1 Introduction

The main point of this paper is to point out that the number of samples a kernel method (including Neural Tangent Kernels (NTKs)) requires to learn a composed hypothesis class up to a certain error is strictly more than a 3-layer ResNet learned with SGD requires in the multi-output regression regime – this amounts to a polynomial separation in the error (in a distribution-free setting, we will see that in N samples, 3-layer ResNet gets ℓ_2^2 error α^4, while there exists a distribution and function class for which kernel methods require more samples to merely get error α^2). Their analysis also implies the same result for linear functions over features. The main point is to illustrate the benefit of learning in a hierarchical fashion all at once.

The good solution that the ResNet finds (in their analysis) is required to have a distance away from the initialization bounded above in Frobenius norm as in the NTK analysis; however, the distinction from this analysis and the NTK regime is that the amount of overparameterization is significantly smaller — this allows the model which is learned to perform better than NTK would. As an addendum, there is unpublished work which keeps the overparameterization the same but increases the step size (no longer have infinitesmally small steps) and also adds noise to the updates — it is possible to show a separation from kernel methods here as well. Thus their work suggests that even if you analyze in the close-to-initialization regime, residual nets have a better sample complexity than kernels for some distributions and target concepts.

2 Concept Classes Considered

2.1 The learner

The learner is a residual net function $\mathbb{R}^d \to \mathbb{R}^k$ of the form

$$NN(\theta; x) := A(\sigma(Wx + b_1) + \sigma(U\sigma(Wx + b_1) + b_2))$$

where $W \in \mathbb{R}^{m\times d}, U \in \mathbb{R}^{m\times m}, A \in \mathbb{R}^{k\times m}$, σ is ReLU, and $b_1, b_2 \in \mathbb{R}^m$. $\theta = (W, U, b_1, b_2)$ are learned with SGD, while A is left untrained from init for simplicity.

2.2 Hierarchical Concept Class to Learn (Improperly)

We state the simple form of the concept class:

$$H(x) = F(x) + \alpha G(F(x))$$

where $\alpha \in [0, 1)$, $G : \mathbb{R}^k \to \mathbb{R}^k$, $F : \mathbb{R}^d \to \mathbb{R}^k$. Note that we require F and G to be expressed as a two-layer network with smooth activations. In particular, for each output coordinate $r \in [k]$,

$$F_r(x) = \sum_{i=1}^{p_F} a_{1,r,i} F_{r,i}(\langle w_{r,i}, x\rangle)$$

and

$$G_r(h) = \sum_{i=1}^{p_G} a_{2,r,i} G_{r,i}(\langle v_{r,i}, h\rangle)$$

where $a_1, a_2 \in \{-1, 1\}$. Assume the data inputs are unit norm. We also have $\|F(x)\|_2 \leq \sqrt{k} p_F C_s(F)$, $\|G(F(x))\|_2 \leq k p_F C_s(F) p_G C_s(G)$, and that G is $\sqrt{k} p_G C_s(G)$-Lipschitz.

Here,

$$C_s(F) = C \sum_{i=0}^{\infty} (i+1)|c_i|$$

where $F(z) = \sum_{i=0}^{\infty} c_i z^i$ is the Taylor expansion of F, C is a constant. When we consider the concept class above, to calculate upper bounds on $C_s(F)$, just take the maximum over $r \in [k]$.

Another similar measure which will show up in the amount of overparametrization (but not the sample complexity) required for the network is given by

$$C_\alpha(F) = \sum_{i=0}^{\infty} \left(C^i + \left(C \sqrt{\frac{\log(1/\alpha)}{i}} \right)^i \right) |c_i|$$

The idea here is that F is the simple class which accounts for most of the signal, and $G(F)$ is the hard part of the signal, which accounts for the "last 11% of the error", which is typically much harder to learn in practice.

The goal is to use the ResNet to improperly learn this class H in a distribution free setting under the ℓ_2^2 risk:

$$\mathbb{E}_{(x,y)\sim D} \|H(x) - NN(\theta; x)\|_2^2$$

2.3 Separation Concept Class

Consider the following concept class: Let

$$F(x) = \sqrt{d}(e_{i_1}, \cdots, e_{i_k})^T x$$

$$G(y) = \left[\prod_{j \in [k]} y_j \right]_{i \in [k]}$$

(the output is k-dimensional). Then the concept class is

$$\{H(x) = F(x) + \alpha G(F(x))\}$$

This concept class will be used to show a separation in sample complexity between 3-layer ResNets and kernels.

3 ResNets are good

The main theorem here says

Theorem 3.1. *For every* $\alpha \in (0, \tilde{\Theta}(1/kp_G C_s(G)))$, *with*

$$\delta > \tilde{\Theta}(\alpha^4 (kp_G C_s(G))^4 (1 + \sqrt{k} p_F C_s(F))^2)$$

Then, there exists $M = \text{poly}(C_\alpha(F), C_\alpha(G), p_F, 1/\alpha)$ *such that for every* $m > M$ *(over-parametrization size) with high probability over the initialization and for*

$$T = \tilde{\Theta}(\frac{(kp_F C_s(F))^2}{\min(1, \delta^2)})$$

and a specific choice of learning rate, the SGD algorithm satisfies

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{(x,y)\sim D} \|H(x) - NN(\theta_t; x)\|_2^2 \leq O(\delta)$$

Thus, with sample complexity T, *we can achieve population risk*

$$\leq \tilde{\Theta}(\alpha^4 (kp_G C_s(G))^4)$$

assuming that optimum error is 0.

First, assume $U = VA$. Then, we write down a decomposition:

$$\begin{aligned} NN(\theta; x) &= A\sigma(Wx + b_1) + A\sigma(VA\sigma(Wx + b_1) + b_2)) \\ &= SN(\theta; x) + A\sigma(VSN(\theta; x) + b_2) \end{aligned}$$

where

$$SN(\theta; x) = A\sigma(Wx + b_1)$$

Proof. (sketch) This theorem is proved by suggesting that

(a) Nice theoretical properties hold for weight matrices with bounded spectral norm (not far from random init). In particular, nonconvex interaction terms end up being negligible and can be safely ignored without linearizing.

(b) There exist good weight matrices $\hat{\theta}$ with small Frobenius norm, where "good" means that $SN(\hat{\theta}; x) \approx= F(x)$ and $A\sigma(VSN(\theta; x) + b_2) \approx \alpha G(F(x))$.

(c) Finally, look at the quantity $\langle \nabla_\theta \text{loss}(\theta_t, z_t), \theta_t - \hat{\theta} \rangle$ and show it is $\geq \text{loss}(\theta_t, z_t) -$ error term. This error term in expectation is upper bounded by $\tilde{\Theta}(\alpha^4 (kp_G C_s(G))^4)$. Since the top term can be shown to go to 0 for appropriately chosen learning rate, we thus get an upper bound on the loss in terms of the error term. Then using concentration arguments, one can show that the true error is close to the expectation.

□

4 Kernels are bad compared to 3-Layer ResNets

We now exhibit a specific distribution which shows a separation between 3-layer ResNets and kernel methods in sample complexity.

Definition 4.1. Bad distribution.
Take $x \in \{\pm 1/\sqrt{d}\}^d$, the scaled Boolean cube. Fix $d_1 < d$. Let the distribution be drawn from $x \sim D := \text{Unif}(\{\pm 1/\sqrt{d}\}_1^d) \times D_2$. Thus, the first d_1 coordinates are drawn from the uniform distribution over the Boolean cube, while the rest are drawn from an arbitrary distribution D_2. We will take D_2 to also be uniform, though the lower bound holds for arbitrary D_2.

Applying the ResNet theorem from before (recall it was a distribution free statement), we get that for overparametrization of order $\text{poly}(d, 2^k, 1/\alpha)$ then with high probability over the random init and optimum possible error 0, it requires

$$N = \tilde{\Theta}(\frac{k^2 d}{\alpha^8}) \text{ samples}$$

to get error

$$\mathbb{E}_{x\sim D}\|H(x) - NN(\theta; x)\|_2^2 \leq \tilde{O}(\alpha^4 2^{\mathcal{O}(k)})$$

We can also re-write this in terms of error parameter $\epsilon = \alpha^4 2^{\mathcal{O}(k)}$. Then, the sample complexity is

$$N = \tilde{\Theta}(\frac{k^2 4^{\mathcal{O}(k)} d}{\epsilon^2})$$

and we see there's an exponential dependence on k that goes away if you assume k is constant (which is somewhat reasonable, k is the output dimension). This follows from calculating $C_s(F) = \mathcal{O}(\sqrt{d}), p_F = 1, C_s(G) = 2^{\mathcal{O}(k)}, p_G = 2^k$.

Now we want to compare to kernel methods. It is possible to construct a specific bad distribution where the kernel method performs worse than the neural net, in that there is a lower bound on the possible attainable error for kernel method that is larger than the error a 3-layer ResNet can attain with a smaller number of samples.

The main theorem here states

Theorem 4.2. *For all k, d, N such that $2 \leq k \leq d$ and sample complexity $N \leq \frac{1}{1000}\binom{d}{k} \sim \mathcal{O}(d^k)$, for every $\alpha \in (0, 1)$, for every set of Mercer kernels $K_1, \cdots, K_k : \mathbb{R}^{d\times d} \to \mathbb{R}$, for at least 99% of target functions $H(x)$ in the previously defined concept class, for all kernel regression functions*

$$R_i(x) = \sum_{n\in[N]} K_i(x, x^{(n)}) \cdot w_{i,n} \text{ for } i \in [k]$$

where the weights can depend on α, X, K and training labels, $R(x)$ must suffer from population risk

$$\mathbb{E}_{x\sim\text{Unif}(\{\pm 1/\sqrt{d}\}^d)}\|H(x) - R(x)\|_2^2 > \alpha^2/16$$

Concretely, suppose we think of k as constant, d sufficiently large, and take $\alpha = \mathcal{O}(\frac{1}{d^{0.1}})$. Then,

(a) 3-layer ResNet achieves regression error $\alpha^{3.9}$ on the distribution with $N_{\text{ResNet}} = \tilde{\mathcal{O}}(d^{1.8})$ samples to learn any function in our special concept class;

(b) Kernel methods cannot achieve regression error $\alpha^2/16$ even with $N_{\text{Kernel}} \leq N_{\text{ResNet}}^{k/2} << o(d^k)$ samples. So you need at least $N_{\text{ResNet}}^{k/2}$ samples with Kernel method to achieve a LARGER generalization error.

(c) Note that this distinction between α^2 and α^4 matters – α^2 corresponds to the error you would get if you were able to learn F but not $\alpha G(F)$ in square ℓ_2 error. Thus, the separation is really saying the 3-layer ResNets can learn both error terms, while kernels can only learn the first F term and fail to learn the more complex part.

One caveat here – note that the complexity constants (smoothness, norm bound) from the ResNet theorem are getting hid as constant factors when k is taken to be constant.

5 Simliar result for Linear Regression over Features

When you consider the function class

$$L : \mathbb{R}^d \to \mathbb{R}^k$$

where

$$L_j(x) = w_j^T \phi(x)$$

where $w_j \in \mathbb{R}^D$, and some feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$. Then, a similar lower bound as in the kernel case holds as well.

5.1 Computational separation

Any algorithm over D-dimensional features runs in time $\Omega(D)$ with the absence of extra struture. Thus choosing $D =$ time to run ResNet will establish a computational gap between these methods for attaining error $\alpha^{3.9}$ for ResNet and error α^2 for linear feature map.

6 Experiments

One experiment is to suggest that SGD on neural nets doesn't find minimum norm solutions in practice, which is one approach that can be taken to show neural nets generalize better than kernel methods. They also validate that the model which theory predicts to do best in fact does best on a toy example, plotting amount of overparametrization against test error.

7 Key Points and Future Directions

Some key points from the discussion:

(a) Just learning F gets you already to α^2 error, and is a good starting point.

(b) Some high-level handwavy intuition: The hard function class and distribution for kernels is possible to learn if you don't try to learn them separately, but rather together – F tells you what the correct subset of coordinates is, and if you know the correct subset of coordinates, you can efficiently learn the parity function (as opposed to sparse parity, which is hard when you don't know the coordinates!). The kernel is running into problems because it is not "learning F and then learning $G(F)$ in an iterative manner" – It tries to learn both things individually, and sparse parity is hard. In essence, you need to extract that information of the correct subset from F first (which is easy if you do it in the right order, since most of the signal is F). Also importantly, however, is that you need to iterate this method, since you only get to see a corrupted signal from F (corrupted by $\alpha(G(F)))$ – you can think of the neural net as repeatedly learning F and then $G(F)$, and starting another round with F and G initialized with the solutions of the previous round, allowing both F and G to be learned well. This behavior is in some sense equivalent to being able to learn F purely, and after having learned F correctly, to learn $G(F)$ purely, where "purely" means without any contamination of the signals – thus, the neural net training procedure decouples the problems of learning F and learning $\alpha G(F))$ automatically.

Future work could involve many approaches to studying the power of function composition in efficient learning:

(a) Going beyond just 3 layers, and going beyond ResNets!

(b) Focusing on target function classes which are just compositions of functions (e.g., $G(F(x))$, not $F(x) + \alpha G(F(x)))$.

(c) Can we characterize the function classes and bad distributions which separate kernels from neural nets? The sparse parity situation is not necessarily typical of real-world practice, so it would be interesting to find separations more reflective of what practice observes.

8 Bibliography

[1] Allen-Zhu, Zeyuan and Li, Yuanzhi. What Can ResNet Learn Efficiently, Going Beyond Kernels?. ArXiv. https://arxiv.org/pdf/1905.10337.pdf.