Memorizing Gaussians with no over-parameterizaion via gradient decent on neural networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many results in recent years established polynomial time learnability of various models via neural networks algorithms (e.g. [3, 9, 7, 6, 17, 28, 18, 10, 4, 23, 19, 13, 5]). However, unless the model is linearly separable [5], or the activation is quadratic [13], these results require very large networks – much more than what is needed for the mere existence of a good predictor.

In this paper we make a step towards learnability results with near optimal network size. We give a tight analysis on the rate in which the Neural Tangent Kernel[16], a fundamental tool in the analysis of SGD on networks, converges to its expectations. This results enable us to prove that SGD on depth two neural networks, starting from a (non standard) variant of Xavier initialization [15] can memorize samples, learn polynomials with bounded weights, and learn certain kernel spaces, with *near optimal* network size, sample complexity, and runtime. In particular, we show that SGD on depth two network with $\tilde{O}\left(\frac{m}{d}\right)$ hidden neurons (and hence $\tilde{O}(m)$ parameters) can memorize m random labeled points in $\mathbb{S}^{d-1}$.

1 Introduction

Understanding the models (i.e. pairs $(\mathcal{D}, f^*)$ of input distribution $\mathcal{D}$ and target function f^*) on which neural networks algorithms guaranteed to learn a good predictor is at the heart of deep learning theory today. In recent years, there has been an impressive progress in this direction. It is now known that neural networks algorithms can learn, in polynomial time, linear models, certain kernel spaces, polynomials, and memorization models (e.g. [3, 9, 7, 6, 17, 28, 18, 10, 4, 23, 19, 13, 5]).

Yet, while such models has been shown to be learnable in polynomial time and polynomial sized networks, the required size (i.e., number of parameteres) of the networks is still very large, unless the model is linear separable [5], or the activation is quadratic [13]. This means that the proofs are valid for networks whose size is significantly larger then the minimal size of the network that implements a good predictor[1].

In this paper we make a progress in this direction. We first consider the neural tangent kernel [16], which is a linearization of the functions that can be computed by the network, with weights that are close to a given weight vector $\mathbf{w}$. The NTK is one of the main technical tools in recent analysis of SGD on neural networks. Our first result is a near optimal bound on the rate in which the NTK converge to its expectation. We then utilize this results, and prove that it implies that SGD on depth two networks, starting form a (somewhat non-standard) variant of Xavier initialization [15] can

[1]More specifically, we mean that the proofs require number of parameters that is suboptimal by a multiplicative factor that grows polynomially with one of the problem parameters – either the model capacity (margin, VC dimension, etc.), the desired error (i.e. ϵ), or the input dimension.

learn memorization models, polynomials, and kernel spaces, with *near optimal* network size, sample complexity, and runtime (i.e. SGD iterations).

To the best of our knowledge, this is the first result which shows near optimal learnability of these models, and we believe that the result about NTK will be an essential tool for further progress, and in particular for proving a similar results for additional settings, architectures, and initialization schemes (particularly, the standard Xavier initialization). We next give more details about our results.

Neural Network Algorithm We assume that the instance space is $\mathbb{S}^{d-1}$ and consider depth 2 networks with $2q$ hidden neurons. Such networks calculate a function of the form

$$h_{W,\mathbf{u}}(\mathbf{x}) = \sum_{i=1}^{2q} u_i\sigma\left(\langle \mathbf{w}_i, \mathbf{x}\rangle\right) = \langle \mathbf{u}, \sigma\left(W\mathbf{x}\right)\rangle$$

We assume that the network is trained via SGD, starting from random weights that are sampled from the following variant of Xavier initialization [15]: W will be initialized to be a duplication $W = \begin{bmatrix} W' \\ W' \end{bmatrix}$ of a matrix W' of standard Gaussians and $\mathbf{u}$ will be a duplication of the all-B vector in dimension q, for some $B > 0$, with its negation. We will use rather large B, that will depend on the model that we want to learn.

Bounded distributions Some of our results will depend on what we call the boundedness of the data distribution. We say that a distribution $\mathcal{D}$ on $\mathbb{S}^{d-1}$ is R-bounded if for every $\mathbf{u} \in \mathbb{S}^{d-1}$, $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \langle \mathbf{u}, \mathbf{x}\rangle^2 \le \frac{R^2}{d}$. To help the reader to calibrate our results, we first note that by Cauchy-Schwartz, any distribution $\mathcal{D}$ is $\sqrt{d}$-bounded, and this bound is tight in the cases that $\mathcal{D}$ is supported on a single point. Despite that, many distributions of interest are $O(1)$-bounded or even $(1 + o(1))$-bounded. This includes the uniform distribution on $\mathbb{S}^{d-1}$, the uniform distribution on the discrete cube $\left\{\pm\frac{1}{\sqrt{d}}\right\}^d$, the uniform distribution on $\Omega\left(d\right)$ random points, and more (see section A.5). For simplicity, we will phrase our results in the introduction for $O(1)$-bounded distribution. We note that if the distribution is R-bounded (rather than $O(1)$-bounded), our results suffer a multiplicative factor of R^2 in the number of parameters, and remains the same in the runtime (SGD steps).

NTK Convergence For weights $(W, \mathbf{u})$ and $\mathbf{x} \in \mathbb{S}^{d-1}$ we denote by $\Psi_{W,\mathbf{u}}(\mathbf{x}) \in \mathbb{R}^{2q\times d}$ the gradient, w.r.t. the hidden weights W, of $h_{W,\mathbf{u}}(\mathbf{x})$. (A slight variant of) The NTK at W is

$$k_W(\mathbf{x}, \mathbf{y}) = \frac{\langle \Psi_{W,\mathbf{u}}(\mathbf{x}), \Psi_{W,\mathbf{u}}(\mathbf{y})\rangle}{2qB^2}$$

And the expected initial NTK is $k(\mathbf{x}, \mathbf{y}) = \mathbb{E}_W\, k_W(\mathbf{x}, \mathbf{y})$ Our main technical contribution is near optimal analysis of the rate (it terms of the size of the network) in which k_W converges to k. Specifically, we show that for any $O(1)$-bounded distribution, and every function $f : \mathbb{R}^d \to \mathbb{R}$ in the kernel space $\mathcal{H}_k$ corresponding to k, there is a function $\hat{f}$ in the kernel space $\mathcal{H}_{k_W}$ corresponding to k_W such that

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}(f(x) - \hat{f}(x))^2 = O\left(\frac{\|f\|_k^2}{dq}\right)$$

Here, $\|\cdot\|_k$ denotes the kernel norm of f. The proof of the aforementioned result is based on a new analysis of *vector* random feature schemes. While standard analysis of random feature schemes would lead to a bound of the form $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}(f(x) - \hat{f}(x))^2 = O\left(\frac{\|f\|_k^2}{q}\right)$, our new analysis show that for $O(1)$-bounded distributions, a factor of the input dimension d can be saved.

As mentioned above, we utilize our result for NTK convergence to prove various learnability results for SGD on depth two networks.

Memorization In the problem of memorization, we consider SGD training on top of a sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$. The goal is to understand how large the networks should be, and (to somewhat leaser extent) how many SGD steps are needed in order to memorize $1 - \epsilon$ fraction of the examples, where an example is considered memorized if $y_i h(\mathbf{x}_i) > 0$ for the output function h. Many results assumes that the points are random or “look like random" in some sense.

In order to memorize even just slightly more that half of the m examples we need a network with at least m parameters (up to poly-log factors). However, unless $m \le d$ (in which case the points are linearly separable), best know results require much more than m parameters, and the current state of the art results [23, 19] require m^2 parameters. We show that if the points are sampled uniformly at random from $\mathbb{S}^{d-1}$, and the labels are random, then *any fraction* of the examples can be memorized by a network with $\tilde{O}(m)$ parameters, and $\tilde{O}\left(\frac{m}{\epsilon^2}\right)$ SGD iterations. Our result is valid for the hinge loss, and most popular activation functions, including the ReLU.

Learning Polynomials For the sake of clarity, we will describe our result for learning even polynomials, with ReLU networks, and the loss being the logistic loss or the hinge loss. Fix a constant integer $c > 0$ and consider the class of even polynomials of degree $\le c$ and coefficient vector norm at most M. Namely, $\mathcal{P}_c^M = \left\{p(\mathbf{x}) = \sum_{|\alpha| \text{ is even and } \le c} a_\alpha \mathbf{x}^\alpha : \sum_{|\alpha| \text{ is even and } \le c} a_\alpha^2 \le M^2\right\}$ where for $\alpha \in \{0, 1, 2, \ldots\}^d$ and $\mathbf{x} \in \mathbb{R}^d$ we denote $\mathbf{x}^\alpha = \prod_{i=1}^d x_i^{\alpha_i}$ and $|\alpha| = \sum_{i=1}^d \alpha_i$. Learning the class $\mathcal{P}_d^M$ requires a networks with at least $\Omega\left(M^2\right)$ parameters (and this remains true even if we restrict to $O(1)$-bounded distributions). We show that for $O(1)$-bounded distributions, SGD learns $\mathcal{P}_c^M$, with error parameter ϵ (that is, it returns a predictor with error $\le \epsilon$), using a network with $\tilde{O}\left(\frac{M^2}{\epsilon^2}\right)$ parameters and $O\left(\frac{M^2}{\epsilon^2}\right)$ SGD iterations.

Learning Kernel Spaces Our result for polynomials is a corollary of a more general result about learning certain kernel spaces, that we describe next. Our result about memorization is not a direct corollary, but is also a refinement of that result. We consider the kernel $k : \mathbb{S}^{d-1} \times \mathbb{S}^{d-1} \to \mathbb{R}$ given by

$$k(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle \cdot \mathop{\mathbb{E}}_{\mathbf{w} \sim \mathcal{N}(I,0)} \sigma'\left(\langle \mathbf{w}, \mathbf{x} \rangle, \langle \mathbf{w}, \mathbf{y} \rangle\right) \tag{1}$$

which is a variant of the Neural Tangent Kernel [16]. We show that for $O(1)$-bounded distributions, SGD learns functions with norm $\le M$ in the corresponding kernel space, with error parameter ϵ, using a network with $\tilde{O}\left(\frac{M^2}{\epsilon^2}\right)$ parameters and $O\left(\frac{M^2}{\epsilon^2}\right)$ SGD iterations. We note that the network size is optimal up to the dependency on ϵ and poly-log factors, and the number of iteration is optimal up to a constant factor. This result is valid for most Lipschitz losses including the hinge loss and the log-loss, and for most popular activation functions, including the ReLU.

1.1 Related Work

The connection between networks, kernels and random features has a long history. Early work includes [25, 21]. In recent years, this connection was utilized to analyze neural networks algorithm (e.g. [3, 9, 7, 6, 17, 28, 18, 10, 4, 23, 19, 13]). In fact, the vast majority of known non-linear learnable models, including memorization models, polynomials, and kernel spaces utilize this connection. It is worth mentioning very recent papers [8, 27, 1, 14] that proves learnability beyond NTK.

It is hard to quantitatively compare the various result about learning polynomials and kernels, as they often depend on various parameters of the distributions, and talk about different kernels and polynomial spaces. Yet, to the best of our knowledge, in none of the known results the network size is optimal in both the input dimension and the kernel norm as theorems 5 and 6. In this regard, we would like to mention Ji and Telgarsky [17] which has optimal (logarithmic) dependence on ϵ in the case that the input distribution is realizable with margin in the NTK space. This should be compared to our dependance which is on one hand quadratic in $1/\epsilon$, but on the other hand valid for both the realizable and un-realizable settings.

As for memorization results, as mentioned above, results with about near optimal network size either consider linearly separable data [5] or quadratic activation [13]. As for non-polynomial activations and non-linearly-separable data, the results of Daniely [7] imply that under rather mild conditions, m points with arbitrary labels can be memorized by networks of size $\mathrm{poly}(m)$, but without an exact specification of the exponent of the polynomial. Allen-Zhu et al. [2] showed a memorization result using $\tilde{O}(m^{24})$ parameters. Zou and Gu [28] improved the bound to $\tilde{O}(m^8)$, then to $\tilde{O}(m^6)$ by Du et al. [10] and Wu et al. [26], to $\tilde{O}(m^4)$ by Du et al. [11], and finally, the state of the art until our work was memorization with $\tilde{O}(m^2)$ parameters [23, 20]. We would also like to mention Fiat et al. [12]

whose result shares some ideas with our proof. In their paper it is shown that for the ReLU activation, linear optimization over the embedding $\Psi_{W,\mathbf{u}}$ can memorize m points with $\tilde{O}(m)$ parameters.

2 Preliminaries

2.1 Notation

We denote vectors by bold-face letters (e.g. $\mathbf{x}$), matrices by upper case letters (e.g. W), and collection of matrices by bold-face upper case letters (e.g. $\mathbf{W}$). We denote the i's row in a matrix W by $\mathbf{w}_i$. The p-norm of $\mathbf{x} \in \mathbb{R}^d$ is denoted by $\|\mathbf{x}\|_p = \left(\sum_{i=1}^d |x_i|^p\right)^{\frac{1}{p}}$, and for a matrix W, $\|W\|$ is the spectral norm $\|W\| = \max_{\|\mathbf{x}\|=1} \|W\mathbf{x}\|$. We will also use the convention that $\|\mathbf{x}\| = \|\mathbf{x}\|_2$. For a distribution $\mathcal{D}$ on a space $\mathcal{X}$, $p \geq 1$ and $f : \mathcal{X} \to \mathbb{R}$ we denote $\|f\|_{p,\mathcal{D}} = (\mathbb{E}_{x\sim\mathcal{D}} |f(x)|^p)^{\frac{1}{p}}$. We denote by $L^2(\mathcal{X}, \mathbb{R}^d)$ the space of functions $f : \mathcal{X} \to \mathbb{R}^d$ with $\mathbb{E}_{x\sim\mathcal{D}} \|f(x)\|^2 < \infty$. Note that it is an inner product space w.r.t. the inner product $\langle f, g\rangle_{L^2(\mathcal{X},\mathbb{R}^d)} = \mathbb{E}_{x\sim\mathcal{D}} \langle f(x), g(x)\rangle$. We use $\tilde{O}$ to hide poly-log factors.

2.2 Supervised learning

The goal in supervised learning is to devise a mapping from the input space $\mathcal{X}$ to an output space $\mathcal{Y}$ based on a sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, where $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ drawn i.i.d. from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. In our case, the instance space will always be $\mathbb{S}^{d-1}$. A supervised learning problem is further specified by a loss function $\ell : \mathbb{R} \times \mathcal{Y} \to [0, \infty)$, and the goal is to find a predictor $h : \mathcal{X} \to \mathbb{R}$ whose loss, $\mathcal{L}_{\mathcal{D}}(h) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \ell(h(\mathbf{x}), y)$, is small. The *empirical* loss $\mathcal{L}_S(h) := \frac{1}{m}\sum_{i=1}^m \ell(h(\mathbf{x}_i), y_i)$ is commonly used as a proxy for the loss $\mathcal{L}_{\mathcal{D}}$. When h is defined by a vector $\mathbf{w}$ of parameters, we will use the notations $\mathcal{L}_{\mathcal{D}}(\mathbf{w}) = \mathcal{L}_{\mathcal{D}}(h)$, $\mathcal{L}_S(\mathbf{w}) = \mathcal{L}_S(h)$ and $\ell_{(\mathbf{x},y)}(\mathbf{w}) = \ell(h(\mathbf{x}), y)$. For a class $\mathcal{H}$ of predictors from $\mathcal{X}$ to $\mathbb{R}$ we denote $\mathcal{L}_{\mathcal{D}}(\mathcal{H}) = \inf_{h\in\mathcal{H}} \mathcal{L}_{\mathcal{D}}(h)$ and $\mathcal{L}_S(\mathcal{H}) = \inf_{h\in\mathcal{H}} \mathcal{L}_S(h)$

A loss ℓ is L-Lipschitz if for all $y \in \mathcal{Y}$, the function $\ell_y(\hat{y}) := \ell(\hat{y}, y)$ is L-Lipschitz. Likewise, it is convex if ℓ_y is convex for every $y \in \mathcal{Y}$. We say that ℓ is *L-decent* if for every $y \in \mathcal{Y}$, ℓ_y is convex, L-Lipschitz, and twice differentiable in all but finitely many points.

2.3 Neural network learning

We will consider fully connected neural networks of depth 2 with $2q$ hidden neurons and activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Throughout, we assume that the activation function is continuous, is twice differentiable in all but finitely many points, and there is $M > 0$ such that $|\sigma'(x)|, |\sigma''(x)| \leq M$ for every point $x \in \mathbb{R}$ for which f is twice differentiable in x. We call such an activation a *decent* activation. This includes most popular activations, including the ReLU activation $\sigma(x) = \max(0, x)$, as well as most sigmoids.

Denote $\mathcal{N}^{\sigma}_{d,q} = \left\{h_{\mathbf{W}}(\mathbf{x}) = \langle \mathbf{u}, \sigma(W\mathbf{x})\rangle : W \in M_{2q,d}, \mathbf{u} \in \mathbb{R}^{2q}\right\}$. We also denote by $\mathbf{W} = (W, \mathbf{u})$ the aggregation of all weights. We next describe the learning algorithm that we analyze in this paper. We will use a variant of the popular Xavier initialization [15] for the network weights, which we call *Xavier initialization with zero outputs*. The neurons will be arranged in pairs, where each pair consists of two neurons that are initialized identically, up to sign. Concretely, the weight matrix W will be initialized to be a duplication $W = \begin{bmatrix} W' \\ W' \end{bmatrix}$ of a matrix W' of standard Gaussians[2] and $\mathbf{u}$ will be a duplication of the all-B vector in dimension q, for some $B > 0$, with its negation. We denote the distribution of this initialization scheme by $\mathcal{I}(d, q, B)$. Note that if $\mathbf{W} \sim \mathcal{I}(d, q, B)$ then w.p. 1, $\forall\mathbf{x},\ h_{\mathbf{W}}(\mathbf{x}) = 0$. Finally, the training algorithm is described in 1.

[2]It is more standard to assume that the instances has L^2 norm $O\left(\sqrt{d}\right)$, or infinity norm $O(1)$, and the entries of W' has variance $\frac{1}{d}$. For the sake of notational convenience we chose a different scaling—divided the instances by $\sqrt{d}$ and accordingly multiplied the initial matrix by $\sqrt{d}$. Identical results can be derived for the more standard convention.

Algorithm 1 Neural Network Training

```
Input: Network parameters $\sigma$ and $d, q$, loss $\ell$, initialization parameter $B > 0$, learning rate $\eta > 0$, batch size $b$, number of steps $T > 0$, access to samples from a distribution $\mathcal{D}$
Sample $\mathbf{W}^1 \sim \mathcal{I}(d, q, B)$
for $t = 1, \dots, T$ do
    Obtain a mini-batch $S_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^b \sim \mathcal{D}^b$
    With back-propagation, calculate a stochastic gradient $\nabla \mathcal{L}_{S_t}(\mathbf{W}^t)$ and update $\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \nabla \mathcal{L}_{S_t}(\mathbf{W}^t)$
end for
Choose $t \in [T]$ uniformly at random and return $\mathbf{W}_t$
```

2.4 Kernel spaces

Let $\mathcal{X}$ be a set. A *kernel* is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for every $x_1, \dots, x_m \in \mathcal{X}$ the matrix $\{k(x_i, x_j)\}_{i,j}$ is positive semi-definite. A *kernel space* is a Hilbert space $\mathcal{H}$ of functions from $\mathcal{X}$ to $\mathbb{R}$ such that for every $x \in \mathcal{X}$ the linear functional $f \in \mathcal{H} \mapsto f(x)$ is bounded. The following theorem describes a one-to-one correspondence between kernels and kernel spaces.

Theorem 1. *For every kernel k there exists a unique kernel space $\mathcal{H}_k$ such that for every $x, x' \in \mathcal{X}$, $k(x, x') = \langle k(\cdot, x), k(\cdot, x') \rangle_{\mathcal{H}_k}$. Likewise, for every kernel space $\mathcal{H}$ there is a kernel k for which $\mathcal{H} = \mathcal{H}_k$.*

We denote the norm and inner product in $\mathcal{H}_k$ by $\| \cdot \|_k$ and $\langle \cdot, \cdot \rangle_k$. The following theorem describes a tight connection between kernels and embeddings of X into Hilbert spaces.

Theorem 2. *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only if there exists a mapping $\Psi : \mathcal{X} \to \mathcal{H}$ to some Hilbert space for which $k(x, x') = \langle \Psi(x), \Psi(x') \rangle_{\mathcal{H}}$. In this case, $\mathcal{H}_k = \{f_{\Psi,\mathbf{v}} \mid \mathbf{v} \in \mathcal{H}\}$ where $f_{\Psi,\mathbf{v}}(x) = \langle \mathbf{v}, \Psi(x) \rangle_{\mathcal{H}}$. Furthermore, $\|f\|_k = \min\{\|\mathbf{v}\|_{\mathcal{H}} : f_{\Psi,\mathbf{v}}\}$ and the minimizer is unique.*

For a kernel k and $M > 0$ we denote $\mathcal{H}_k^M = \{h \in \mathcal{H}_k : \|h\|_k \le M\}$. We note that spaces of the form $\mathcal{H}_k^M$ often form a benchmark for learning algorithms.

2.5 The Neural Tangent Kernel

Fix network parameters σ, d, q and B. The *neural tangent kernel* corresponding to weights $\mathbf{W}$ is[3]

$$\mathrm{tk}_{\mathbf{W}}(\mathbf{x}, \mathbf{y}) = \frac{\langle \nabla_{\mathbf{W}} h_{\mathbf{W}}(\mathbf{x}), \nabla_{\mathbf{W}} h_{\mathbf{W}}(\mathbf{y}) \rangle}{2qB^2}$$

The neural tangent kernel space, $\mathcal{H}_{\mathrm{tk}_{\mathbf{W}}}$, is a linear approximation of the trajectories in which h_W changes by changing W a bit. Specifically, $h \in \mathcal{H}_{\mathrm{tk}_{\mathbf{W}}}$ if and only if there is $\mathbf{U}$ such that

$$\forall \mathbf{x} \in \mathbb{S}^{d_1 - 1}, \;\; h(\mathbf{x}) = \lim_{\epsilon \to 0} \frac{h_{\mathbf{W} + \epsilon \mathbf{U}}(\mathbf{x}) - h_{\mathbf{W}}(\mathbf{x})}{\epsilon} \tag{2}$$

Furthermore, we have that $\sqrt{q}B \cdot \|h\|_{\mathrm{tk}_{\mathbf{W}}}$ is the minimal Euclidean norm of $\mathbf{U}$ that satisfies equation (2). The *expected initial neural tangent kernel* is

$$\mathrm{tk}_{\sigma,B}(\mathbf{x}, \mathbf{y}) = \mathrm{tk}_{\sigma,d,q,B}(\mathbf{x}, \mathbf{y}) = \mathop{\mathbb{E}}_{\mathbf{W} \sim (d,q,B)} \mathrm{tk}_{\mathbf{W}}(\mathbf{x}, \mathbf{y})$$

We will later see that $\mathrm{tk}_{\sigma,d,q,B}$ depends only on σ and B. If the network is large enough, we can expect that at the onset of the optimization process, $\mathrm{tk}_{\sigma,B} \approx k_{\mathbf{W}}$. Hence, approximately, $\mathcal{H}_{\mathrm{tk}_{\sigma,B}}$ consists of the directions in which the initial function computed by the network can move. Since the initial function (according to Xavier initialization with zero outputs) is 0, $\mathcal{H}_{\mathrm{tk}_{\sigma,B}}$ is a linear approximation of the space of functions computed by the network in the vicinity of the initial weights. NTK theory based of the fact close enough to the initialization point, the linear approximation is good, and hence SGD on NN can learn functions in $\mathcal{H}_{\mathrm{tk}_{\sigma,B}}$ that has sufficiently small norm. The main question is how small should the norm be, or alternatively, how large should the network be.

[3]The division by $2qB^2$ is for notational convenience.

We next derive a formula for $\mathrm{tk}_{\sigma,B}$. We have, for $\mathbf{W} \sim \mathcal{I}(d,q,B)$

$$\begin{aligned}\mathrm{tk}_{\mathbf{W}}(\mathbf{x},\mathbf{y}) &= \frac{\langle \nabla_{\mathbf{W}} h_{\mathbf{W}}(\mathbf{x}), \nabla_{\mathbf{W}} h_{\mathbf{W}}(\mathbf{y})\rangle}{2qB^2} \\ &= \frac{1}{qB^2}\sum_{i=1}^{q} \langle B\sigma'\left(\langle \mathbf{w}_i,\mathbf{x}\rangle\right)\mathbf{x}, B\sigma'\left(\langle \mathbf{w}_i,\mathbf{y}\rangle\right)\mathbf{y}\rangle + \frac{1}{qB^2}\sum_{i=1}^{q}\sigma\left(\langle \mathbf{w}_i,\mathbf{x}\rangle\right)\sigma\left(\langle \mathbf{w}_i,\mathbf{y}\rangle\right) \\ &= \frac{\langle \mathbf{x},\mathbf{y}\rangle}{q}\sum_{i=1}^{q}\sigma'\left(\langle \mathbf{w}_i,\mathbf{x}\rangle\right)\sigma'\left(\langle \mathbf{w}_i,\mathbf{y}\rangle\right) + \frac{1}{qB^2}\sum_{i=1}^{q}\sigma\left(\langle \mathbf{w}_i,\mathbf{x}\rangle\right)\sigma\left(\langle \mathbf{w}_i,\mathbf{y}\rangle\right)\end{aligned}$$

Taking expectation we get

$$\mathrm{tk}_{\sigma,B}(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle\,\hat{\sigma}'\left(\langle \mathbf{x},\mathbf{y}\rangle\right) + \frac{1}{B^2}\hat{\sigma}\left(\langle \mathbf{x},\mathbf{y}\rangle\right) = \langle \mathbf{x},\mathbf{y}\rangle\,k_{\sigma'}(\mathbf{x},\mathbf{y}) + \frac{1}{B^2}k_{\sigma}(\mathbf{x},\mathbf{y})$$

Finally, we decompose the expected initial neural tangent kernel into two kernels, that corresponds to the hidden and output weights respectively. Namely, we let

$$\mathrm{tk}_{\sigma,B} = \mathrm{tk}^h_{\sigma,B} + \mathrm{tk}^o_{\sigma,B} \text{ for } \mathrm{tk}^h_{\sigma}(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle\,\hat{\sigma}'\left(\langle \mathbf{x},\mathbf{y}\rangle\right) \text{ and } \mathrm{tk}^o_{\sigma,B}(\mathbf{x},\mathbf{y}) = \frac{1}{B^2}\hat{\sigma}\left(\langle \mathbf{x},\mathbf{y}\rangle\right)$$

Accordingly, we denote

$$\mathrm{tk}^h_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = \frac{\langle \mathbf{x},\mathbf{y}\rangle}{q}\sum_{i=1}^{q}\sigma'\left(\langle \mathbf{w}_i,\mathbf{x}\rangle\right)\sigma'\left(\langle \mathbf{w}_i,\mathbf{y}\rangle\right) \text{ and } \mathrm{tk}^o_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = \frac{1}{qB^2}\sum_{i=1}^{q}\sigma\left(\langle \mathbf{w}_i,\mathbf{x}\rangle\right)\sigma\left(\langle \mathbf{w}_i,\mathbf{y}\rangle\right)$$

2.6 Vector Random Feature Schemes

Random features schemes [25, 21] introduced as a mean for developing fast algorithm for learning kernel spaces. Here, we will use random features as a tool for analyzing SGD on networks. Let $\mathcal{X}$ be a measurable space and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel. A *random features scheme* (RFS) for k is a pair (ψ, μ) where μ is a probability measure on a measurable space Ω, and $\psi : \Omega \times \mathcal{X} \to \mathbb{R}^d$ is a measurable function such that

$$\forall \mathbf{x},\mathbf{x}' \in \mathcal{X}, \quad k(\mathbf{x},\mathbf{x}') = \mathop{\mathbb{E}}_{\omega\sim\mu}\left[\langle \psi(\omega,\mathbf{x}), \psi(\omega,\mathbf{x}')\rangle\right] . \tag{3}$$

We often refer to ψ (rather than (ψ,μ)) as the RFS. Our motivation form considering vector RFS in this paper steams from the *NTK RFS*, which is given by the mapping $\psi : \mathbb{R}^d \times \mathbb{S}^{d-1} \to \mathbb{R}^d$ defined by $\psi(\omega,\mathbf{x}) = \sigma'(\langle \omega,\mathbf{x}\rangle)\mathbf{x}$ and μ being the standard Gaussian measure on $\mathbb{R}^d$. Note that it is an RFS for the kernel tk^h_σ (see section 2.5).

We define the *norm* of ψ as $\|\psi\| = \sup_{\omega,\mathbf{x}} \|\psi(\omega,\mathbf{x})\|$. We say that ψ is *C-bounded* if $\|\psi\| \le C$. We say that an RFS $\psi : \Omega \times \mathbb{S}^{d-1} \to \mathbb{R}^d$ is *factorized* if there is a function $\psi' : \Omega \times \mathbb{S}^{d-1} \to \mathbb{R}$ such that $\psi(\omega,\mathbf{x}) = \psi'(\omega,\mathbf{x})\mathbf{x}$. We note that the NTK RFS is factorized and C-bounded for $C = \|\sigma'\|_\infty$.

Fix a C-bounded RFS ψ for a kernel k. A random *q-embedding* generated from ψ is the random mapping $\Psi_{\boldsymbol{\omega}}(\mathbf{x}) := \frac{(\psi(\omega_1,\mathbf{x}),\ldots,\psi(\omega_q,\mathbf{x}))}{\sqrt{q}}$, where $\omega_1,\ldots,\omega_q \sim \mu$ are i.i.d. The random *q-kernel* corresponding to $\Psi_{\boldsymbol{\omega}}$ is $k_{\boldsymbol{\omega}}(\mathbf{x},\mathbf{x}') = \langle \Psi_{\boldsymbol{\omega}}(\mathbf{x}), \Psi_{\boldsymbol{\omega}}(\mathbf{x}')\rangle$. Likewise, the random *$q$-kernel space* corresponding to $\Psi_{\boldsymbol{\omega}}$ is $\mathcal{H}_{k_{\boldsymbol{\omega}}}$. We note that in the case of the NTK RFS, a random q-embedding is, up to scaling, the gradient of a randomly initialized network. Likewise, tk^h_W is a random q-kernel generated from the NTK RFS.

It would be useful to consider the embedding

$$\mathbf{x} \mapsto \Psi^{\mathbf{x}} \text{ where } \Psi^{\mathbf{x}} := \psi(\cdot,\mathbf{x}) \in L^2(\Omega,\mathbb{R}^d)\,. \tag{4}$$

From (3) it holds that for any $\mathbf{x},\mathbf{x}' \in \mathcal{X}$, $k(\mathbf{x},\mathbf{x}') = \left\langle \Psi^{\mathbf{x}}, \Psi^{\mathbf{x}'}\right\rangle_{L^2(\Omega)}$. In particular, from Theorem 2, for every $f \in \mathcal{H}_k$ there is a unique function $\check{f} \in L^2(\Omega,\mathbb{R}^d)$ such that

$$\|\check{f}\|_{L^2(\Omega)} = \|f\|_k \tag{5}$$

and for every $\mathbf{x} \in \mathcal{X}$,

$$f(\mathbf{x}) = \langle \check{f}, \Psi^{\mathbf{x}} \rangle_{L^2(\Omega,\mathbb{R}^d)} = \mathop{\mathbb{E}}_{\omega\sim\mu} \langle \check{f}(\omega), \psi(\omega,\mathbf{x}) \rangle \ . \tag{6}$$

Let us denote $f_{\boldsymbol{\omega}}(\mathbf{x}) = \frac{1}{q}\sum_{i=1}^{q} \langle \check{f}(\omega_i), \psi(\omega_i,\mathbf{x}) \rangle$. From (6) we have that $\mathbb{E}_{\boldsymbol{\omega}}[f_{\boldsymbol{\omega}}(\mathbf{x})] = f(\mathbf{x})$. Furthermore, for every $\mathbf{x}$, the variance of $f_{\boldsymbol{\omega}}(\mathbf{x})$ is at most

$$\frac{1}{q}\mathop{\mathbb{E}}_{\omega\sim\mu} \left| \langle \check{f}(\omega), \psi(\omega,\mathbf{x}) \rangle \right|^2 \le \frac{C^2}{q}\mathop{\mathbb{E}}_{\omega\sim\mu} \left|\check{f}(\omega)\right|^2 = \frac{C^2\|f\|_k^2}{q} \ .$$

An immediate consequence is the following corollary.

Corollary 3 (Function Approximation)**.** *For all* $\mathbf{x} \in \mathcal{X}$*,* $\mathbb{E}_{\boldsymbol{\omega}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})|^2 \le \frac{C^2\|f\|_k^2}{q}$.

Now, if $\mathcal{D}$ is a distribution on $\mathcal{X}$ we get that

$$\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \|f - f_{\boldsymbol{\omega}}\|_{2,\mathcal{D}} \overset{\text{Jensen}}{\le} \sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \|f - f_{\boldsymbol{\omega}}\|_{2,\mathcal{D}}^2} = \sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\omega}}\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})|^2} = \sqrt{\mathop{\mathbb{E}}_{\mathbf{x}}\mathop{\mathbb{E}}_{\boldsymbol{\omega}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})|^2} \le \frac{C\|f\|_k}{\sqrt{q}}$$

Using the above inequality, it is possible to show that (see theorem 10 below) SGD on top of a random q-embedding, using a convex and Lipschitz loss, is guaranteed to find a function $\hat{f}$ that satisfies $\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(\hat{f}) \le \mathcal{L}_{\mathcal{D}}(f^*) + O\left(\frac{\|f^*\|_k}{\sqrt{q}}\right)$ for any $f^* \in \mathcal{H}_k$.

3 Results

We next present our results in detail. Due to lack of space, all proofs are differed to the appendix.

3.1 Verctor RFS and NTK Convergence

Fix a C-bounded RFS $\psi : \Omega \times \mathcal{X} \to \mathbb{R}^d$ for a kernel k. Corollary 3 implies that $O\left(\frac{\|f\|_k^2}{\epsilon^2}\right)$ random features suffices to guarantee that for every $f \in \mathcal{H}_k$, in expectation, the empirical kernel space will contain an ϵ approximation of f. This bound does not depend on d, the dimension of a single random feature. We might expect that at least in some cases, d-dimensional random feature is as good as d one-dimensional random features. The next result show that for factorized RFS and $O(1)$-bounded distributions this is indeed the case and $O\left(\frac{\|f\|_k^2}{d\epsilon^2}\right)$ random features suffices to guarantee ϵ-approximation.

Theorem 4. *Assume that* $\psi : \Omega \times \mathbb{S}^{d-1} \to \mathbb{R}^d$ *is factorized and* $\mathcal{D}$ *is* R*-bounded distribution. Then,*

$$\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \|f - f_{\boldsymbol{\omega}}\|_{2,\mathcal{D}} \le \sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \|f - f_{\boldsymbol{\omega}}\|_{2,\mathcal{D}}^2} \le \frac{RC\|f\|_k}{\sqrt{qd}}$$

Furthermore, if $\ell : \mathbb{S}^{d-1} \times Y \to [0,\infty)$*, is* L*-Lipschitz loss and* $\mathcal{D}'$ *is a distribution of* $\mathbb{S}^{d-1} \times Y$ *with* R*-bounded marginal then* $\mathbb{E}_{\boldsymbol{\omega}}\,\mathcal{L}_{\mathcal{D}'}(f_{\boldsymbol{\omega}}) \le \mathcal{L}_{\mathcal{D}'}(f) + \frac{LRC\|f\|_k}{\sqrt{qd}}$

Using the above inequality, it is possible to show that (see theorem 10 below) SGD on top of a random q-embedding, using a convex and Lipschitz loss, is guaranteed to find a function $\hat{f}$ that satisfies $\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(\hat{f}) \le \mathcal{L}_{\mathcal{D}}(f^*) + O\left(\frac{\|f^*\|_k}{\sqrt{qd}}\right)$ for any $f^* \in \mathcal{H}_k$. Applying this to the NTK RFS, and via further reduction to neural network learning, we can show that a similar guarantee is valid for algorithm 1. This is described in the next section.

3.2 Learning the neural tangent kernel space with SGD on NN

Fix a decent activation function σ and a decent loss ℓ. We shows that algorithm 1 can learn the class $\mathcal{H}^M_{\mathrm{tk}^h_\sigma}$ using a network with $\tilde{O}\left(\frac{M^2}{\epsilon^2}\right)$ parameters and using $O\left(\frac{M^2}{\epsilon^2}\right)$ examples. We note that unless σ is linear, the number of samples is optimal up to constant factor, and the number of parameters is optimal, up to poly-log factor and the dependency on ϵ. This remains true even if we restrict to $O(1)$-bounded distributions.

Theorem 5. *Given d, $M > 0$, $R > 0$ and $\epsilon > 0$ there is a choice of $q = \tilde{O}\left(\frac{M^2R^2}{d\epsilon^2}\right)$, $T = O\left(\frac{M^2}{\epsilon^2}\right)$, as well as $B > 0$ and $\eta > 0$, such that for every R-bounded distribution $\mathcal{D}$ and batch size b, the function h returned by algorithm 1 satisfies $\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(h) \leq \mathcal{L}_{\mathcal{D}}\left(\mathcal{H}^M_{\mathrm{tk}^h_\sigma}\right) + \epsilon$*

As an application, we conclude that for the ReLU activation, algorithm 1 can learn even polynomials of bounded norm with near optimal sample complexity and network size. We denote

$$\mathcal{P}^M_c = \left\{ p(\mathbf{x}) = \sum_{|\alpha| \text{ is even and } \leq c} a_\alpha \mathbf{x}^\alpha : \sum_{|\alpha| \text{ is even and } \leq c} a_\alpha^2 \leq M^2 \right\}$$

For the ReLU activation σ, it holds that for every constant c, $\mathcal{P}^M_c \subset \mathcal{H}^{O(M)}_{\mathrm{tk}^h_\sigma}$ (e.g. [9]). Theorem 5 therefore implies that

Theorem 6. *Fix a constant $c > 0$ and assume that the activation is ReLU. Given d, $M > 0$, $R > 0$ and $\epsilon > 0$ there is a choice of $q = \tilde{O}\left(\frac{M^2R^2}{d\epsilon^2}\right)$, $T = O\left(\frac{M^2}{\epsilon^2}\right)$, as well as $B > 0$ and $\eta > 0$, such that for every R-bounded distribution $\mathcal{D}$ and batch size b, the function h returned by algorithm 1 satisfies $\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(h) \leq \mathcal{L}_{\mathcal{D}}\left(\mathcal{P}^M_c\right) + \epsilon$*

We note that as in theorem 5, the number of samples is optimal up to constant factor, and the number of parameters is optimal, up to poly-log factor and the dependency on ϵ, and this remains true even if we restrict to $O(1)$-bounded distributions.

3.3 Memorization

Theorem 5 can be applied to analyze memorization by SGD. Assume that ℓ is the hinge loss (similar result is valid for many other losses such as the log-loss) and σ is any decent non-linear activation. Let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be m random, independent and uniform points in $\mathbb{S}^{d-1} \times \{\pm 1\}$ with $m = d^c$ for some $c > 1$. Suppose that we run SGD on top of S. Namely, we run algorithm 1 where the underlying distribution is the uniform distribution on the points in S. Let $h : \mathbb{S}^{d-1} \to \mathbb{R}$ be the output of the algorithm. We say that the algorithm memorized the i'th example if $y_i h(\mathbf{x}_i) > 0$. The memorization problem investigate how many points the algorithm can memorize, were most of the focus is on how large the network should be in order to memorize $1 - \epsilon$ fraction of the points.

As shown in section A.5, the uniform distribution on the examples in S is $(1 + o(1))$-bounded w.h.p. over the choice of S. Likewise, it is not hard to show that w.h.p. over the choice of S there is a function $h^* \in \mathcal{H}^{O(m)}_k$ such that $h^*(\mathbf{x}_i) = y_i$ for all i. By theorem 5 we can conclude the by running SGD on a network with $\tilde{O}\left(\frac{m}{\epsilon^2}\right)$ parameters and $O\left(\frac{m}{\epsilon^2}\right)$ steps, the network will memorize $1 - \epsilon$ fraction of the points. This size of networks is optimal up to poly-log factors, and the dependency of ϵ. This is satisfactory is ϵ is considered a constant. However, for small ϵ, more can be desired. For instance, in the case that we want to memorize all points, we need $\epsilon < \frac{1}{m}$, and we get a network with m^3 parameters. To circumvent that, we perform a more refined analysis of this memorization problem and show that even perfect memorization of m points can be done via SGD on a network with $\tilde{O}(m)$ parameters, which is optimal, up to poly-log factors.

Theorem 7. *There is a choice of $q = \tilde{O}\left(\frac{m}{d}\right)$, $T = \tilde{O}\left(\frac{m}{\epsilon^2}\right)$, as well as $B > 0$ and $\eta > 0$, such that for every batch size b, w.p. $1 - o_m(1)$, the function h returned by algorithm 1 memorizes $1 - \epsilon$ fraction of the examples.*

We emphasize the our result is true for any non-linear and decent activation function.

3.4 Open Questions

The most obvious open question is to generalize our results to the standard Xavier initialization, where W is a matrix of independent standeard Gaussians, while $\mathbf{u}$ is a vector of independent centered Gaussians of variance $\frac{1}{q}$. Another open question is to generalize our result to deeper networks.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. What can resnet learn efficiently, going beyond kernels? *arXiv preprint arXiv:1905.10337*, 2019.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

[3] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning polynomials with neural networks. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1908–1916, 2014.

[4] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

[5] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. In *ICLR*, 2018.

[6] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *arXiv preprint arXiv:1905.13210*, 2019.

[7] Amit Daniely. Sgd learns the conjugate class of the netwirk. In *NIPS*, 2017.

[8] Amit Daniely and Eran Malach. Learning parities with neural networks. *arXiv preprint arXiv:2002.07400*, 2020.

[9] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *NIPS*, 2016.

[10] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

[11] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

[12] Jonathan Fiat, Eran Malach, and Shai Shalev-Shwartz. Decoupling gating from linearity. *arXiv preprint arXiv:1906.05032*, 2019.

[13] Rong Ge, Runzhe Wang, and Haoyu Zhao. Mildly overparametrized neural nets can memorize training data efficiently. *arXiv preprint arXiv:1909.11837*, 2019.

[14] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 9108–9118, 2019.

[15] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[16] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[17] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*, 2019.

[18] Chao Ma, Lei Wu, et al. A comparative analysis of the optimization and generalization property of two-layer neural network and random feature models under gradient descent dynamics. *arXiv preprint arXiv:1904.04326*, 2019.

[19] Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *arXiv preprint arXiv:1902.04674*, 2019.

[20] Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *arXiv:1902.04674 [cs, math, stat]*, February 2019. URL `http://arxiv.org/abs/1902.04674`. arXiv: 1902.04674.

[21] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

[22] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[23] Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv preprint arXiv:1906.03593*, 2019.

[24] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027 v7*, 2010.

[25] C.K.I. Williams. Computation with infinite neural networks. pages 295–301, 1997.

[26] Xiaoxia Wu, Simon S Du, and Rachel Ward. Global convergence of adaptive gradient methods for an over-parameterized neural network. *arXiv preprint arXiv:1902.07111*, 2019.

[27] Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. *arXiv preprint arXiv:1904.00687*, 2019.

[28] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *arXiv preprint arXiv:1906.04688*, 2019.

A Proofs

A.1 More preliminaries: inner product kernels and Hermite polynomials

A special type of kernels that we will useful for us are *inner product kernels*. These are kernels $k : \mathbb{S}^{d-1} \times \mathbb{S}^{d-1} \to \mathbb{R}$ of the form

$$k(\mathbf{x}, \mathbf{y}) = \sum_{n=0}^{\infty} b_n \langle \mathbf{x}, \mathbf{y} \rangle^n$$

For scalars $b_n \geq 0$ with $\sum_{n=0}^{\infty} b_n < \infty$. It is well known that for any such sequence k is a kernel. The following lemma summarizes a few properties of inner product kernels.

Lemma 8. *Let k be the inner product kernel $k(\mathbf{x}, \mathbf{y}) = \sum_{n=0}^{\infty} b_n \langle \mathbf{x}, \mathbf{y} \rangle^n$. Suppose that $b_n > 0$*

1. *If $p(\mathbf{x}) = \sum_{|\alpha|=n} a_\alpha \mathbf{x}^\alpha$ then $p \in \mathcal{H}_k$ and furthermore $\|p\|_k^2 \leq \frac{1}{b_n} \sum_{|\alpha|=n} a_\alpha^2$*

2. *For every $\mathbf{u} \in \mathbb{S}^{d-1}$, the function $f(\mathbf{x}) = \langle \mathbf{u}, \mathbf{x} \rangle^n$ belongs to $\mathcal{H}_k$ and $\|f\|_k^2 = \frac{1}{b_n}$*

Hermite polynomials $h_0, h_1, h_2, \ldots$ are the sequence of orthonormal polynomials corresponding to the standard Gaussian measure on $\mathbb{R}$. Fix an activation $\sigma : \mathbb{R} \to \mathbb{R}$. Following the terminology of [9] we define the *dual activation* of σ as

$$\hat{\sigma}(\rho) = \mathop{\mathbb{E}}_{X,Y \text{ are } \rho\text{-correlated standard Gaussian}} \sigma(X)\sigma(Y)$$

It holds that if $\sigma = \sum_{n=0}^{\infty} a_n h_n$ then

$$\hat{\sigma}(\rho) = \sum_{n=0}^{\infty} a_n^2 \rho^n$$

In particular, $k_\sigma(\mathbf{x}, \mathbf{y}) := \hat{\sigma}(\langle \mathbf{x}, \mathbf{y} \rangle)$ is an inner product kernel.

A.2 Vector random feature schemes

For the rest of this section, let us fix a C-bounded RFS ψ for a kernel k and a random q embedding $\Psi_{\boldsymbol{\omega}}$. For every $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

$$k_{\boldsymbol{\omega}}(\mathbf{x}, \mathbf{x}') = \frac{1}{q} \sum_{i=1}^{q} \langle \psi(\omega_i, \mathbf{x}), \psi(\omega_i, \mathbf{x}') \rangle$$

is an average of q independent random variables whose expectation is $k(\mathbf{x}, \mathbf{x}')$. By Hoeffding's bound we have.

Theorem 9 (Kernel Approximation)**.** *Assume that $q \geq \frac{2C^4 \log\left(\frac{2}{\delta}\right)}{\epsilon^2}$, then for every $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ we have $\Pr\left(|k_{\boldsymbol{\omega}}(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{x}')| \geq \epsilon\right) \leq \delta$.*

We next discuss approximation of functions in $\mathcal{H}_k$ by functions in $\mathcal{H}_{k_{\boldsymbol{\omega}}}$, and prove theorem 3

Proof. (of theorem 4) Let $\mathbf{x} \sim \mathcal{D}$ and $\omega \sim \mu$. We have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \|f - f_{\boldsymbol{\omega}}\|_{2,\mathcal{D}} &\overset{\text{Jensen's Inequality}}{\leq} \sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \|f - f_{\boldsymbol{\omega}}\|_{2,\mathcal{D}}^2} \\
&= \sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\mathbf{x}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})|^2} \\
&= \sqrt{\mathop{\mathbb{E}}_{\mathbf{x}} \mathop{\mathbb{E}}_{\boldsymbol{\omega}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})|^2} \\
&= \sqrt{\frac{\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\omega\sim\mu} \left|\langle \check{f}(\omega), \psi(\omega, \mathbf{x})\rangle - f(\mathbf{x})\right|^2}{q}} \\
&\overset{\text{Variance is bounded by squared } L^2 \text{ norm}}{\leq} \sqrt{\frac{\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\omega\sim\mu} \left|\langle \check{f}(\omega), \psi(\omega, \mathbf{x})\rangle\right|^2}{q}} \\
&= \sqrt{\frac{\mathbb{E}_{\omega\sim\mu} \mathbb{E}_{\mathbf{x}} \left|\langle \check{f}(\omega), \psi'(\omega, \mathbf{x})\mathbf{x}\rangle\right|^2}{q}} \\
&\overset{\psi \text{ and hence also } \psi' \text{ is } C\text{-bounded}}{\leq} C\sqrt{\frac{\mathbb{E}_{\omega\sim\mu} \mathbb{E}_{\mathbf{x}} \left|\langle \check{f}(\omega), \mathbf{x}\rangle\right|^2}{q}} \\
&\overset{\mathcal{D} \text{ is } R\text{-bounded}}{\leq} CR\sqrt{\frac{\mathbb{E}_{\omega\sim\mu} \left\|\check{f}(\omega)\right\|^2}{qd}} \\
&\overset{\text{Equation (5)}}{=} \frac{CR\|f\|_k}{\sqrt{qd}} .
\end{aligned}$$

Finally, for L-Lipschitz ℓ, and $(\mathbf{x}, y) \sim \mathcal{D}'$ then

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\omega}} L_{\mathcal{D}'}(f_{\boldsymbol{\omega}}) &= \mathop{\mathbb{E}}_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\mathbf{x},y} \ell(f_{\boldsymbol{\omega}}(\mathbf{x}), y) \\
&\leq \mathop{\mathbb{E}}_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\mathbf{x},y} \ell(f(\mathbf{x}), y) + L \mathop{\mathbb{E}}_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\mathbf{x}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})| \\
&= \mathop{\mathbb{E}}_{\mathbf{x},y} \ell(f(\mathbf{x}), y) + L \mathop{\mathbb{E}}_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\mathbf{x}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})| \\
&= \mathcal{L}_{\mathcal{D}'}(f) + L \mathop{\mathbb{E}}_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\mathbf{x}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})| \\
&\overset{L^1 \leq L^2}{\leq} \mathcal{L}_{\mathcal{D}'}(f) + L \mathop{\mathbb{E}}_{\boldsymbol{\omega}} \sqrt{\mathop{\mathbb{E}}_{\mathbf{x}} |f(\mathbf{x}) - f_{\boldsymbol{\omega}}(\mathbf{x})|^2} \\
&\overset{\text{first part of the lemma}}{\leq} \mathcal{L}_{\mathcal{D}'}(f) + \frac{LCR\|f\|_k}{\sqrt{qd}}
\end{aligned}$$

□

We next consider an algorithm for learning $\mathcal{H}_k$, by running SGD on top of random features.

Algorithm 2 SGD on RFS

```
**Input:** RFS $\psi : \Omega \times \mathcal{X} \to \mathbb{R}^d$, number of random features $q$, loss $\ell$, learning rate $\eta > 0$, batch size $b$, number of steps $T > 0$, access to samples from a distribution $\mathcal{D}$
Sample $\boldsymbol{\omega} \sim \mu^q$
Initialize $\mathbf{v}^1 = 0 \in \mathbb{R}^{q \times d}$
for $t = 1, \ldots, T$ do
    Obtain a mini-batch $S_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^b \sim \mathcal{D}^b$
    Update $\mathbf{v}_{t+1} = \mathbf{v}_t - \eta\nabla\mathcal{L}_{S_t}(\mathbf{v}_t)$ where $\mathcal{L}_{S_t}(\mathbf{v}) = \mathcal{L}_{S_t}(f_{\Psi_{\boldsymbol{\omega}},\mathbf{v}})$.
end for
Choose $t \in [T]$ uniformly at random and return $f_{\Psi_{\boldsymbol{\omega}},\mathbf{v}_t}$
```

Theorem 10. *Assume that ψ is factorized and C-bounded RFS for k, that ℓ is convex and L-Lipschitz, and that $\mathcal{D}$ has R-bounded marginal. Let f be the function returned by algorithm 2. Fix a function*

$f^* \in \mathcal{H}_k$. Then

$$\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(f) \le \mathcal{L}_{\mathcal{D}}(f^*) + \frac{LRC\|f^*\|_k}{\sqrt{qd}} + \frac{\|f^*\|_k^2}{2\eta T} + \frac{\eta L^2 C^2}{2}$$

In particular, if $\|f^*\|_k \le M$ and $\eta = \frac{M}{\sqrt{T}LC}$ we have

$$\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(f) \le L_{\mathcal{D}}(f^*) + \frac{LRCM}{\sqrt{qd}} + \frac{LCM}{\sqrt{T}}$$

Proof. Denote by $\mathbf{v}^* \in \mathbb{R}^{dq}$ the vector

$$v_i^* = \frac{1}{\sqrt{q}}\left(\check{f}^*(\omega_1), \ldots, \check{f}^*(\omega_1)\right)$$

By standard results on SGD (e.g. [22]) we have that given ω,

$$\mathcal{L}_{\mathcal{D}}(f) \quad \le \quad \mathcal{L}_{\mathcal{D}}(f^*_{\boldsymbol{\omega}}) + \frac{1}{2\eta T}\|\mathbf{v}^*\|^2 + \frac{\eta L^2 C^2}{2}$$

Taking expectation over the choice of $\boldsymbol{\omega}$ and using theorem 4 and equation (5) we have

$$\mathcal{L}_{\mathcal{D}}(f) \quad \le \quad \mathcal{L}_{\mathcal{D}}(f^*) + \frac{LRC\|f^*\|_k}{\sqrt{qd}} + \frac{\|f^*\|_k^2}{2\eta T} + \frac{\eta L^2 C^2}{2}$$

□

We conclude the section with a few calculations of $\check{f}$, that will be useful later.

Example 11. Fix $\sigma : \mathbb{R} \to \mathbb{R}$ with Hermite expansion $\sigma = \sum_{n=0}^{\infty} a_n h_n$ and let $\Omega = \mathbb{R}^d$ and $\mathcal{X} = \mathbb{S}^{d-1}$

1. Consider the RFS $\psi(\omega, \mathbf{x}) = \sigma(\langle \omega, \mathbf{x}\rangle)$ with μ being the standard Gaussian measure on $\mathbb{R}^d$. We have that ψ is an RFS for the kernel $k(\mathbf{x}, \mathbf{y}) = \hat{\sigma}(\langle \mathbf{x}, \mathbf{y}\rangle)$. Consider the function $f(\mathbf{x}) = \langle \mathbf{x}_0, \mathbf{x}\rangle^n$. We claim that $\check{f}(\boldsymbol{\omega}) = \frac{1}{a_n} h_n(\langle \mathbf{x}_0, \omega\rangle)$. Indeed, we have,

$$\begin{aligned}\mathop{\mathbb{E}}_{\omega\sim\mu} \sigma(\langle \omega, \mathbf{x}\rangle)\frac{1}{a_n} h_n(\langle \mathbf{x}_0, \omega\rangle) &= \frac{1}{a_n}\sum_{k=0}^{\infty} \mathop{\mathbb{E}}_{\omega\sim\mu} a_k h_k(\langle \omega, \mathbf{x}\rangle) h_n(\langle \mathbf{x}_0, \boldsymbol{\omega}\rangle) \\ &= \frac{1}{a_n}\sum_{k=0}^{\infty} a_k \delta_{kn} \langle \mathbf{x}, \mathbf{x}_0\rangle^k \\ &= \langle \mathbf{x}, \mathbf{x}_0\rangle^n\end{aligned}$$

   and

$$\left\|\omega \mapsto \frac{1}{a_n} h_n(\langle \mathbf{x}_0, \omega\rangle)\right\|_{L^2(\Omega)} = \mathop{\mathbb{E}}_{\omega\sim\mu} \frac{1}{a_n^2} h_n^2(\langle \mathbf{x}_0, \omega\rangle) = \frac{1}{a_n^2} = \|f\|_k^2$$

2. Consider the NTK RFS $\psi(\omega, \mathbf{x}) = \sigma(\langle \omega, \mathbf{x}\rangle)\,\mathbf{x}$ with μ being the standard Gaussian measure on $\mathbb{R}^d$. We have that ψ is an RFS for the kernel $k(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle\,\hat{\sigma}(\langle \mathbf{x}, \mathbf{y}\rangle)$. Consider the function $f(\mathbf{x}) = (\langle \mathbf{x}_0, \mathbf{x}\rangle)^n$. As in the item above, it is not hard to show that $\check{f}(\omega) = \frac{1}{a_{n-1}} h_{n-1}(\langle \mathbf{x}_0, \omega\rangle)\,\mathbf{x}_0$.

A.3 Reduction of NN learning to SGD over vector random features

We will prove our result via a reduction to linear learning over the initial neural tangent kernel space, corresponding the the hidden weights.

That is, we define by $\Psi_{\mathbf{W}}(\mathbf{x})$ the gradient of the function $\mathbf{W} \mapsto h_{\mathbf{W}}(\mathbf{x})$ w.r.t. the hidden weights. Namely,

$$\Psi_{\mathbf{W}}(\mathbf{x}) = (u_1\sigma'(\langle \mathbf{w}_1, \mathbf{x}\rangle)\mathbf{x}, \ldots, u_{2q}\sigma'(\langle \mathbf{w}_{2q}, \mathbf{x}\rangle)\mathbf{x}) \in \mathbb{R}^{2q\times d}$$

Denote $f_{\Psi_{\mathbf{W}}, \mathbf{V}}(\mathbf{x}) = \langle \mathbf{V}, \Psi_{\mathbf{W}}(\mathbf{x})\rangle$ and consider algorithm 3.

It is not hard to show that by taking large enough B, algorithm 1 is essentially equivalent to algorithm 3. Namely,

Algorithm 3 Neural Tangent Kernel Training

```
Input: Network parameters $\sigma$ and $d, q$, loss $\ell$, learning rate $\eta > 0$, batch size $b$, number of steps
$T > 0$, access to samples from a distribution $\mathcal{D}$
Sample $\mathbf{W} \sim \mathcal{I}(d, q, 1)$
Initialize $\mathbf{V}^1 = 0 \in \mathbb{R}^{2q \times d}$
for $t = 1, \dots, T$ do
    Obtain a mini-batch $S_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^b \sim \mathcal{D}^b$
    Using back-propagation, calculate the gradient $\nabla$ of $\mathcal{L}_{S_t}(\mathbf{V}) = \mathcal{L}_{S_t}(f_{\Psi_{\mathbf{W}}, \mathbf{V}})$ at $\mathbf{V}^t$
    Update $\mathbf{V}^{t+1} = \mathbf{V}^t - \eta\nabla$
end for
Choose $t \in [T]$ uniformly at random and return $f_{\Psi_W, \mathbf{V}_t}$
```

Lemma 12. *Fix a decent activation σ as well as convex a decent loss ℓ. There is a choice $B = poly(d, q, 1/\eta, T, 1/\epsilon)$, such that for every input distribution the following holds. Let h_1, h_2 be the functions returned algorithm 1 with parameters $d, q, \frac{\eta}{B^2}, b, B, T$ and algorithm 3 with parameters d, q, η, b, T. Then, $|\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(h_1) - \mathbb{E}\,\mathcal{L}_{\mathcal{D}}(h_2)| < \epsilon$*

Proof. (sketch) For simplicity, instead of assuming that σ is M-decent, we assume that the activation is twice differentiable and satisfies $\|\sigma'\|_\infty, \|\sigma''\|_\infty < M$. At the end of the sketch we will later explain how to handle M-decent activation.

Consider a run of algorithm 1 starting from the initial weights $\mathbf{W} = (W, \mathbf{u})$ in the support of $\mathcal{I}(d, q, 1)$. Consider now another run, running on the same mini-batches, hyper-parameters and initial weights, except that in the second run the output weight are multiplied by B, and the learning rate is multiplied by $\frac{1}{B^2}$. Our goal is to show that for large B, the second run approximates algorithm 3, with the approximation becoming better as B gets larger.

The process of multiplying the output weights by B cause the gradient, $\nabla_W h_{\mathbf{W}}(x)$, of the hidden layer to be multiplied by B, and the gradient, $\nabla_{\mathbf{u}} h_{\mathbf{W}}(x)$, of the output layer to remain the same. Thus, for large enough B, we can use this observation in order to ignore the gradient of the output weights. We therefore assume that algorithm 1 only updates the hidden weight. Likewise, while the gradient is multiplied by B, the step size is multiplied by $\frac{1}{B^2}$. Hence, the total movement is multiplied by $\frac{1}{B}$. It therefore holds that the optimization process takes place in a ball of radius $\frac{R}{B}$ around W, where $R = poly(M, d, q, 1/\eta, T, 1/\epsilon)$ does not depend on B.

Now by multiplying the output weights by B, we move from the network function $h_W(x)$ to $\tilde{h}_W(x) := Bh_W(x)$. The first order approximation of $\tilde{h}$ around the initial weights is

$$\tilde{h}_{W+V}(x) = Bh_W(x) + B\langle\nabla_W h_W(x), V\rangle + \frac{H}{2}\|V\|^2 = B\langle\nabla_W h_W(x), V\rangle + \frac{H}{2}\|V\|^2$$

Where H is a uniform bound on the Hessian of $h_W(x)$ (such a bound exists since $\|\sigma'\|_\infty, \|\sigma''\|_\infty < M$). Now, since the optimization in a ball of radius $\frac{R}{B}$ around W, we can ignore the quadratic part for large enough B, and reduce to the case of optimization over the linear function $B\langle\nabla_W h_W(x), V\rangle$ with learning rate of $\frac{\eta}{B^2}$ starting at 0. This is equivalent to optimization over the linear function $\langle\nabla_W h(W, x), V\rangle$ with learning rate of η starting at 0, which is exactly algorithm 3.

Finally, to handle general M-decent activation, we note that any such activation locally satisfies, $\|\sigma'\|_\infty, \|\sigma''\|_\infty < M$. Now, for large enough B, the output of the hidden layer, before the activation, barely moves throughout the optimization process, and hence, for each example in the min-batches, we don't move between different regions in which σ satisfies $\|\sigma'\|_\infty, \|\sigma''\|_\infty < M$.

$\square$

By lemma 11 in order to prove theorem 5 it is enough to analyze algorithm 3. Specifically, theorem 5 follows form the following theorem:

Theorem 13. *Given d, $M > 0$, $R > 0$ and $\epsilon > 0$ there is a choice of $q = \tilde{O}\left(\frac{M^2R^2}{d\epsilon^2}\right)$, $T = O\left(\frac{M^2}{\epsilon^2}\right)$, as well as $\eta > 0$, such that for every R-bounded distribution $\mathcal{D}$ and batch size b, the function h returned by algorithm 3 satisfies $\mathbb{E}\,\mathcal{L}_{\mathcal{D}}(h) \le \mathcal{L}_{\mathcal{D}}\left(\mathcal{H}^M_{\mathrm{tk}^h_\sigma}\right) + \epsilon$*

Our next step is to rephrase algorithm 3 in the language of (vector) random features. We note that algorithm 3 is SGD on top of the random embedding $\Psi_{\mathbf{W}}$. This embedding composed of q i.i.d. random mappings $\psi_{\mathbf{w}}(\mathbf{x}) = (\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle)\mathbf{x}, -\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle)\mathbf{x})$ where $\mathbf{w} \in \mathbb{R}^d$ is a standard Gaussian. This can be slightly simplified to SGD on top of the i.i.d. random mappings $\psi_{\mathbf{w}}(\mathbf{x}) = \sigma'(\langle \mathbf{w}, \mathbf{x}\rangle)\mathbf{x}$. Indeed, if we make this change the inner products between the different examples, after the mapping is applied, do not change (up to multiplication by $\sqrt{2}$), and SGD only depends on these inner products. This falls in the framework of learning with (vector) random features scheme, which we define next, and analyze in the next section.

We note that since the NTK RFS is factorized and C-bounded (for $C = \|\sigma'\|_\infty$), theorem 12 follows from theorem 10. Together with lemma 11, this implies theorem 5.

A.4 Memorization of random set of points – proof of theorem 7

Consider the NTK RFS $\psi(\omega, \mathbf{x}) = \sigma'\left(\langle \omega, \mathbf{x}\rangle\right)\mathbf{x}$ with μ being the standard Gaussian measure on $\mathbb{R}^d$. Recall that ψ is an RFS for the kernel $\mathrm{tk}^h_\sigma(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle\, \hat{\sigma}'\left(\langle \mathbf{x}, \mathbf{y}\rangle\right)$. As in the proof of theorem 5, it is enough to show that for $q = \tilde{O}\left(\frac{m}{d}\right) = \tilde{O}\left(d^{c-1}\right)$, w.p. $1 - o(1)$ over the choice of S and $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_q)$, there is $\mathbf{v} \in \mathbb{R}^{dq}$ such that

$$\langle \mathbf{v}, \Psi_{\boldsymbol{\omega}}(\mathbf{x}_i)\rangle = y_i + o(1) \text{ for all } i \text{ and } \|\mathbf{v}\|_2^2 = \tilde{O}(m) \tag{7}$$

Choose a constant integer $c' > 4c + 2$ such that $a_{c'-1} \neq 0$. Such a constant exists since σ is not a polynomial. Define

$$f(\mathbf{x}) = \sum_{i=1}^m y_i \left(\langle \mathbf{x}_i, \mathbf{x}\rangle\right)^{c'}$$

Lemma 14. *With probability* $1 - \delta$ *we have that*

$$f(\mathbf{x}_i) = y_i + O\left(\frac{\log^{\frac{c'}{2}}(d/\delta)}{d}\right) \textit{ for all } i \textit{ and } \|f\|^2_{k_\sigma} = O\left(m\right) + O\left(\frac{\log^{\frac{c'}{2}}(d/\delta)}{d}\right)$$

Proof. W.p $1 - \delta$ we have that $\langle \mathbf{x}_i, \mathbf{x}_j\rangle \le O\left(\sqrt{\frac{\log(m/\delta)}{d}}\right) = O\left(\sqrt{\frac{\log(d/\delta)}{d}}\right)$ for all $i, j \in [m]$. In this case we have that for any i

$$f(\mathbf{x}_i) = y_i + O\left(m\left(\frac{\log\left(d/\delta\right)}{d}\right)^{\frac{c'}{2}}\right) = y_i + O\left(\log^{\frac{c'}{2}}\left(d/\delta\right) d^{c - \frac{c'}{2}}\right) = y_i + O\left(\frac{\log^{\frac{c'}{2}}(d/\delta)}{d}\right)$$

Likewise,

$$\|f\|^2_{k_\sigma} = a_{c'}^{-2} m + O\left(m^2\left(\frac{\log\left(d/\delta\right)}{d}\right)^{\frac{c'}{2}}\right) = a_{c'}^{-2} m + O\left(\log^{\frac{c'}{2}}\left(d/\delta\right) d^{2c - \frac{c'}{2}}\right) = a_{c'}^{-2} m + O\left(\frac{\log^{\frac{c'}{2}}(d/\delta)}{d}\right)$$

□

Based on lemma 13, in order to find $\mathbf{v}$ that satisfies equation (7) it is natural to take

$$\mathbf{v} = \frac{1}{\sqrt{q}}\left(\check{f}(\omega_1), \ldots, \check{f}(\omega_q)\right)$$

In which case $\mathbb{E}\,\|\mathbf{v}\|_2^2 = \|f\|^2_{k_\sigma}$ and $\mathbb{E}\left[\langle \mathbf{v}, \Psi_{\boldsymbol{\omega}}(\mathbf{x})\rangle\right] = \mathbb{E}\left[f_{\boldsymbol{\omega}}(\mathbf{x})\right] = f(\mathbf{x})$. In fact, theorem 4 together with Chebyshev's inequality indeed implies that for large q equation (7) holds. However, this analysis requires $q \approx \frac{m^2}{d}$ while we want $q \approx \frac{m}{d}$. In the remaining part of this section we undertake a more delicate anlysis of the rate in which $f_{\boldsymbol{\omega}}$ approximates f in our specific case. This analysis will imply that $q = \tilde{O}\left(\frac{m}{d}\right)$ suffices for equation (7) to hold w.h.p. Indeed, we will prove that

Lemma 15. *W.p.* $1-\delta-2^{\Omega(d)}$ *over the choice of* S *and* $\boldsymbol{\omega}$*, we have that*

$$\forall i\in[m],\quad |f_{\boldsymbol{\omega}}(\mathbf{x}_i)-f(\mathbf{x}_i)|\le O\left(\sqrt{\frac{m\log^{c'+2}(m/\delta)}{dq}}\right)$$

Togeter with lemma 13 and Markov's inequality we have

Theorem 16. *W.p.* $1-\delta-2^{\Omega(d)}$ *over the choice of* S *and* $\boldsymbol{\omega}$*, we have that*

$$\langle\mathbf{v},\Psi_{\boldsymbol{\omega}}(\mathbf{x}_i)\rangle=f_{\boldsymbol{\omega}}(\mathbf{x}_i)=y_i+O\left(\frac{\log^{\frac{c'}{2}}(d/\delta)}{d}\right)+O\left(\sqrt{\frac{d^{c-1}\log^{c'+2}(d/\delta)}{q}}\right)\text{ for all } i$$

and

$$\|\mathbf{v}\|_2^2=O\left(m/\delta\right)+O\left(\frac{\log^{\frac{c'}{2}}(d/\delta)}{d\delta}\right)$$

Choosing $\delta=\frac{1}{\log(m)}$ we get that for $q=\tilde{O}\left(d^{c-1}\right)$ equation (7) holds w.p. $1-o(1)$. This proves theorem 7. The remaining part of the section is a proof of lemma 14. We will need the following version of Hoeffding's bound. A distribution μ on $\mathbb{R}$ is called (δ,B)*-bounded* if $\Pr_{X\sim\mu}(|X|>B)\le\delta$.

Lemma 17. *Let* μ *be a* (δ,B)*-bounded distribution and let* $X_1,\ldots,X_m$ *be i.i.d. r.v. from* μ*. Then, w.p.* $1-m\delta-\delta'$

$$\left|\mathop{\mathbb{E}}_{X\sim\mu}[X]-\frac{1}{m}\sum_{i=1}^m X_i\right|\le B\sqrt{\frac{2\ln(\delta'/2)}{m}}+\frac{2\sqrt{\delta\,\mathbb{E}_{X\sim\mu}X^2}}{1-\delta}$$

Proof. We note that given that $X_i\in[-B,B]$ for all i we have by Hoeffding's bound that w.p. $1-\delta'$

$$\left|\frac{1}{m}\sum_{i=1}^m X_i-\mathop{\mathbb{E}}_{X\sim\mu}[X|X\in[-B,B]]\right|\le B\sqrt{\frac{2\ln(\delta'/2)}{m}}$$

We note that

$$\begin{aligned}\mathop{\mathbb{E}}_{X\sim\mu}[X|X\in[-B,B]] &= \frac{\mathbb{E}_{X\sim\mu}X+\delta\,\mathbb{E}_{X\sim\mu}[X|X\notin[-B,B]]}{1-\delta}\\ &= \frac{\mathbb{E}_{X\sim\mu}X+\mathbb{E}_{X\sim\mu}[X1[X\notin[-B,B]]}{1-\delta}\end{aligned}$$

Hence, by Cauchy-Schwartz,

$$\left|\mathop{\mathbb{E}}_{X\sim\mu}[X|X\in[-B,B]]-\mathop{\mathbb{E}}_{X\sim\mu}[X]\right|\le\frac{\delta}{1-\delta}\left|\mathop{\mathbb{E}}_{X\sim\mu}X\right|+\frac{\sqrt{\delta\,\mathbb{E}_{X\sim\mu}X^2}}{1-\delta}\le\frac{2\sqrt{\delta\,\mathbb{E}_{X\sim\mu}X^2}}{1-\delta}$$

□

Recall now that by example **??**

$$\check{f}(\omega)=\sum_{i=1}^m\frac{y_i}{a_{c'-1}}h_{c'-1}\left(\langle\mathbf{x}_i,\omega\rangle\right)\mathbf{x}_i$$

Hence, for any $\mathbf{x}$,

$$f_{\boldsymbol{\omega}}(\mathbf{x})=\frac{1}{q}\sum_{j=1}^q\sum_{i=1}^m\frac{y_i}{a_{c'-1}}h_{c'-1}\left(\langle\mathbf{x}_i,\omega_j\rangle\right)\langle\mathbf{x}_i,\mathbf{x}\rangle\,\sigma\left(\langle\omega_j,\mathbf{x}\rangle\right)$$

In particular, fixing S, $f_{\boldsymbol{\omega}}(\mathbf{x})$ is an average of the q i.i.d. random variables

$$f_{\boldsymbol{\omega}}(\mathbf{x})=\frac{1}{q}\sum_{j=1}^q Y(\omega_i,\mathbf{x})$$

Where

$$Y(\omega,\mathbf{x})=\sum_{i=1}^m\frac{y_i}{a_{c'-1}}h_{c'-1}\left(\langle\mathbf{x}_i,\omega\rangle\right)\langle\mathbf{x}_i,\mathbf{x}\rangle\,\sigma\left(\langle\omega,\mathbf{x}\rangle\right)$$

Lemma 18. *W.p.* $\geq 1-\delta$ *over the choice of* S*, we have that for every* $i \in [m]$*,* $Y(\omega, \mathbf{x}_i)$ *is* $\left(\delta + 2^{-\Omega(d)}, O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right)\right)$*-bounded.*

Proof. Fix ω with $\|\omega\| \leq 2\sqrt{d}$. We have that $Y(\omega, \mathbf{x}_i)$, as a function of S, is a random variable that is a sum of a single random variable (the summand that corresponds $\mathbf{x}_i$) that is $\left(\delta, O\left(\sqrt{\log^{c'-1}(1/\delta)}\right)\right)$-bounded, as well as $(m-1)$ additional i.i.d random variables that have mean 0, are $\left(\delta, O\left(\sqrt{\frac{\log^{c'}(1/\delta)}{d}}\right)\right)$-bounded, and has second moment $O\left(\frac{1}{d}\right)$. By lemma 16 we have that

$$|Y(\omega, \mathbf{x}_i)| \leq O\left(\sqrt{\frac{m \log^{c'+1}(1/\delta)}{d}}\right) + O\left(\frac{2m\sqrt{\delta/d}}{1-\delta}\right)$$

w.p. $1-(m+1)\delta$. Equivalently,

$$|Y(\omega, \mathbf{x}_i)| \leq O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right) + O\left(\frac{2\sqrt{(m+1)\delta/d}}{1-\delta}\right) = O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right)$$

w.p. $1-\delta$. We have shown that

$$\mathop{\mathbb{E}}_{\omega}\mathop{\mathbb{E}}_{S}\left[1\left[|Y(\omega, \mathbf{x}_i)| \geq O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right) \text{ and } \|\omega\| \leq 2\sqrt{d}\right]\right] \leq \delta$$

Changing the order of summation and using Markov, we get that w.p. $\geq 1-\sqrt{\delta}$ over the choice of S, we have that

$$\Pr_{\omega}\left[|Y(\omega, \mathbf{x}_i)| \geq O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right) \text{ and } \|\omega\| \leq 2\sqrt{d}\right] \leq \sqrt{\delta}$$

Replacing δ with $\sqrt{\delta}$ and using the fact that $\log\left(m/\delta^2\right) \leq 2\log(m/\delta)$ we get that that w.p. $\geq 1-\delta$ over the choice of S, we have that

$$\Pr_{\omega}\left[|Y(\omega, \mathbf{x}_i)| \geq O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right) \text{ and } \|\omega\| \leq 2\sqrt{d}\right] \leq \delta$$

Hence, since $\Pr_{\omega}\left(\|\omega\| > 2\sqrt{d}\right) \leq 2^{-\Omega(d)}$, we conclude that w.p. $\geq 1-\delta$ over the choice of S, $Y(\omega, \mathbf{x}_i)$ is $\left(\delta + 2^{-\Omega(d)}, O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right)\right)$-bounded. Finally, using a union bound, and the fact that $\log\left(m^2/\delta\right) \leq 2\log(m/\delta)$ we conclude that w.p. $\geq 1-\delta$ over the choice of S, we have that for every $i \in [m]$, $Y(\omega, \mathbf{x}_i)$ is $\left(\delta + 2^{-\Omega(d)}, O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right)\right)$-bounded. □

Proof. (of lemma 14) By lemma 17 we conclude that w.p $1-\delta$ over the choice of S, for every i, $f_{\boldsymbol{\omega}}(x_i)$ is an average of q i.i.d. $\left(\delta + 2^{-\Omega(d)}, O\left(\sqrt{\frac{m \log^{c'+1}(m/\delta)}{d}}\right)\right)$-bounded random variables. Furthermore, the second moment of each of these variables is $O(m)$. Using lemma 16 we have that w.p. $1-(m+1)\delta - m2^{-\Omega(d)}$ over the choice of $\boldsymbol{\omega}$,

$$|f_{\boldsymbol{\omega}}(\mathbf{x}_i) - f(\mathbf{x}_i)| \leq O\left(\sqrt{\frac{m \log^{c'+2}(m/\delta)}{dq}}\right)$$

Using the assumption that $m = d^c$ and simple manipulation we get that w.p. $1 - \delta - 2^{-\Omega(d)}$ over the choice of $\boldsymbol{\omega}$,

$$|f_{\boldsymbol{\omega}}(\mathbf{x}_i) - f(\mathbf{x}_i)| \le O\left(\sqrt{\frac{m \log^{c'+2}(m/\delta)}{dq}}\right)$$

$\square$

A.5 Boundness of distributions

Recall that a distribution $\mathcal{D}$ on $\mathbb{S}^{d-1}$ is R-bounded if for every $\mathbf{u} \in \mathbb{S}^{d-1}$, $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \langle \mathbf{u}, \mathbf{x}\rangle^2 \le \frac{R^2}{d}$. We next describe a few examples of 1-bounded and $(1 + o(1))$-bounded distributions.

1. The uniform distribution is 1-bounded. Indeed, for any $\mathbf{u} \in \mathbb{S}^{d-1}$ and uniform $\mathbf{x}$ in $\mathbb{S}^{d-1}$ we have

$$\mathbb{E}_{\mathbf{x}} \langle \mathbf{u}, \mathbf{x}\rangle^2 = \sum_{i,j} \mathbb{E}_{\mathbf{x}} u_i u_j x_i x_j = \sum_i \mathbb{E}_{\mathbf{x}} u_i^2 x_i^2 = \sum_i u_i^2 \mathbb{E}_{\mathbf{x}} x_i^2 = \frac{1}{d}\sum_i u_i^2 = \frac{\|\mathbf{u}\|^2}{d} = \frac{1}{d}$$

2. Similarly, the uniform distribution on the discrete cube $\left\{-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right\}^d$ is 1-bounded. Indeed, for any $\mathbf{u} \in \mathbb{S}^{d-1}$ and uniform $\mathbf{x}$ in $\left\{-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right\}^d$ we have

$$\mathbb{E}_{\mathbf{x}} \langle \mathbf{u}, \mathbf{x}\rangle^2 = \sum_{i,j} \mathbb{E}_{\mathbf{x}} u_i u_j x_i x_j = \sum_i \mathbb{E}_{\mathbf{x}} u_i^2 x_i^2 = \sum_i u_i^2 \mathbb{E}_{\mathbf{x}} x_i^2 = \frac{1}{d}\sum_i u_i^2 = \frac{\|\mathbf{u}\|^2}{d} = \frac{1}{d}$$

3. Let $\mathcal{D}$ be the uniform distribution on the points $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{S}^{d-1}$. Denote by X the $d \times m$ matrix whose i' column is $\frac{\mathbf{x}_i}{\sqrt{m}}$ We have

$$\begin{aligned}
\max_{\mathbf{u}\in\mathbb{S}^{d-1}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \langle \mathbf{u}, \mathbf{x}\rangle^2 &= \max_{\mathbf{u}\in\mathbb{S}^{d-1}} \frac{1}{m}\sum_{i=1}^m \langle \mathbf{u}, \mathbf{x}_i\rangle^2 \\
&= \max_{\mathbf{u}\in\mathbb{S}^{d-1}} \frac{1}{m}\sum_{i=1}^m \mathbf{u}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{u} \\
&= \max_{\mathbf{u}\in\mathbb{S}^{d-1}} \mathbf{u}^T X X^T \mathbf{u} \\
&= \|X\|^2
\end{aligned}$$

Hence, $\mathcal{D}$ is $\|X\|$-bounded. In particular, by standard results in random matrices (e.g. theorem 5.39 in [24]), if $\{\mathbf{x}_i\}_{i=1}^m$ are independent and uniform points in the sphere and $m = \omega(d)$ then w.p. $1 - o(1)$ over the choice of the points, $\mathcal{D}$ is $(1 + o(1))$-bounded.

4. The uniform distribution on any orthonormal basis $\mathbf{v}_1, \ldots, \mathbf{v}_d$ is 1-bounded. Indeed, for any $\mathbf{u} \in \mathbb{S}^{d-1}$ and uniform $i \in [d]$ we have

$$\mathbb{E}_i \langle \mathbf{u}, \mathbf{v}_i\rangle^2 = \frac{1}{d}\sum_{i=1}^d \langle \mathbf{u}, \mathbf{v}_i\rangle^2 = \frac{\|\mathbf{u}\|^2}{d} = \frac{1}{d}$$