Learnability using GD follows from Claim 3.1, noting that $\frac{\tau}{C_f} = \frac{\alpha}{n 2^k} = \frac{\gamma_n}{n^2}$ is inverse-polynomial in n. Since $\binom{n}{\log_2 n} = n^{\Omega(\log n)} \gg \operatorname{poly}(n)$, if the kernel dimension $p(n)$ (or similarly, the norm B_n) is polynomial in n, then $\frac{p}{2\binom{n}{k}} = o(1)$, and so for large enough n, the error lower bound in Claim 3.3 would be $> \frac{1}{2} - \gamma_n$. □

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 3 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 3.2 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 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:

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

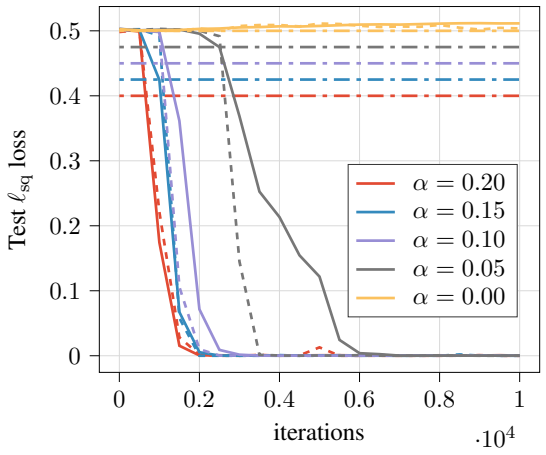


Figure 3: Two-layer fully-connected ReLU network (128 hidden units) training on data sampled from $\mathcal{D}_I$ with $n = 128$, $k = 7$ for different values of α, trained using Adam optimizer with learning-rate of 0.01. Solid lines show the test performance at each iteration averaged over 20 runs. 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 3.3) on best accuracy attainable by any kernel method with feature map with as many number of parameters as the NTK.

Following Chizat et al. (2019a), 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. (2019a), 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 3.1 of Section 3 also satisfies (14).

In Theorem 1 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 by Theorem 1, the Theorem would still ensure the tangent kernel has an edge over null prediction.

Theorem 1. *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*

$$\mathcal{L}_{\mathcal{D}}(h) \leq \mathcal{L}_{\mathcal{D}}(0) - \frac{\tau^2}{2|\ell''|C_f^2}. \tag{16}$$

Proof. 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.

Let us consider first the general (not unbiased) case, and establish (15). If $\mathcal{L}_{\mathcal{D}}(f_{\theta_0}) \leq \varepsilon$, then $f_{\theta_0} \in \mathsf{NTK}^f_{\theta_0}(C_f)$ already satisfies (15). Otherwise, we must have that $\|\nabla_\theta \mathcal{L}_{\mathcal{D}}(f_{\theta_0})\|_2 \geq \tau$, since otherwise g_t can be 0 at every iteration of gradient-descent, forcing gradient-descent to return f_{θ_0}. Denote $h_{\boldsymbol{w}}(x) = f_{\theta_0}(x) + \langle \boldsymbol{w}, \nabla_\theta f_{\theta_0}(x) \rangle$, for $\boldsymbol{w} \in \mathbb{R}^p$, noting that $h_{\boldsymbol{w}} \in \mathsf{NTK}^f_{\theta_0}((1 + \|w\|)C_f)$ and $h_0 = f_{\theta_0}$. Observe that:

$$\nabla_\theta \mathcal{L}_{\mathcal{D}}(f_{\theta_0}) \;=\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\ell'(h_0(x), y) \nabla_\theta f_{\theta_0}(x)\right] \;=\; \nabla_{\boldsymbol{w}} \mathcal{L}(h_0) \tag{17}$$

For any $\alpha > 0$, denoting $\boldsymbol{w}_* = -\alpha \frac{\nabla_{\boldsymbol{w}} \mathcal{L}(h_0)}{\|\nabla_{\boldsymbol{w}} \mathcal{L}(h_0))\|}$, we have:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}_*) \;&=\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \ell(\langle \boldsymbol{w}_*, \nabla_\theta f_{\theta_0}(x) \rangle, y) \\
&\leq\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\ell(h_0(x), y) \;+\; \ell'(h_0(x), y) \langle \boldsymbol{w}_*, \nabla_\theta f_{\theta_0}(x) \rangle \;+\; \tfrac{|\ell''|}{2} \langle \boldsymbol{w}_*, \nabla_\theta f_{\theta_0}(x) \rangle^2 \right] \\
&\leq\; \mathcal{L}(h_0) + \langle \boldsymbol{w}_*, \nabla_{\boldsymbol{w}} \mathcal{L}(h_0) \rangle + \frac{|\ell''|}{2} \|\boldsymbol{w}_*\|^2 C_f^2 \\
&=\; \mathcal{L}(h_0) - \alpha \|\nabla_{\boldsymbol{w}} \mathcal{L}(h_0)\| + \frac{\alpha^2 |\ell''| C_f^2}{2} \\
&<\; \mathcal{L}(h_0) - \alpha\tau + \frac{\alpha^2 |\ell''| C_f^2}{2}.
\end{aligned}$$

Choosing $\alpha = \frac{\tau}{|\ell''| C_f^2}$ completes the proof of (15). For unbiased initialization, if $\varepsilon < \mathcal{L}_{\mathcal{D}}(0) = \mathcal{L}_{\mathcal{D}}(f_{\theta_0})$ then again we must have $\|\nabla_\theta \mathcal{L}_{\mathcal{D}}(f_{\theta_0})\|_2 \geq \tau$ and the proof proceeds as above, noting that $f_{\theta_0} = h_0 = 0$. □

Separation 1 establishes that Theorem 1 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 1 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 1 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 2 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 2. *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*

$$\mathop{\mathbb{E}}_{\theta\sim\mathcal{W}} \inf_{h\in\mathsf{NTK}^f_\theta(B)} \mathcal{L}_{\mathcal{D}}(h) < \mathcal{L}_{\mathcal{D}}(0) - \gamma' \tag{18}$$

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

$$\inf_{h\in\mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}}(h) \le \mathcal{L}_{\mathcal{D}}(0) - \gamma'. \tag{19}$$

Proof. We first define a sequence of iterates $\tilde{\theta}_t$ which corresponds to gradient descent descent on $\mathcal{L}_{\tilde{\mathcal{D}}}(f_\theta)$, where $\tilde{\mathcal{D}}$ is defined as a distribution with marginal $\mathcal{D}_{\mathcal{X}}$ and $y = 0$ with probability one. These iterates are given by:

$$\tilde{\theta}_0 = \theta_0 \tag{20}$$

$$\tilde{\theta}_{t+1} = \tilde{\theta}_t - \eta \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}} \left[f_{\tilde{\theta}_t}(x) \cdot \nabla_{\theta_t} f_{\tilde{\theta}_t}(x) \right] \tag{21}$$

Let $\mathcal{W}$ be the uniform distribution over $\{\tilde{\theta}_0, \ldots, \tilde{\theta}_T\}$. At a high level, if the Tangent Kernel at all of these iterates does not have an edge, then gradient descent will not deviate from them, in the same way that in Theorem 1 the Tangent Kernel had to have an edge in order for gradient descent to move. But then gradient descent leads to $f_{\tilde{\theta}_T} \in \mathsf{NTK}^f_{\tilde{\theta}_t}$, and so any edge gradient descent has is also obtained in this Tangent Kernel.

In a sense, we are decomposing the gradient into two components: the gradient if all labels were zero, and the deviation from this gradient due to the labels not being zero. The second component is non-zero only when the tangent kernel has an edge. If the initialization is unbiased, the first component is zero, and so if the tangent kernel doesn't have an edge, gradient descent will not move at all. But if the initialization is not unbiased, the first component (corresponding to zero labels) might be non-zero even if the tangent kernel does not have an edge, and so gradient descent can explore additional iterates $\tilde{\theta}_t$, and any one of these having the edge could be sufficient for gradient descent succeeding. The key is that this sequence of iterates depends only on the input distribution $\mathcal{D}_{\mathcal{X}}$, but not on the labels (since it is defined as the behaviour on zero labels).

Denote $v^{(t)} := \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y \cdot \nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right]$, and let $\boldsymbol{w}^{(t)} = -\alpha v^{(t)} / \left\|v^{(t)}\right\|$, with α to be set later s.t. $(x \mapsto \left\langle \boldsymbol{w}^{(t)}, \nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\rangle) \in \mathsf{NTK}^f_{\tilde{\theta}^{(t)}}(B)$. Assume for contradiction that (18) does not hold (i.e. the Tangent Kernels do not have an edge on average), and so:

$$\mathop{\mathbb{E}}_{t\sim[0..T]} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\ell\left(\left\langle \boldsymbol{w}^{(t)}, \nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\rangle, y\right)\right] \;\ge\; \mathop{\mathbb{E}}_{\theta\sim\mathcal{W}} \inf_{h\in\mathsf{NTK}^f_\theta(B)} \mathcal{L}_{\mathcal{D}}(h) \;\ge\; \mathcal{L}_{\mathcal{D}}(0) - \gamma' \tag{22}$$

On the other hand, we have for every t:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\left\langle \boldsymbol{w}^{(t)}, \nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\rangle, y\right)\right] \\
&= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{1}{2}y^2 \;-\; y\left\langle \boldsymbol{w}^{(t)}, \nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\rangle \;+\; \frac{1}{2}\left\langle \boldsymbol{w}^{(t)}, \nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\rangle^2\right] \\
&\leq \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\frac{1}{2}y^2 - \left\langle \boldsymbol{w}^{(t)}, \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} y\nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\rangle \;+\; \frac{1}{2}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\|\boldsymbol{w}^{(t)}\|^2\left\|\nabla_\theta f_{\tilde{\theta}^{(t)}}(x)\right\|^2 \\
&\leq \mathcal{L}_\mathcal{D}(0) - \alpha\left\|v^{(t)}\right\| + \frac{\alpha^2}{2}C_f^2
\end{aligned}$$

Using the above, taking $\alpha = \frac{\sqrt{2\gamma'}}{C_f}$ we get that for every t:

$$\|v^{(t)}\| \leq (T+1)\mathop{\mathbb{E}}_{t\sim[0..T]}\|v^{(t)}\| \leq (T+1)\sqrt{2\gamma'}C_f \tag{23}$$

We will now argue inductively that $\tilde{\theta}^{(t)}$ is a valid choice for the iterate $\theta^{(t)}$, i.e. satisfying (1). For iteration $t = 0$, $\tilde{\theta}^{(0)} = \theta^{(0)} = \theta_0$. If up to iteration t we can have $\tilde{\theta}^{(t)} = \theta^{(t)}$ then:

$$\begin{aligned}
\nabla_\theta\left[\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\ell(f_{\theta^{(t)}}(x), y)\right] &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left(f_{\theta^{(t)}}(x) - y\right)\cdot\nabla_\theta f_{\theta^{(t)}}(x) &(24)\\
&= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left(f_{\tilde{\theta}^{(t)}}(x) - y\right)\cdot\nabla_\theta f_{\tilde{\theta}^{(t)}}(x) &(25)\\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}} f_{\tilde{\theta}^{(t)}}(x)\cdot\nabla_\theta f_{\tilde{\theta}^{(t)}}(x) - v^{(t)} &(26)
\end{aligned}$$

Now, from (23), and since $\tau \geq (T+1)\sqrt{2\gamma'}C_f$, we have that $g_t = \mathbb{E}_{x\sim\mathcal{D}_\mathcal{X}} f_{\theta^{(t)}}(x)\nabla_\theta f_{\theta^{(t)}}(x)$ satisfies (2), and so $\theta^{(t+1)} = \tilde{\theta}^{(t+1)}$ satisfies (1). And by induction, GD can return $f_{\theta^{(T)}} = f_{\tilde{\theta}^{(T)}}$. Since $f_{\tilde{\theta}^{(T)}} \in \mathsf{NTK}^f_{\tilde{\theta}^{(T)}}(B)$ we get:

$$\begin{aligned}
\mathcal{L}_\mathcal{D}(0) - \mathcal{L}_\mathcal{D}(f_{\theta^{(T)}}) &= \mathcal{L}_\mathcal{D}(0) - \mathcal{L}_\mathcal{D}(f_{\tilde{\theta}^{(T)}}) \\
&\leq \mathcal{L}_\mathcal{D}(0) - \inf_{h\in\mathsf{NTK}^f_{\tilde{\theta}^{(T)}}(B)}\mathcal{L}_\mathcal{D}(h) \\
&\leq (T+1)\mathop{\mathbb{E}}_{\theta\sim\mathcal{W}(\mathcal{D}_\mathcal{X})}\left[\mathcal{L}_\mathcal{D}(0) - \inf_{h\in\mathsf{NTK}^f_\theta(B)}\mathcal{L}_\mathcal{D}(h)\right] \\
&\leq \gamma'(T+1) \;\leq\; \gamma\,.
\end{aligned}$$

This leads to a contradiction, thereby completing the proof of Theorem 2. For the average Kernel, note that it corresponds to a $(T+1)(p+1)$ dimensional feature space which is the concatenation of the feature spaces of each $\mathsf{NTK}^f_{\tilde{\theta}^{(t)}}$, and consider a predictor which is the concatenation of the best predictors for each kernel scaled by $1/(T+1)$. □

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 1 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 2 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 $|I| \geq 2$. 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

$$\forall \theta \in \left(\left[0, \tfrac{2}{n}\right] \cup \left[\tfrac{3}{n}, \tfrac{5}{n}\right]\right)^n \quad : \quad f_\theta(x) \begin{cases} \approx -1 + 2\left\langle \theta, x + \frac{5}{3}\alpha \mathbf{1} \right\rangle & \text{if } \theta \approx 0 \\ = \prod_{i:\theta_i \geq \frac{3}{n}} x_i & \text{if } \exists i \,:\, \theta_i \geq \frac{3}{n} \end{cases}, \tag{27}$$

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, (27) 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 (27), and the model will output the desired parity[8]. 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 2).

As with the model of Section 3, we can implement (27) as a continuous differentiable model with scale $C_f = O(n)$, through a slight modification of the architecture described in Equations (9)-(13):

$$f_\theta(x) \;:=\; \sigma_{0,\frac{2}{n}}^{-1,0}\left(\langle \theta, \mathbf{1}\rangle\right) + \sigma_{-1,1}^{-1,1}\left(\left\langle \theta, x + \tfrac{5}{3}\alpha\mathbf{1}\right\rangle + \mathcal{G}(\theta, x)\right) \tag{28}$$

$$\mathcal{G}(\theta,x) \;:=\; S(\xi(\theta \circ x)) \cdot \left(H(\xi(\theta \circ x)) - \left\langle \theta, x + \tfrac{5}{3}\alpha\mathbf{1}\right\rangle\right) \tag{29}$$

where S, H and ξ are as defined in (11), (12) and (13). The first term in (28) is -1 at $\theta^{(0)} = 0$ and satisfies $\nabla_\theta\, \sigma_{0,\frac{2}{n}}^{-1,0}\left(\left\langle \theta^{(0)}, \mathbf{1}\right\rangle\right) = 0$. 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 (27) holds and that $C_f^2 \leq O(n^2)$. Furthermore, we can implement f_θ using a similar feed-forward network of depth $O(\log n)$ and $O(n)$ edges.

[8]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 (27) when implementing the model

We prove the following claims in Appendix B.2 and Appendix C.1.2 respectively (cf. Claims 3.1 and 3.2).

Claim 5.1. *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 $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(T)}}) \leq \alpha$.*

Claim 5.2. *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 $n \geq 2$, $\alpha \in (0,1)$, 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 $B \geq 0$.*

We already see that in contrast to a situation where the initialization is unbiased, and so Theorem 1 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 1:

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

- ▷ *[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 5.1 and 5.2 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 C.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 (18) of Theorem 2, we state the Claim also for learning using a random kernel, i.e. in expectation over an arbitrary distribution of kernels.

Claim 5.3. *For all $\alpha \in (0,1), p, B, n$, and any randomized kernel K with $\mathrm{rank}(K) = p$ 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*

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \;\geq\; \mathcal{L}_{\mathcal{D}_I}(0) - \min\left\{ \frac{p}{2|\mathcal{P}^{\mathsf{lp}}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}^{\mathsf{lp}}|^{1/3}}\right)\right\}.$$

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 2, 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 2.

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 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 5.1 and 5.3 to $\mathcal{P}_n = \mathcal{P}^{\mathsf{lp}}[n, \alpha = \varepsilon_n]$. □

Since we are working over a domain $\mathcal{X} = \{-1, +1\}^n$ or 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 5.3 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).

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.*

Proof. Consider $\mathcal{P}_n = \mathcal{P}^{\mathsf{lp}}[n, \alpha = \varepsilon_n]$. Claim 5.1 ensures learnability with a biased initialization. Suppose $\mathcal{P}_n$ was also learnable by τ-approximate gradient descent with unbiased initialization, using a model f of size $p_n = \mathrm{poly}(n)$, scale C_f satisfying $\tau/C_f = 1/\mathrm{poly}(n)$, to within any error better than 0.5, then Theorem 1 would imply the tangent kernel at initialization would ensure error $\leq \frac{1}{2} - \frac{1}{\mathrm{poly}(n)}$. This is a fixed kernel that depends only on the model f_θ, and corresponds to a feature map of dimension $p_n + 1$. Since $p_n = \mathrm{poly}(n)$, this would contradict Claim 5.3. □

6 Survey of Separation Results

In Sections 3 and 5 we described learning problems for which gradient descent learning succeeds, but where the NTK, or indeed any other (reasonably sized) kernel, has either a very small edge, or even no edge at all. Our emphasis was on establishing not only that the NTK or kernel methods get worse error than gradient descent (as in previous separation results), but on bounding how close to the null prediction they must be (i.e. how small an edge they must have). Here we survey such prior separation results, understanding the relationships between them, and how they relate to the new separations from this paper.

Survey Table. Table 2 summarizes prior results showing separations between the error that can be ensured using gradient descent versus using kernel methods, as well as our Separations 1, 2, 3 and 4. All problems are over $\mathcal{X} = \mathbb{R}^n$ or $\{\pm 1\}^n$ with scalar labels ($\mathcal{Y} = \mathbb{R}$ or $\{\pm 1\}$) and w.r.t. the squared loss, except Allen-Zhu and Li (2019) which uses multiple outputs (y is a vector), and Daniely and Malach (2020) which is w.r.t. the hinge loss.

The middle set of green columns indicate what error can be ensured by running the indicated type of gradient descent variant on the indicated type of model. An error of "$\to 0$" indicates that the learning problem only depends on n, and that for every n, gradient descent on a fixed model (that depends on n but not ε) can make the error ε arbitrarily small using $\mathrm{poly}(n/\varepsilon)$ samples and/or steps. An error of "$\leq \varepsilon$" indicates that the learning problem also depends on ε: for all n and $\varepsilon > 0$, there exists a learning problem on which gradient descent ensures error $\leq \varepsilon$ (but

Table 2: Prior and new separation results between differentiable learning and kernel methods

Learning Problem		**Learning with Gradient Descent**				**Kernel Lower Bound**		
Labels	**Inputs**	**Model**	**Type of GD**	**Error**	**I**	**Kernel**	**Error**	$\mathcal{D}$
Single ReLU $y = [\langle w, x\rangle + b]_+$	Gaussian	Single ReLU Zero Init	Batch GD $\mathrm{poly}(n \log \frac{1}{\varepsilon})$ steps $\mathrm{poly}(\frac{n}{\varepsilon})$ samples	$\varepsilon \to 0$ when $b = 0$	0	Yehudai and Shamir (2019)		
						$\dim \le 2^{O(n)}$	$\Omega(1/n^8)$: see (1)	y
						Ghorbani et al. (2019b)		
						rotationally inv. #samples $\le n^c$	$\ge \varepsilon'(c) > 0$ even if $b = 0$	–
						NTK of depth-2 ReLU Net with $\le n^c$ units		
Quadratic function	Gaussian	Depth-2 Net quad. activations # units = $\rho \cdot n$ for $\rho < 1$	Gradient Flow on Population	$\to \varepsilon(\rho)$ see (2)	$\approx$	Ghorbani et al. (2019a)		–
Binary classification between two zero mean Gaussians (diff. cov.)						**NTK** at Init	$\to \varepsilon'(\rho) > \varepsilon(\rho)$ see (2)	
Depth-2 Net with restricted weights abs. value activations	Gaussian	Depth-2 ReLU $\mathrm{poly}(n)$ units Random Init	Truncated(3) GD $\mathrm{poly}(n)$ steps $\mathrm{poly}(n)$ samples	$< \frac{1}{n^{1+\delta}}$ for some $\delta > 0$	×	Li et al. (2020)		
						$\dim \le \mathrm{poly}(n)$	$\Omega(\frac{1}{n}) > \Omega(\varepsilon^{1-\frac{\delta}{2}})$	y
$y \in \mathbb{R}^{\log n}$ $y = x_S + \varepsilon^{\frac{1}{4}} \chi_S(x)\mathbf{1}$	Uniform	Depth-2 ResNet $\mathrm{poly}(n/\varepsilon)$ units Random Init	SGD $\mathrm{poly}(n/\varepsilon)$ steps	$< \varepsilon$	×	Allen-Zhu and Li (2019)		
						$\dim \le \mathrm{poly}(n)$	$> \sqrt{\varepsilon}$	y
ResNet-type model of depth $\Theta(\mathrm{loglog}\, n)$ with restricted weights	Gaussian	Same as teacher $\mathrm{poly}(n)$ units Zero Init	Regularized(4) SGD $\mathrm{poly}(n/\varepsilon)$ steps	$\varepsilon \to 0$	0	Allen-Zhu and Li (2020)		
						$\dim \le \mathrm{poly}(n)$	$\Omega(\frac{1}{n^{0.01}})$	y
Sparse parities with input distribution leaking the support		Depth-2 ReLU6 $\mathrm{poly}(n)$ units Unbiased(5) Init	Regularized Population GD $\mathrm{poly}(n)$ steps	$\frac{1}{\mathrm{poly}(n)}$ hinge loss	0	Daniely and Malach (2020)		
						$\dim \le \mathrm{poly}(n)$ norm $\le \mathrm{poly}(n)$	$\Omega(1)$ hinge loss	x
Our Results : Separations 1, 2, 3, 4								
XOR of two bits	Mix of uniform & biased bits	Specialized	τ-approximate Grad Descent (e.g. GD with $\Omega(1/\tau^2)$ samples) 1 step	0	0	**NTK** at Init	$\ge \mathcal{L}(0) - \tau$ arbitrarily small τ	–
						$\dim \le \mathrm{poly}(n)$	$\mathcal{L}(0) - n^{-\Omega(1)}$	y
$\log n$-sparse parities								
Mix of parities and input distribution leaking the support (mixture ratio depends on ε)		Specialized		$\le \varepsilon$	×	**NTK** at Init	$\mathcal{L}(0)$	–
						$\dim \le \mathrm{poly}(n)$	$\mathcal{L}(0) - 2^{-\Omega(n)}$	x

See Section 6 for table description and notation, and Appendix A for further details.
Column **I**: 0 indicates unbiased initialization, $\approx$ almost unbiased, $\times$ not unbiased.
Column $\mathcal{D}$: "–" indicates lower bound for specific source distribution, "y" for learning problem with fixed input distribution $\mathcal{D}_X$, "x" for learning problem with variable input distributions.

(1) There is a discrepancy between the lower bound of Yehudai and Shamir (2019) and the upper bounds on learning a Single ReLU with gradient descent (e.g. Soltanolkotabi, 2017; Mei et al., 2018) in that the upper bounds are only without a bias (i.e. when $b = 0$) but the lower bound relies crucially on the bias b. Yehudai and Shamir thus do not establish a separation between differentiable learning and kernel methods. See Appendix A for discussion on possible remedies.

(2) The gradient descent error $\varepsilon(\rho)$ and kernel error $\varepsilon'(\rho)$ depend on ρ and on the specific source distribution (i.e. the quadratic target or the covariances of the classes), or rather sequence of source distributions as $n \to \infty$. But Ghorbani et al. (2019a) show that for non-degenerate distribution sequences, and any $\rho < 1$, the limit $\varepsilon(\rho)$ of the gradient descent error (as $n \to \infty$) will be strictly lower than a lower bound $\varepsilon'(\rho)$ on the error of the NTK.

(3) Li et al. (2020) analyze a form of GD where large coordinates of the gradient are set to zeros.

(4) Allen-Zhu and Li (2020) use non-standard but simple regularization.

(5) Daniely and Malach (2020) use random initialization, but units are paired with complimentary initialization, as suggested by Chizat et al. (2019a), so as to ensure an unbiased initialization.

kernel methods incur larger error). The annotations $0/$ $\approx$ $/\times$ under column "I" indicates: $\mathbf{0}$ if the initialization used is unbiased, or could be easily made unbiased; $\approx$ if it is nearly unbiased (error at initialization close to null prediction); or $\times$ if the initialization is far from null.

The right set of pink columns shows the lower bound on the error for the kernel, or class of kernels indicated, or under other restrictions ("dim" is the number of features or rank of the kernel, "norm $\leq B$" means restricting to $\mathcal{F}(K, B)$ as in (6), NTK is the tangent kernel of the model used for gradient descent unless otherwise specified). In all cases, the error is normalized so that the error of null perdition is $\mathcal{L}(0) = 1/2$. For our separations, the error is given in terms of the edge (if any) over null prediction. In all prior separations, the error lower bound is bounded away from the null prediction (the edge is at least 0.1), and the actual error lower bound is indicated. The notation $\geq \varepsilon'(c) > 0$ and $\geq \varepsilon'(\rho) > \varepsilon(\rho)$ indicates that for any choice of c or ρ, there is some $\varepsilon'(c) > 0$ or $\varepsilon'(\rho) > \varepsilon(\rho)$ which lower bounds the error.

The last column indicates the nature of the learning problem used to establish the lower bound, and thus whether the separation is distribution dependent or independent (in the sense introduced in Section 2): "—" indicates that the separation is for a specific source distribution (or sequence of source distribution with increasing input dimension n). This is the strongest form of separation, implying also distribution dependent separation, and is only possible by restricting to a specific kernel or kernel class. The notation "y" indicates the lower bound is for a learning problem $\mathcal{P}$ where the input distribution $\mathcal{D}_{\mathcal{X}}$ is fixed, and the source distributions in $\mathcal{P}$ vary only in the labeling function $y(x)$ (which is deterministic in all such problems considered in the table). This yields a "distribution dependent" separation (i.e. a separation that holds even if the kernel is allowed to depend on hte input distribution). The notation "$\boldsymbol{x}$" indicates that the lower bound is for a learning problem $\mathcal{P}$ where different source distributions $\mathcal{D} \in \mathcal{P}$ have varying input distributions $\mathcal{D}_{\mathcal{X}}$, but we seek a kernel that would work well for all distributions in the learning problem. This yields a weaker "distribution independent" separation, as it only implies a separation if the kernel choice is not allowed to depend on the input distribution $\mathcal{D}_{\mathcal{X}}$. It should be noted that these differences are due to differences in how the *lower bound* is established, and not differences in limitations or generality of the gradient descent guarantees (in all settings considered, the gradient descent guarantees are under severe restrictions on the input distribution, but the model used for training is independent of the specific input distribution meeting these restrictions).

Discussion. As can be seen from Table 2, in all the prior separation results, the lower bound on the error of kernel methods is bounded away from $\mathcal{L}(0)$, i.e. the upper bound on the edge is $O(1)$. In fact, in all prior separations except for Daniely and Malach (2020), the kernel error goes to zero when the error of gradient descent goes to zero, it just goes to zero slower (Daniely and Malach establish a lower bound on the error that is bounded away from zero, but it still corresponds to a constant edge). In contrast, we exhibit a situation where the NTK, or any kernel method, has only vanishing (or zero) edge, and study how quickly this edge goes to zero. In order to obtain a separation where the edge of the NTK at initialization is zero, and the edge of any kernel is exponentially small, we construct a learning problem where the input distribution is not fixed, and where gradient descent learning is possible only with a biased initialization—this is necessary otherwise Theorems 1 and 2 would tell us the edge must be at least polynomial. It is interesting to note that none of the prior constructions are of this nature: the only prior construction that relies on variable input distributions is that of Daniely and Malach (2020), but they use an unbiased initialization.

The different separation results differ in the type of kernels for which they establish lower bounds. Many of the separation results (including our Separations 2 and 4) establish lower bounds for any kernel corresponding to a poly-dimensional feature space, and thus for the NTK of any poly-sized model. When considering the square loss, such lower bounds also imply lower bounds for any kernel (of any dimensionality) using poly-norm predictors, as in our Claims 3.3 and 5.3 (also see Lemma 5 in Kamath et al., 2020). Since Daniely and Malach (2020) considered the hinge loss, they required that *both* the dimension *and* the norm be polynomial (see Kamath et al. (2020) for a discussion on kernel lower bounds for the squared norm versus the hinge loss. In yet unpublished followup work, we obtain lower bounds on the hinge loss also without a norm bound).

As discussed in Section 2, in order to establish lower bounds with respect to any kernel, one necessarily needs to consider an entire learning problem (with multiple different possibly labelings) rather than a specific source

distribution. Ghorbani et al. (2019a,b) take a different approach: they restrict the kernels being considered to NTKs of a specific model, NTKs of a class of models, or general rotationally invariant kernels, and so are able to establish separations for specific source distributions (between differentiable learning and using one of these specific kernels). These separations are more similar to our Separations 1 and 3, which are specific to the NTK of the model being used (theirs are more general), but their separations holds only for large input dimensions n, while ours holds already for dimension $n = 2$.

A deficiency of our Separations 3 and 4, shared also by Li et al. (2020); Allen-Zhu and Li (2019); Daniely and Malach (2020) and Ghorbani et al. (2019a), is that we do not demonstrate a fixed learning problem where the gradient descent error goes to zero. Instead, for every $\varepsilon > 0$, we construct a different problem where the gradient descent error is $\leq \varepsilon$ (while the kernel error is high; close to that of the null predictor). Ghorbani et al. (2019b) and Allen-Zhu and Li (2020), as well as our Separations 1 and 2, do use learning problem where gradient descent can get arbitrarily small error. It would be interesting to strengthen Separations 3 and 4 so that gradient descent could converge to zero error (with a fixed learning problem and model).

The separation results also differ in the form of gradient descent they analyze. Ghorbani et al. (2019a) and Daniely and Malach (2020) analyze gradient descent or gradient flow on the population, which is the limit of GD/SGD when the number of samples and/or iterations goes to infinity (although how quickly these should increase is not analyzed). But most analysis is for batch or stochastic gradient descent (which is what would be used in practice) with polynomial samples and iterations (Soltanolkotabi, 2017; Mei et al., 2018; Allen-Zhu and Li, 2019), perhaps with slight modifications or regularization (Li et al., 2020; Allen-Zhu and Li, 2020). Our analyses is for τ-approximate gradient descent with polynomial accuracy τ, which subsumes batch (or mini-batch) gradient descent with polynomially many samples (or batch size). But our analysis is perhaps odd and unnatural in that it relies on a single step with a large stepsize, rather than allowing the number of steps to increase and the stepsize to decrease. It should be possible, though is technically much more complicated, to extend our analysis to cover also gradient descent with any stepsize smaller than the one we use, and thus also gradient flow. This would establish strong separation also based on a more restrictive definition of differentiable learning, which requires robustness with respect to the stepsize.

Finally, the models we use are chosen to be simple to analyze, but they are perhaps not "natural". We do show how they can be implemented with a sigmoidal network, but this is a network with a very specific architecture, or alternatively a fully connected network with very specific initialization and where some of the weights are fixed rather than trained. Showing similarly strong separations with more natural architectures and random initialization would be desirable. The empirical demonstration in Figure 3 indicates this should be possible (though perhaps technically involved). Allen-Zhu and Li (2019, 2020) also use a specialized residual architecture (though perhaps not as specific as ours), while the others (Soltanolkotabi, 2017; Mei et al., 2018; Ghorbani et al., 2019a; Li et al., 2020; Daniely and Malach, 2020) use fairly benign networks and initialization.

7 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 3. 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. (2019a). 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 3 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 and Shamir (2019). 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

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

E. Abbe and C. Sandon. 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.

Z. Allen-Zhu and Y. Li. 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*, pages 9015–9025, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5857d68cd9280bc98d079fa912fd6740-Abstract.html.

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

Z. Allen-Zhu, Y. Li, and Y. Liang. 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*, pages 6155–6166, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/62dad6e273d32235ae02b7d321578ee8-Abstract.html.

S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang. 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*, pages 8139–8148, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/dbc4d84bfcfe2284ba11beffb853a8c4-Abstract.html.

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

L. Chizat, E. Oyallon, and F. Bach. 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*, pages 2933–2943, 2019a. URL http://papers.nips.cc/paper/8559-on-lazy-training-in-differentiable-programming.

L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2933–2943, 2019b.

A. Daniely and E. Malach. 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.

A. Daniely, R. Frostig, and Y. Singer. 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*, pages 2253–2261, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/abea47ba24142ed16b7d8fbf2c740e0d-Abstract.html.

S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine*

Learning, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 09–15 Jun 2019a. URL http://proceedings.mlr.press/v97/du19c.html.

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

M. Geiger, S. Spigler, A. Jacot, and M. Wyart. 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.

B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari. 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*, pages 9108–9118, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/c133fb1bb634af68c5088f3438848bfd-Abstract.html.

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

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

A. Jacot, C. Hongler, and F. Gabriel. 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*, pages 8580–8589, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html.

P. Kamath, O. Montasser, and N. Srebro. 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*, pages 2236–2262. PMLR, 2020. URL http://proceedings.mlr.press/v125/kamath20b.html.

Y. Li and Y. Liang. 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.

Y. Li, T. Ma, and H. R. Zhang. 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*, pages 2613–2682. PMLR, 2020. URL http://proceedings.mlr.press/v125/li20a.html.

S. Mei, Y. Bai, and A. Montanari. 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.

B. Schölkopf and A. J. Smola. *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.

M. Soltanolkotabi. 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*, pages 2007–2017, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/e034fb6b66aacc1d48f445ddfb08da98-Abstract.html.

B. E. Woodworth, S. Gunasekar, J. D. Lee, E. Moroshko, P. Savarese, I. Golan, D. Soudry, and N. Srebro. 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*, pages 3635–3673. PMLR, 2020. URL http://proceedings.mlr.press/v125/woodworth20a.html.

G. Yehudai and O. Shamir. 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*, pages 6594–6604, 2019. URL http://papers.nips.cc/paper/8886-on-the-power-and-limitations-of-random-features-for-understan

D. Zou, Y. Cao, D. Zhou, and Q. Gu. 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.

A Prior work on Separations between Differentiable Learning and Kernels

We provide more details on the prior work presented in Table 2 that demonstrate learning tasks where gradient-descent based learning can ensure smaller error than any reasonably sized kernel method.

In the summary below, whenever a $\mathrm{poly}(\cdot)$ terms appears under "Learnability with Gradient Descent", it is always for a particular $\mathrm{poly}(\cdot)$ that has been explicitly or implicitly specified in the corresponding reference. On the other hand, for every $\mathrm{poly}(\cdot)$ term that appears under "Non-learnability with any Kernel Method", it is always meant to hold for *any* $\mathrm{poly}(\cdot)$.

1. Yehudai and Shamir (2019); Ghorbani et al. (2019b) : **Class of ReLU neurons over Gaussian inputs**
   - ▷ *Learning Problem:* With $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}$, a source distribution is given by $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_n)$ and $y = [\langle w, x\rangle + b]_+$. To make the result comparable to others, in terms of the value of the loss, we consider a normalization of w and b such that the loss of the null predictor $\mathcal{L}(0) = 1/2$. In the original formulation of Yehudai and Shamir (2019), w, b are chosen s.t. $\|w\|, |b| \leq O(n^4)$, so that the value y is bounded by $O(n^4)$, and the loss of the null predictor is $O(n^8)$ (due to the square loss). Therefore, in order to have $\mathcal{L}(0) = \Theta(1)$, the values of w, b are scaled down by $1/n^4$, and the lower-bound on the loss is $\Omega(1/n^8)$ (and not $\Omega(1)$ as stated in the paper).
   - ▷ *Learnability with Gradient Descent:* Soltanolkotabi (2017); Mei et al. (2018) showed that, in the case when $b = 0$, projected batch gradient descent over the ReLU model with $\mathrm{poly}(n/\varepsilon)$ samples with zero initialization (i.e., unbiased initialization with $w^{(0)} = 0$), can ensure square loss at most ε with $O(\mathrm{poly}(n)\log(1/\varepsilon))$ steps. Importantly, this holds for any $\varepsilon > 0$. In the analysis of the positive result, initializing from zero is important, since in this case the first step of gradient-descent already captures the "direction" of the target neuron.
   - ▷ *Non-learnability with any Kernel Method:* Yehudai and Shamir (2019) show that any randomized feature map of dimension at most $2^{c_1 n}$ with coefficients of the predictors bounded by $2^{c_1 n}$, for some universal constant c_1, must incur a square loss of at least $\Omega(1/n^8)$; this is a scaled version of their stated result, which was for $\|w\| = n^3$ and $|b| \leq 6n^4 + 1$. Subsequently, Kamath et al. (2020) showed the same bound can be obtained without the restriction on coefficients of the predictors, leading to the lower bound as presented in the Table 2. Both of these lower bounds crucially rely on a non-zero bias term b.
     Because the upper bounds are specific to the case $b = 0$ where there is no bias term, while the lower bound relies on a bias term $b \neq 0$, the analysis of Yehudai and Shamir does not establish a separation between differentiable learning and kernel methods. It was communicated to use by Gilad Yehudai that ongoing work with Ohad Shaimr and Gal Vardi indicates that gradient descent also succeeds in the presence of a non-zero bias term, which would yield a separation. Alternatively, in our own yet unpublished work, we observed it is also possible to obtain a lower bound for ReLU without the bias term, which also yields a separation.
     Ghorbani et al. (2019b) show that for any constant c_2, any kernel method that is either (i) a rotationally invariant kernel using at most n^{c_2} samples or (ii) NTK of a depth-2 ReLU Net with at most n^{c_2} units must incur a square loss of at least $\varepsilon'(c_2)$, which is a positive constant that depends only on c_2. This result holds even for a fixed source distribution $\mathcal{D}$, that is, a fixed choice of w and $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_n)$. This result holds with probability approaching 1, as the dimension n and the number of features grow to infinity.

2. Ghorbani et al. (2019a) : **(i) Convex Quadratic Functions, (ii) Mixture of Gaussians (Binary Labels)**
   - ▷ *Learning Problems:* Two learning problems are considered here; both w.r.t. square loss.
     (i) With $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}$, a source distribution is given by Gaussian input marginals $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_n)$ and $y = b_0 + x^T B x$ for $B \succeq 0$.
     (ii) With $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \{-1, 1\}$, a source distribution is specified by $(\Sigma^{(1)}, \Sigma^{(-1)})$, where $y \sim \mathcal{U}(\{-1, 1\})$, and $x \mid y \sim \mathcal{N}(0, \Sigma^{(y)})$.
   - ▷ *Learnability with Gradient Descent:* For a depth-2 network with quadratic activations, with N units and any random initialization that is absolutely continuous w.r.t. Lebesgue measure (in particular, the initialization can be chosen to nearly unbiased), it is shown that in the asymptotic regime as $n, N \to \infty$ with $\rho = N/n < 1$, gradient flow w.r.t. population loss ensures square loss that approaches $\varepsilon(\rho)$, which is a closed form expression

that also depends on the specific source distribution (i.e. the quadratic target or the covariances of the classes), or rather sequence of source distributions as $n \to \infty$. The requirement on the initialization is needed since there is a measure-zero set of initializations that converge to a saddle point. Depending on the problem, the saddle points can be escaped also from an unbiased initialization.

- ▷ *Non-learnability with any Kernel Method:* It is shown in the same asymptotic regime of $n, N \to \infty$ with $\rho = N/n < 1$, that the NTK at initialization of the same model incurs a square loss of $\varepsilon'(\rho)$, which is a closed form expression that also depends on the sequence of source distributions as $n \to \infty$. It is shown that for non-degenerate distribution sequences, and any $\rho < 1$, that $\varepsilon'(\rho) > \varepsilon(\rho)$.

3. Li et al. (2020) : **A depth-2 Net, with abs. value activations and restricted weights, over Gaussian inputs**
   - ▷ *Learning Problem:* With $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}$, a source distribution is given by $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_n)$ and $y = \sum_{i=1}^{n} a_i \left|\langle w_i, x\rangle\right|$, which is a two layer network with absolute value activations, and additional constraints that w_i's are orthonormal and $a_i \in [\frac{1}{\kappa n}, \frac{\kappa}{n}]$ for some $\kappa > 1$.
   - ▷ *Learnability with Gradient Descent:* It is shown that there exists a depth-2 ReLU network with $\mathrm{poly}(n)$ units and a random normal initialization that is not unbiased, a type of truncated batch gradient descent (where large coordinates of the gradient are set to zero) using $\mathrm{poly}(n)$ samples, is shown to ensure square loss at most $\varepsilon := 1/n^{1+\delta}$ for a constant $\delta \in (0, 0.01)$ (that depends on κ) with $\mathrm{poly}(n)$ steps. Notably, this procedure is *not* shown to ensure an arbitrarily small error.
   - ▷ *Non-learnability with any Kernel Method:* It is shown that any kernel method with at most $\mathrm{poly}(n)$ features (for any $\mathrm{poly}(n)$) must incur a square loss of at least $\Omega(1/n) \geq \varepsilon^{1-\delta/2}$.

4. Allen-Zhu and Li (2019) : **Sparse Variable Selection + Parity (implementable as a depth-2 ResNet)**
   - ▷ *Learning Problem:* With $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}^k$ (for $k = \Theta(\log n)$), a source distribution is given by $\mathcal{D}_{\mathcal{X}} = \mathcal{U}(\{-1, 1\}^n)$ is uniform over Boolean hypercube and $y = Wx + \alpha\mathcal{G}(Wx)$, where $W = (e_{i_1}, \ldots, e_{i_k}) \in \mathbb{R}^{k \times n}$ for $i_1, \ldots, i_k \in [n]$ and $\mathcal{G}(z) := \prod_{i=1}^{k} z_i \cdot \mathbf{1}$ for $\alpha = 1/2^{\widetilde{\Theta}(k)} = 1/\mathrm{poly}(n)$. We can interpret this as Wx selecting a k sized subset of input coordinates, and $\mathcal{G}$ computes the parity of that subset, in each of the k coordinates.
   - ▷ *Learnability with Gradient Descent:* For an overparameterized depth-2 ResNet with $\mathrm{poly}(n/\varepsilon)$ units and a random normal initialization (that has a high variance, leading to a biased initialization), stochastic gradient descent is shown to ensure square loss of $\varepsilon \leq \alpha^{3.9}$ with $\mathrm{poly}(n, 1/\varepsilon)$ steps. Notably, it is *not* shown to ensure an arbitrarily small error.
   - ▷ *Non-learnability with any Kernel Method:* It is shown that any kernel method with at most $\mathrm{poly}(n)$ (for any $\mathrm{poly}(n)$) features must incur a square loss of at least $\Omega(\varepsilon^{0.52}) \geq \Omega(\alpha^2)$.

     Remark: Note that the learning problem here depends on the choice of ε. That is, for every n and $\varepsilon > 0$, there is a learning problem where stochastic gradient descent ensures square loss $\leq \varepsilon$ whereas no kernel method of dimension at most $\mathrm{poly}(n)$ can achieve a loss smaller than $\varepsilon^{0.52}$.

5. Allen-Zhu and Li (2020) : **A depth-L Net with quadratic activations**
   - ▷ *Learning Problem:* With $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}$, a source distribution is given by $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_n)$ and $y = N_\theta(x)$, where N_θ is a depth-L network with quadratic activations that has additional constraints on presence of skip connections, as well as constraints on the weights, that is referred to as the "information gap". L can be taken to be any parameter that is $o(\log \log n)$.
   - ▷ *Learnability with Gradient Descent:* For an overparameterized depth-L Net with quadratic activations, with identical layer structure as N_θ, with $\mathrm{poly}(n)$ units and zero initialization (unbiased), regularized stochastic gradient descent (with a non-standard regularizer) is shown to ensure square loss ε, with $\mathrm{poly}(n/\varepsilon)$ number of steps. Importantly, this holds for any $\varepsilon > 0$.
   - ▷ *Non-learnability with any Kernel Method:* It is shown that any kernel method of dimension at most $\mathrm{poly}(n)$ (for any $\mathrm{poly}(n)$) must incur a square loss of at least $\Omega(1/n^{0.01})$.

6. Daniely and Malach (2020) : **Sparse parities over a mixture of uniform and leaky marginals**

- ▷ *Learning Problem:* With $\mathcal{X} = \{-1,1\}^n$ and $\mathcal{Y} = \{-1,1\}$, the learning problem is similar to $\mathcal{P}^{\mathsf{lp}}[n,\alpha]$ that we consider in Section 5 with $\alpha = 1/2$. There are two differences: (i) The label under $\mathcal{D}_I^{(1)}$ is given by the parity, instead of being random and (ii) the sparsity of the parity is fixed to be k, which can be taken to be $k = \Theta(\sqrt[10]{n})$.
- ▷ *Learnability with Gradient Descent:* For a particular $\varepsilon = 1/\operatorname{poly}(n)$, a depth-2 net with ReLU6 activation with $\operatorname{poly}(n)$ units and an unbiased initialization (obtained by pairing the units with complimentary initialization, as suggested by Chizat et al. (2019a)), a regularized gradient descent (standard ℓ_2 regularization) over the *population loss* is shown to ensure hinge loss of ε, with $\operatorname{poly}(n)$ steps. Notably, it is *not* shown to ensure an arbitrarily small error.
- ▷ *Non-learnability with any Kernel Method:* It is shown that any kernel method of dimension at most $\operatorname{poly}(n)$ and norm at most $\operatorname{poly}(n)$ must incur a hinge loss of at least $1/3$ (recall that hinge loss is $\ell(\widehat{y},y) := \max\{1-\widehat{y}y,0\}$; and so that $\mathcal{L}_{\mathcal{D}}(0) = 1$ for any distribution $\mathcal{D}$ with ± 1 labels). In yet unpublished work extending Kamath et al. (2020), we show it is possible to obtain a lower bound with only the restriction on the norm (i.e. without restricting the dimension).

B Proofs of Gradient Based Learning

B.1 Proof from Section 3

Claim 3.1. *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 ensures $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(T)}}) = 0$. In particular, if $k \le \log n$ then an accuracy of $\tau \le \alpha/n$ is sufficient.*

Proof of Claim 3.1. We only rely on Equation 8, which gives us that at $\theta^{(0)} = 0$, we have for all $x \in \{-1,1\}^n$ that

$$f_{\theta^{(0)}}(x) = 0 \tag{30}$$
$$\nabla_\theta f_{\theta^{(0)}}(x) = 2x \tag{31}$$

and hence $\nabla_\theta \mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(0)}})$ is given by

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(0)}}) &= \nabla_\theta \left[\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I} \frac{1}{2}\left(f_{\theta^{(0)}}(x) - y\right)^2\right] = -\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I} y\cdot\nabla_\theta f_{\theta^{(0)}}(x) \\ &= -2\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I} y\cdot x = -2\mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\left(\prod_{i\in I} x_i\right)\cdot x.\end{aligned}$$

Thus, for all $j \in I$ we have

$$\frac{\partial}{\partial\theta_j}\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(0)}}) = -2\mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\prod_{i\in I\setminus\{j\}} x_i = -2\alpha\prod_{i\in I\setminus\{j\}}\mathop{\mathbb{E}}_{x\sim\mathcal{D}_1} x_i = -\frac{2\alpha}{2^{k-1}} = -\frac{4\alpha}{2^k},$$

and for all $j \notin I$ we have:

$$\frac{\partial}{\partial\theta_j}\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(0)}}) = -2\mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\prod_{i\in I\cup\{j\}} x_i = -\frac{2\alpha}{2^{k+1}} = -\frac{\alpha}{2^k}.$$

With step size $\eta = 2^k/(\alpha n)$, we have that after the first gradient step, (i) for all $j \in I$, $\theta_j^{(1)} \in [\frac{2^k}{\alpha n}\cdot(\frac{4\alpha}{2^k}\pm\tau)] = [\frac{3}{n},\frac{5}{n}]$, (ii) for all $j \notin I$, $\theta_j^{(1)} \in [\frac{2^k}{\alpha n}\cdot(\frac{\alpha}{2^k}\pm\tau)] = [0,\frac{2}{n}]$ (since $\tau = \alpha/2^k$). From Equation 8 we get $f_{\theta^{(1)}}(x) = \prod_{i\in I} x_i$, thereby concluding the proof. ∎

B.2 Proof from Section 5

Claim 5.1. *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 $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(T)}}) \leq \alpha$.*

Proof of Claim 5.1. We only rely on Equation 27, which gives us that at $\theta^{(0)} = 0$, we have for all $x \in \{-1,1\}^n$ that

$$f_{\theta^{(0)}}(x) = -1 \tag{32}$$
$$\nabla_\theta f_{\theta^{(0)}}(x) = 2 \cdot (x + \tfrac{5}{3}\alpha \mathbf{1}) \tag{33}$$

and hence $\nabla_\theta \mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(0)}})$ is given by

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(0)}}) &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I} (f_{\theta^{(0)}}(x) - y) \cdot \nabla_\theta f_{\theta^{(0)}}(x) \\
&= 2 \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I} (-1-y) \cdot (x + \tfrac{5}{3}\alpha\mathbf{1}) \\
&= 2(1-\alpha) \underbrace{\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I^{(0)}} (-1-y)x}_{=0} + 2\alpha \underbrace{\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I^{(1)}} (-1-y)x}_{=-x^I} + \underbrace{\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_I} (-1-y)}_{=-1} \cdot \tfrac{5}{3} \cdot 2\alpha\mathbf{1} \\
&= -2\alpha(x^I + \tfrac{5}{3}\mathbf{1})
\end{aligned}$$

where we get that $\mathbb{E}_{(x,y)\sim\mathcal{D}_I^{(0)}} x = 0 = \mathbb{E}_{(x,y)\sim\mathcal{D}_I^{(0)}} x \cdot y$, since $|I| \geq 2$.

With gradient accuracy of $\tau = \frac{4}{3}\alpha$, we get that any valid gradient estimate g must satisfy $g_i \in [-\frac{12\alpha}{3}, -\frac{20\alpha}{3}]$ for $i \in I$, and $g_i \in [0, -\frac{8\alpha}{3}]$ for $i \notin I$. Using step size $\eta = \frac{3}{4\alpha n}$, we get

$$\theta_i^{(1)} \in [\tfrac{3}{n}, \tfrac{5}{n}] \quad \text{for } i \in I \qquad \text{and} \qquad \theta_i^{(1)} \in [0, \tfrac{2}{n}] \quad \text{for } i \notin I$$

From Equation 27, we get that $f_{\theta^{(1)}}(x) = \prod_{i\in I} x_i$ and it is easy to see that $\mathcal{L}_{\mathcal{D}_I}(f_{\theta^{(1)}}) = \alpha$. ☐

C Lower Bounds on Kernel Methods

C.1 Lower Bounds for Tangent Kernels

C.1.1 NTK Edge in Section 3

Claim 3.2. *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 3.1. On the other hand, $\exists h \in \mathsf{NTK}^f_{\theta_0}(n)$ s.t. $\mathcal{L}_{\mathcal{D}_I}(h) \leq \frac{1}{2} - \frac{\alpha^2}{2^{2k}} = \frac{1}{2} - \Omega\left(\frac{n^2\tau^2}{C_f^2}\right)$*

Proof. It follows from (30) and (31) that the tangent feature map is $\phi_{\theta_0}(x) = x$, establishing that the NTK is the standard linear kernel, and that predictors in the NTK are just linear in x. For uniform inputs (i.e. from $\mathcal{D}_0$), no linear predictor $h_w(x) = \langle w, x\rangle$ is correlated with the parity if $k \geq 2$ bits, and so the expected square loss of any such predictor on $\mathcal{D}_0$ will be at least $\frac{1}{2}$, yielding $\mathcal{L}_{\mathcal{D}_I}(h_w) \geq (1-\alpha)\frac{1}{2} = \frac{1}{2} - \frac{\alpha}{2}$.

To obtain an upper bound on the error (lower bound on the edge), consider a linear predictor $h(x) = c \cdot Z$ where $Z = \sum_{i\in I} x_i$. By calculating $\mathbb{E}[Z^2] = k + \alpha(k^2-k)/4$ and $\mathbb{E}[Zy] = \alpha k/2^{k-1}$, and verifying the optimal scaling is $c = \mathbb{E}[Zy]/E[Z^2]$, we get that with this value of c, $\mathcal{L}_{\mathcal{D}_I}(h) = \frac{1}{2} - \gamma$, where

$$\gamma = \frac{\mathbb{E}[Zy]^2}{2\,\mathbb{E}[Z^2]} = \frac{\alpha^2}{2^{2k}} \cdot \frac{8}{\alpha + (4-\alpha)/k} \geq \frac{\alpha^2}{2^{2k}} \cdot \frac{8}{3} \geq \tau^2. \tag{34}$$

The actual optimal linear predictor is of the form $h(x) = c\sum_{i\in I} x_i + b\sum_{i\notin I} x_i$, and has a slightly better edge. ☐

C.1.2 NTK Edge in Section 5

Claim 5.2. *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 $n \geq 2$, $\alpha \in (0, 1)$, 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 $B \geq 0$.*

Proof. This follows from noting that for any $h_{w,b} \in \mathsf{NTK}^f_{\theta_0}(B)$ of the form $h_{w,b}(x) = \langle w, x\rangle + b$, it holds that $\mathbb{E}_{(x,y)\sim\mathcal{D}_I} h_{w,b}(x) \cdot y = 0$. Therefore, the optimal scaling of $h_{w,b}$ is 0, which incurs a loss $\mathcal{L}_{\mathcal{D}_I}(0) = \frac{1}{2}$. ☐

C.2 Lower Bounds for General Kernels

In order to prove lower bounds against general kernels, we recall the lower bounds proved on the dimension or norm of a (probabilistic) feature map needed to be able to represent certain hypothesis classes that were proved by Kamath et al. (2020), via probabilistic variants of dimensional and margin complexity.

Consider a "fixed-marginal" learning problem $\mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} \subseteq \mathbb{R}$, where each $\mathcal{D} \in \mathcal{P}$ corresponds to a hypotheses in a class $\mathcal{H} \subseteq (\mathcal{X} \to \mathcal{Y})$. Namely, sampling $(x, y) \sim \mathcal{D}_h$ (for any $h \in \mathcal{H}$) is equivalent to sampling $x \sim \mathcal{D}_{\mathcal{X}}$ (for a fixed marginal $\mathcal{D}_{\mathcal{X}}$) and setting $y = h(x)$. We say that $\mathcal{P}$ is "orthogonal" if $\mathbb{E}_{x\sim\mathcal{D}_{\mathcal{X}}} h(x)h'(x) = 0$ for all $h, h' \in \mathcal{H}$, and we say $\mathcal{P}$ is "normalized" if $\mathbb{E}_{x\sim\mathcal{D}_{\mathcal{X}}} h(x)^2 = 1$ for all $h \in \mathcal{H}$.

Theorem 3. *[Kamath et al. (2020)] Consider a fixed-marginal, orthogonal and normalized learning problem $\mathcal{P}$. For any probabilistic kernel K corresponding to p-dimensional feature maps, if for all $\mathcal{D} \in \mathcal{P}$ it holds for ℓ_{sq} that*

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$$

then, it holds that (i) $p \geq 2\gamma \cdot |\mathcal{P}|$ and (ii) $B^2 \geq \Omega(\gamma^3 \cdot |\mathcal{P}|)$.

Equivalently, we have that there exists $\mathcal{D} \in \mathcal{P}$ such that

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}}(h) \geq \frac{1}{2} - \min\left\{\frac{p}{2|\mathcal{P}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}|^{1/3}}\right)\right\}$$

Proof. Part (i). The claim of $p \geq 2\gamma \cdot |\mathcal{P}|$ follows immediately from Theorem 19 in Kamath et al. (2020).

Part (ii). It follows from Lemma 5 in Kamath et al. (2020) that irrespective of p, it is possible to construct a new randomized kernel K' corresponding to a feature map $\phi' : \mathcal{X} \to \mathbb{R}^{p'}$ with $p' \leq B^2 \cdot O\left(\frac{\frac{1}{2}-\gamma+\eta}{\eta^2}\right)$ for any choice of $\eta > 0$, to get that

$$\mathop{\mathbb{E}}_{K'} \inf_{h \in \mathcal{F}(K',B)} \mathcal{L}_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma + \eta$$

We set $\eta = \gamma/2$, so that we can take $p' \leq B^2 \cdot O(1/\gamma^2)$. From Part (i), we now have that $p' \geq 2(\gamma - \eta) \cdot |\mathcal{P}| = \gamma \cdot |\mathcal{P}|$. Putting everything together, we get that $B^2 \geq \Omega(\gamma^3 \cdot |\mathcal{P}|)$. ☐

We now use Theorem 3 to prove lower bounds in Sections 3 and 5. The key idea is that while, the learning problems considered are either not orthogonal, or do not have fixed marginals, they contain a large component that is a learning problem with fixed marginals that is orthogonal.

C.2.1 Proof from Section 3

Claim 3.3. *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 $|\mathcal{P}^{\mathsf{bsp}}[n, k, \alpha]| = \binom{n}{k}$.

Proof of Claim 3.3. Fix a probabilistic kernel K corresponding to p-dimensional feature maps. For all $\mathcal{D}_I \in \mathcal{P}$, we have

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \;\geq\; (1-\alpha) \cdot \mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I^{(0)}}(h) \;+\; \alpha \cdot \mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I^{(1)}}(h) \tag{35}$$

where $\mathcal{D}_I^{(b)}$ corresponds to sampling $x \sim \mathcal{D}_b$ and setting $y = \chi_I(x)$. We note that the second term, involving $\mathcal{D}_I^{(0)}$ is non-negative. To lower bound the first term, observe that the learning problem $\mathcal{P}^{(0)}$, consisting of distributions $\mathcal{D}_I^{(0)}$, has a fixed-marginal, and is orthogonal and normalized. Thus from Theorem 3 it follows that

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I^{(0)}}(h) \;\geq\; \frac{1}{2} - \min\left\{ \frac{p}{2|\mathcal{P}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}|^{1/3}} \right) \right\}$$

Combining with (35), we get

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \;\geq\; (1-\alpha) \cdot \left(\frac{1}{2} - \min\left\{ \frac{p}{2|\mathcal{P}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}|^{1/3}} \right) \right\} \right) \;\geq\; \frac{1}{2} - \frac{\alpha}{2} - \min\left\{ \frac{p}{2|\mathcal{P}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}|^{1/3}} \right) \right\}$$

This completes the proof by noting that $\mathcal{L}_{\mathcal{D}_I}(0) = 1/2$ for all $\mathcal{D}_I \in \mathcal{P}$. □

Remark 4. *One can also directly bound the optimal edge of a kernel with feature map $\phi^p = (\phi_1, \ldots, \phi_p)$ when the input distribution is i.i.d. of bias $b = 1/2$, without relying on the mixture component. Denoting by $\mathcal{D}_1^{(b)}$ the production measure with $\mathbb{E}x_i = b$, the optimal edge of a p-dimensional kernel*

$$\max_{\phi_1,\ldots,\phi_p} \min_{I \subseteq [n]:|I|=k} \left\| \chi_I / \langle \phi^p \rangle \right\|^2_{L_2(\mathcal{D}_1^{(b)})},$$

can be upper-bounded by splitting the contribution of features of the degree $\leq d$ and degree $> d$ polynomials, and optimizing over d. More specifically, defining the Fourier Walsh basis $\{\tilde{\chi}_I\}_{I \subseteq [n]}$ for the biased measure $\mathcal{D}_1^{(b)}$, one can decompose for $d \leq k$ the features by projecting them on $\{\tilde{\chi}_I\}_{I \subseteq [n]:|I| \leq d}$ and $\{\tilde{\chi}_I\}_{I \subseteq [n]:|I|>d}$. One can then crudely bound the first contribution of the edge by taking the best approximation in $\{\tilde{\chi}_I\}_{I \subseteq [n]:|I| \leq d}$, ignoring the dimension-p constraint of the features, which gives a bound of $\sum_{t=0}^{d} \binom{k}{t}(1-b^2)^t b^{2(k-t)}$. The second contribution can then be upper-bounded with techniques similar as to the previous section; with a bound that is multiplicative in p but exponential in $d \log(k/n)$, due to the high-degree contribution. Choosing d appropriately, such $d = \sqrt{\log(n)}$, then gives a vanishing edge (for both contributions and in total). The mixture distribution $\mathcal{D}_I$ allows instead for the more direct argument based on Kamath et al. (2020).

C.2.2 Proof from Section 5

Claim 5.3. *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*

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \;\geq\; \mathcal{L}_{\mathcal{D}_I}(0) - \min\left\{ \frac{p}{2|\mathcal{P}^{\mathsf{lp}}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}^{\mathsf{lp}}|^{1/3}} \right) \right\}.$$

where $\mathcal{L}_{\mathcal{D}_I}$ is w.r.t. the square loss, and note that $|\mathcal{P}^{\mathsf{lp}}[n, \alpha]| = 2^n - n - 1$.

Proof of Claim 5.3. Fix a probabilistic kernel K corresponding to p-dimensional feature maps. For all $\mathcal{D}_I \in \mathcal{P}$, we have

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I}(h) \;\geq\; (1-\alpha) \cdot \mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I^{(0)}}(h) + \alpha \cdot \mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I^{(1)}}(h)$$

Note that $\mathcal{L}_{\mathcal{D}_I^{(1)}}(h) \geq 1/2$ for any h. Now, the learning problem $\mathcal{P}^{(0)}$, consisting of distributions $\mathcal{D}_I^{(0)}$, has a fixed-marginal, and is orthogonal and normalized. Thus we have from Theorem 3 that

$$\mathop{\mathbb{E}}_{K} \inf_{h \in \mathcal{F}(K,B)} \mathcal{L}_{\mathcal{D}_I^{(0)}}(h) \geq \frac{1}{2} - \min\left\{\frac{p}{2|\mathcal{P}|}, O\left(\frac{B^{2/3}}{|\mathcal{P}|^{1/3}}\right)\right\}$$

This completes the proof by noting that $\mathcal{L}_{\mathcal{D}_I}(0) = 1/2$ for all $\mathcal{D}_I \in \mathcal{P}$. □