Lecture 4: NNGP, Dual Activations, and Over-parameterization

Adityanarayanan Radhakrishnan

Edited by: Max Ruiz Luyten, George Stefanakis, Cathy Cai

January 19, 2023

1 Introduction What is NNGP?

Having established the kernel regression framework, we now make explicit the connection between kernel regression and training the last layer of infinitely wide neural networks given by the Neural Network Gaussian Process (NNGP). We will develop the machinery to compute the NNGP for 1 hidden layer nonlinear networks, and present examples of applying kernel regression with the NNGP for varying activation functions. We will lastly introduce the double descent curve, which demonstrates that using higher dimensional feature maps (e.g. using wider neural networks) leads to improved generalization. Our derivation of the NNGP for 1 hidden layer neural networks lays the foundation for the derivation of the NNGP and Neural Tangent Kernel for networks of arbitrary depth in the upcoming lectures.

2 Random Feature Models and Neural Network Gaussian Processes

Consider a training dataset of samples $X = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}] \in \mathbb{R}^{d\times n}$ and labels $y = [y^{(1)}, y^{(2)}, \ldots, y^{(n)}] \in \mathbb{R}^{1\times n}$. Recall that kernel regression involves first applying a fixed mapping $\psi : \mathbb{R}^d \to \mathcal{H}$ into Hilbert space $\mathcal{H}$, and then performing linear regression in $\mathcal{H}$. In the previous lecture, we showed that linear regression can be performed in any Hilbert space by considering a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by $K(x, \tilde{x}) = \langle \psi(x), \psi(\tilde{x})\rangle_{\mathcal{H}}$ and applying the Representer theorem.

While in the previous lecture we considered a variety of standard kernel functions (e.g. Gaussian, Laplacian, etc.) without considering the corresponding feature maps, we will now proceed in the reverse direction. In particular, we will first consider feature maps given by randomly initialized neural networks and then compute corresponding kernel function as neural network width approaches infinity. As a simple yet powerful example, suppose we use a 1 hidden layer nonlinear neural network where only the last layer is trained and the first layer is a fixed random matrix. Then, we can think of the first layer as applying a fixed feature map ψ to our examples and then performing linear regression with the second layer, as is formalized below.

Example 1. *Let $X = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}] \in \mathbb{R}^{d\times n}$ denote training samples and $y = [y^{(1)}, y^{(2)}, \ldots, y^{(n)}] \in \mathbb{R}^{1\times n}$ denote training labels. Given $A \in \mathbb{R}^{1\times k}, B \in \mathbb{R}^{k\times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ an elementwise activation function, let $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(x) = A\phi(Bx)$ denote a 1 hidden layer neural network. Suppose $B_{i,j} \overset{i.i.d.}{\sim} \mathcal{P}$ for some probability distribution $\mathcal{P}$ and is fixed. Then, using gradient descent with $A^{(0)} = \mathbf{0}$ to minimize the loss:*

$$\mathcal{L}(A) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - f(x^{(i)}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - A\phi(Bx^{(i)}))^2$$

is equivalent to using kernel regression with the kernel $\Sigma(x, \tilde{x}) = \langle \phi(Bx), \phi(B\tilde{x})\rangle$ to map from X to y. See Fig. 1 for a visual depiction of the above setting.

Hence for any fixed finite width k, we can construct the kernel by computing the product $\langle \phi(Bx), \phi(B\tilde{x})\rangle$. Interestingly, under certain conditions on the initialization, we can tractably compute the inner product even in the limit as $k \to \infty$, and the resulting kernel is called the Neural Network Gaussian Process (NNGP).

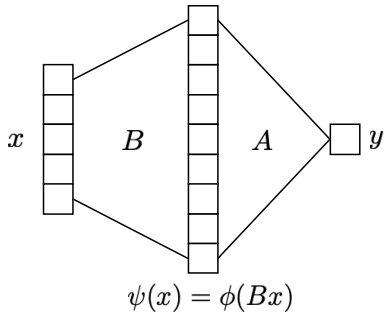

Figure 1: A visualization of how training the last layer of a finite width neural network is equivalent to performing kernel regression. When training only the last layer (i.e. updating only the matrix A), the first layer's weight matrix B is fixed and so the corresponding feature map for kernel regression is given by $\psi(x) = \phi(Bx)$.

Definition 1. *Given $A \in \mathbb{R}^{1\times k}, B \in \mathbb{R}^{k\times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ an elementwise activation function, let $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(x) = \frac{1}{\sqrt{k}} A\phi(Bx)$ denote a 1 hidden layer neural network. If $B_{i,j} \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$, then the **Neural Network Gaussian Process (NNGP)** is given by the kernel $\Sigma(x,\tilde{x}) = \lim_{k\to\infty} \left[\frac{1}{k}\langle\phi(Bx), \phi(B\tilde{x})\rangle\right]$.*

The reason for including the $\frac{1}{\sqrt{k}}$ scaling factor in Definition 1 is such that the kernel $\Sigma(x,\tilde{x})$ can be evaluated using the law of large numbers as follows.

Proposition 1. *Under the setting of Definition 1, $\Sigma(x,\tilde{x}) = \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}[\phi(u)\phi(v)]$ where*

$$\Lambda = \begin{bmatrix} \|x\|_2^2 & x^T\tilde{x} \\ x^T\tilde{x} & \|\tilde{x}\|_2^2 \end{bmatrix}.$$

Proof. From Definition 1, we have:

$$\begin{aligned} \lim_{k\to\infty}\left[\frac{1}{k}\langle\phi(Bx),\phi(B\tilde{x})\rangle\right] &= \lim_{k\to\infty}\left[\frac{1}{k}\sum_{i=1}^{k}\phi(Bx)_i\phi(B\tilde{x})_i\right] \\ &= \lim_{k\to\infty}\left[\frac{1}{k}\sum_{i=1}^{k}\phi(B_{i,:}x)\phi(B_{i,:}\tilde{x})\right] \\ &= \mathbb{E}_{w\sim\mathcal{N}(\mathbf{0},I_d)}[\phi(w^Tx)\phi(w^T\tilde{x})]\ ; \end{aligned}$$

where the last equality follows from the law of large numbers. Now, we can use the substitution $u = w^Tx$ and $v = w^T\tilde{x}$ where u, v are now jointly Gaussian random variables. As will be shown in the homework, we have:

$$\mathbb{E}[(u,v)] = (0,0) \ \ ; \ \ \mathrm{Cov}(u,v) = x^T\tilde{x}$$

Hence, we conclude:

$$\begin{aligned} \Sigma(x,\tilde{x}) &= \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}[\phi(u)\phi(v)] \\ \Lambda &= \begin{bmatrix} \|x\|_2^2 & x^T\tilde{x} \\ x^T\tilde{x} & \|\tilde{x}\|_2^2 \end{bmatrix}. \end{aligned}$$

□

In many cases, we directly compute such expectations via integration. For example, we can directly compute the NNGP for the ReLU function, i.e. $\phi(x) = \sqrt{2}\max(0,x)$, although the computation is involved (See

Appendix A). In particular, in the homework, we will compute the NNGP for $\phi(x) = \sin(x)$. We here give a list of activation functions for which the NNGP kernel has a closed form (implemented in the Neural Tangents Library [11]) at the time of writing: ReLU, Leaky ReLU, GeLU, Sine, Cosine, Error function (erf), Hermite polynomials. As a simple example, below we present the NNGP for the Random Fourier Feature model [13], which is recognizable as the Gaussian kernel. also called RBF

Example 2. *In the setting of Definition 1, let $\phi(z) = e^{iz}$ for $z \in \mathbb{R}$ and consider the inner product in $\ell_2(\mathbb{C})$ given by $\langle \{a_n\}_{n=0}^{\infty}, \{b_n\}_{n=0}^{\infty} \rangle = \sum_{n=0}^{\infty} a_n \overline{b_n}$. Then, the NNGP is given by:*

$$\begin{aligned}
\Sigma(x, \tilde{x}) &= \mathbb{E}_{w \sim \mathcal{N}(\mathbf{0}, I_d)}[\phi(w^T x)\overline{\phi(w^T \tilde{x})}] \\
&= \mathbb{E}_{w \sim \mathcal{N}(\mathbf{0}, I_d)}[e^{i(w^T x - w^T \tilde{x})}] \\
&= \prod_{j=1}^{d} \mathbb{E}_{w_j \sim \mathcal{N}(0,1)} \left[e^{i w_j (x_j - \tilde{x}_j)} \right] \\
&= \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i w_j (x_j - \tilde{x}_j)} e^{-\frac{w_j^2}{2}} dw_j \\
&= \prod_{j=1}^{d} e^{-\frac{(x_j - \tilde{x}_j)^2}{2}} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{\frac{(w_j - i(x_j - \tilde{x}_j))^2}{2}} dw_j \\
&= \prod_{j=1}^{d} e^{-\frac{(x_j - \tilde{x}_j)^2}{2}} \\
&= e^{-\frac{1}{2}\|x - \tilde{x}\|_2^2} \ ;
\end{aligned}$$

which is exactly the Gaussian kernel with $L = 1$.

Fortunately, even when it is difficult to compute the expectation above exactly, the following general theory of dual activations [7] allows for a simple approximation of the NNGP kernel.

3 Dual Activations

A note to the reader: This section involves more math than will be required for the course (i.e. Hermite polynomials and elements of functional analysis). It is not essential for computing and using the NNGP in practice and proofs may be skipped upon a first read-through if the reader is mainly interested in applications. The notation for dual activations, however, will be used throughout the course given that it considerably simplifies the formulation of the NTK for deep networks.

Given that we know $\Sigma(x, \tilde{x})$ if we can evaluate the expectation in Proposition 1, we now present tools that are useful for calculating this expectation. In particular, when data lie on the unit sphere, $\mathcal{S}^{d-1}$, the expectation we wish to compute is referred to as the dual activation.

Definition 2. *Let $\phi(x) : \mathbb{R} \to \mathbb{R}$ be an activation function. Then the **dual activation** of ϕ is given by $\check{\phi} : [-1, 1] \to \mathbb{R}$ where:*

$$\check{\phi}(\xi) = \mathbb{E}_{(u,v) \sim \mathcal{N}(\mathbf{0}, \Lambda)}[\phi(u)\phi(v)] \quad ; \quad \Lambda = \begin{bmatrix} 1 & \xi \\ \xi & 1 \end{bmatrix}.$$

In particular, we have the following explicit connection between the NNGP and the dual activation, when data lies on the unit sphere.

Corollary 1. *Under the setting of Definition 1, if $x, \tilde{x} \in \mathcal{S}^{d-1}$ and $\xi = x^T \tilde{x}$, then $\Sigma(x, \tilde{x}) = \check{\phi}(\xi)$.*

In order to simplify the expectation term in the dual activation, we turn to identifying a basis for the functions ϕ for which the expectations are easy to compute. In particular, we will show that the expectation term in the dual activation is easy to compute for the Hermite polynomials, which are defined below.

Definition 3 (Probabilist's Hermite Polynomial)**.** *The Hermite polynomials $\{h_i(x)\}_{i=0}^{\infty}$ are an orthonormal basis for the Hilbert space, $L^2(\mu)$, with inner product $\langle f, g\rangle = \int_{\mathbb{R}} f(x)g(x)e^{-\frac{x^2}{2}}dx$ constructed by performing Gram-Schmidt orthogonalization in $L^2(\mu)$ using the polynomials $\{x^i\}_{i=0}^{\infty}$.*

We list the first few probabilisit's Hermite polynomials below.

Example 3. $h_0(x) = 1, h_1(x) = x, h_2(x) = \frac{x^2-1}{\sqrt{2!}}, h_3(x) = \frac{x^3-3x}{\sqrt{3!}}$.

The dual of of a Hermite polynomial is convenient to compute as follows.

Proposition 2. *If $\phi(x) = h_n(x)$ for $x \in \mathcal{S}^{d-1}$, then $\check{\phi}(\xi) = \xi^n$ for $\xi \in [-1, 1]$.*

Proof. The proof is given in [12, Ch.11] and proceeds as follows. Let $x, \tilde{x} \in \mathcal{S}^{d-1}$ with $\xi = x^T\tilde{x}$. Let $w \sim \mathcal{N}(\mathbf{0}, I_d)$, and let $u = w^T x$ and $v = w^T\tilde{x}$. We then consider the following expectation for $s, t \in \mathbb{R}$:

$$\begin{aligned}
\mathbb{E}_w[\exp(su + tv)] &= \mathbb{E}_w\left[\exp\left(\sum_{i=1}^{d} w_i(sx_i + t\tilde{x}_i)\right)\right] \\
&= \prod_{i=1}^{d} \mathbb{E}_{w_i}\left[\exp\left(w_i(sx_i + t\tilde{x}_i)\right)\right] \\
&= \prod_{i=1}^{d} \exp\left(\frac{1}{2}(s^2x_i^2 + 2stx_i\tilde{x}_i + t^2\tilde{x}_i^2)\right) \\
&= \exp\left(\frac{1}{2}(s^2 + 2st\xi + t^2)\right);
\end{aligned}$$

where the third equality follows from completing the square inside the integral.[1] Hence, by multiplying the above by $\exp\left(-\frac{1}{2}(s^2+t^2)\right)$, we conclude:

$$\mathbb{E}_w\left[\exp\left(su + tv - \frac{1}{2}(s^2 + t^2)\right)\right] = \exp(st\xi) = \sum_{j=0}^{\infty} \frac{\xi^j(st)^j}{j!} \tag{1}$$

Now, if we write out the Taylor series for the exponential term on the left hand side, we necessarily have that the coefficients are polynomials in u and v. Namely,

$$\begin{aligned}
\mathbb{E}_w\left[\exp\left(su + tv - \frac{1}{2}(s^2 + t^2)\right)\right] &= \mathbb{E}_w\left[\exp\left(su - \frac{s^2}{2}\right)\exp\left(tv - \frac{t^2}{2}\right)\right] \\
&= \sum_{i,j=0}^{\infty} \frac{s^it^j}{i!j!}\mathbb{E}_w[H_i(u)H_j(v)] \quad ; \qquad (2)
\end{aligned}$$

where $\{H_n(x)\}_{n=0}^{\infty}$ are polynomials in x. Lastly, equating the terms in Eq.(1) and (2), we have:

$$\begin{aligned}
&\sum_{i,j=0}^{\infty} \frac{s^it^j}{i!j!}\mathbb{E}_w[H_i(u)H_j(v)] = \sum_{j=0}^{\infty} \frac{\xi^j(st)^j}{j!} \\
\implies &\mathbb{E}_w[H_i(u)H_j(v)] = \begin{cases} j!\xi^j & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}
\end{aligned}$$

[1] This is also the computation of the moment generating function for the Gaussian distribution.

Hence, we conclude that the polynomials H_i form an orthogonal basis for the space $L^2(\mu)$, and by normalizing them such that $h_i(x) = \frac{1}{\sqrt{i!}} H_i(x)$, we conclude that $\{h_i\}$ are indeed the Hermite polynomials. Thus, for $\phi(x) = h_i(x)$:

$$\check{\phi}(\xi) = \mathbb{E}_w[h_i(u)h_i(v)] = \xi^i$$

□

Assuming our function $\phi \in L^2(\mu)$, we can then use the orthogonal decomposition of ϕ into the Hermite basis to get the following result.

Theorem 1. *Let $\phi \in L^2(\mu)$ such that $\phi(x) = \sum_{n=0}^{\infty} a_n h_n(x)$ for $x \in \mathcal{S}^{d-1}$. Then, $\check{\phi}(\xi) = \sum_{n=0}^{\infty} a_n^2 \xi^n$ for $\xi \in [-1, 1]$.*

Proof. We substitute in the basis expansion of ϕ into the definition of the dual activation and then utilize the result of Proposition 2 with linearity of expectation. Namely,

$$\begin{aligned}
\check{\phi}(\xi) &= \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}[\phi(u)\phi(v)] \\
&= \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}\left[\left(\sum_{n=0}^{\infty} a_n h_n(u)\right)\left(\sum_{m=0}^{\infty} a_m h_m(v)\right)\right] \\
&= \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}\left[\left(\sum_{n=0}^{\infty} a_n^2 h_n(u)h_n(v)\right)\right] \\
&= \sum_{n=0}^{\infty} a_n^2 \xi^n
\end{aligned}$$

□

Theorem 1 implies the following key properties of the NNGP that would be nontrivial to observe otherwise.

Corollary 2. *Let $\phi \in L^2(\mu)$ be an activation function and let $\check{\phi}$ denote the dual activation. Then, $\check{\phi}$ satisfies the following:*

(a) $\check{\phi}$ is non-decreasing and convex in $[0, 1]$.

(b) $(\check{\phi})' = (\check{\phi}')$ (the dual commutes with differentiation).

(c) $\check{\phi}$ is continuous in $[-1, 1]$ and smooth (infinitely differentiable) in $(-1, 1)$.

(d) The range of $\check{\phi}$ is given by $[-\|\phi\|_{L^2(\mu)}^2, \|\phi\|_{L^2(\mu)}^2]$.

We leave the proofs as exercises to the reader. We lastly remark that when computing the dual activation function, it is often convenient to normalize the activation ϕ such that $\|\phi\|_{L^2(\mu)} = 1$ to avoid additional constants in derivations. In particular, for the ReLU activation, this normalizing constant is $\sqrt{2}$ (as will be shown in homework), and so we will compute the infinite width limits of neural networks with the normalized activation $\phi(x) = \sqrt{2}\max(x, 0)$.

With more tools to compute the NNGP in hand, we turn to analyzing the performance of the NNGP in practice. In particular, we will show that taking the NNGP for increasingly wide networks leads to better generalization performance, as predicted by the double descent curve.

4 The Benefit of Over-parameterization

Thus far, we have connected kernel methods to training infinitely wide neural networks via the NNGP, but we have not yet analyzed the effectiveness of using the NNGP over simply using a finite width feature map. Naturally, when network width is large, solving kernel regression with the NNGP offers a clear computational advantage. Remarkably, as we will now show empirically, using the NNGP also leads to

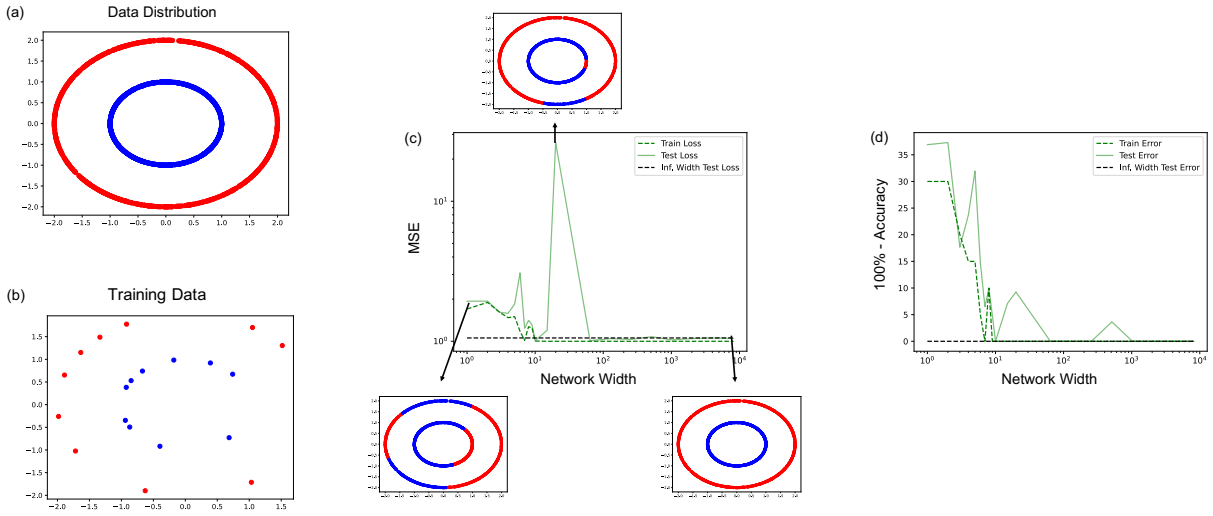

Figure 2: A comparison between using the NNGP for a 1-hidden layer fully connected network with sine activation function and training the last layer of corresponding finite width networks. (a) A visualization of the data distribution used for the task. The data consists of two concentric circles with red points having class label $y = -1$ and blue points having class label $y = 1$. (b) A visualization of the 20 samples used to train all models. (c) The training and test mean squared error as a function of network width (shown in green). We see that wider models achieve the lowest test loss even though they interpolate the data, and that the test loss limits to that of the NNGP (shown in black). Furthermore, the maximum loss occurs when network width equals the number of samples. The subfigures illustrate the functions learned at network widths of 1, 20, and 8192 respectively. (d) The training and test error (100% - accuracy) as a function of network width. Again, increasing width leads to improved accuracy on this task.

improved generalization over the corresponding finite-width feature maps. We begin with a simple empirical demonstration of this phenomenon when classifying data drawn from two concentric circles of differing class label, i.e. Example 1 from Lecture 3.

Example 4. *Let training samples and labels be drawn according to the distribution (shown in Fig. 2a) consisting of two concentric circles with differing class label. We consider the setting in which we only observe 10 samples from each of the circles for training, as shown in Fig. 2b, and evaluate our trained models on* 1000 *samples from test data drawn from the true distribution. We measure the performance of the model via the error, which is just 100% minus the accuracy, and via the mean squared error between the predictions and the true labels.*

In Figs.2c and d (and Appendix B Fig.4), we compare the performance of the NNGP to that of neural networks of finite width where only the last layer is trained. In particular, we consider the NNGP for the activation functions $\phi(x) = C_1 \sin(x)$ *where* $C_1 = \sqrt{\frac{e^2}{e^2-1}}$ *and* $\phi(x) = \sqrt{2}\max(0, x)$*, which are given as follows:*

1. *For* $\phi(x) = C_1 \sin(x)$*:* $\Sigma(x, \tilde{x}) = C_1^2 \left[\exp\left(-\frac{1}{2}(\|x\|_2^2 + \|\tilde{x}\|_2^2) + x^T\tilde{x}\right) - \exp\left(-\frac{1}{2}(\|x\|_2^2 + \|\tilde{x}\|_2^2) - x^T\tilde{x}\right)\right]$
2. *For* $\phi(x) = \sqrt{2}\max(0, x)$*:* $\Sigma(x, \tilde{x}) = \frac{1}{\pi}\left(x^T\tilde{x}\left(\pi - \arccos\left(\frac{x^T\tilde{x}}{\|x\|_2\|\tilde{x}\|_2}\right)\right) + \sqrt{\|x\|_2^2\|\tilde{x}\|_2^2 - (x^T\tilde{x})^2}\right)$

We use the pseudo-inverse to solve kernel regression when the corresponding feature map yields a semi-definite kernel. We make the following observations regarding the trends in the figure.

1. *Increasing network width well past the point of achieving* 0 *training loss (i.e. interpolating the training data) leads to lower test error and loss.*
2. *The model width that yields the worst performance occurs roughly when the width is equal to the number*

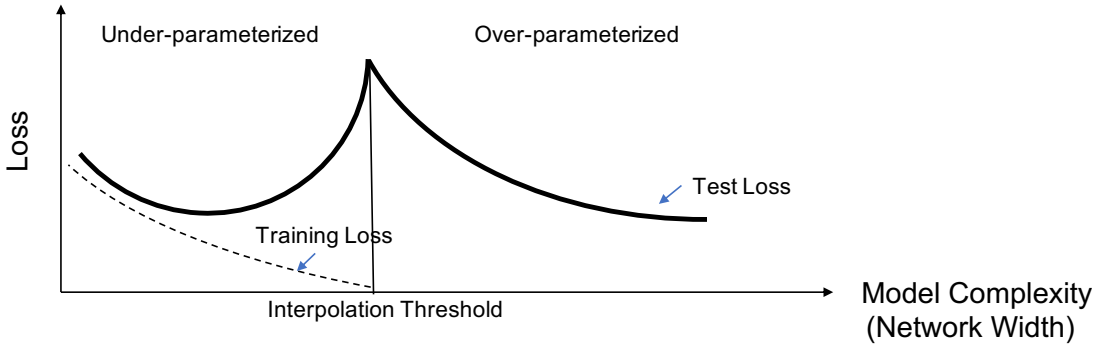

Figure 3: A schematic depicting the double descent phenomenon shown empirically in Fig. 2c. In particular, in the the under-parameterized (or classical) regime, we observe traditional overfitting: we are unable to interpolate the data and as the training loss decreases, the test loss increases. In the over-parameterized (or modern) setting, all models perfectly fit the training data, but the test loss remarkably decreases as the model complexity (for our purposes, neural network width increases. Additionally, as observed in Fig. 2, the loss is maximized at the *interpolation threshold*, which occurs when the model complexity is near the number of training examples.

of training examples.

Remarks. The example above illustrates the benefit of using over-parameterized models, i.e. those that can perfectly fit training data, in machine learning. This phenomenon is indeed quite general [1, 5, 9, 10] and is characterized by the *double descent* curve [5] (See Fig. 3). Moreover, there is an emerging theoretical understanding of the benefit of over-parameterization [2, 6, 8], and an active area of research involves formally characterizing the error as a function of model capacity (e.g. network width). We point the interested reader to [3, 4] for reviews of recent work in the area. For the purposes of this course, the benefit of over-parameterization is a key observation motivating the use of all of the methods in this course.

5 Discussion

In this lecture, we made a first, simple connection between solving kernel regression and training the last layer of a 1 hidden layer neural network. As network width approached infinity, we provided conditions under which we could explicitly calculate a closed form for the kernel, named the NNGP. We then provided machinery based on the theory of dual activations to simplify computation of the NNGP in general. Lastly, we compared the performance of the NNGP with that of training the last layer of finite width networks. We observed that increasing network width led to improved generalization, thus motivating the computational and generalization benefits of the NNGP. Indeed, the last section highlighted an important theme of the course: the kernels corresponding to infinite width limits are often simple to compute and use, but remarkably, also lead to results that are competitive with training finite width neural networks.

In the next lecture, we will extend our simple NNGP connection between kernel regression and training the last layer of a network to the more powerful Neural Tangent Kernel (NTK) connection between kernel regression and training *all layers* of an infinitely wide neural network. The tools developed based on the theory of dual activations will drastically simplify computation of the NTK and allow us to easily compute the NNGP and NTK for deep fully connected networks.

References

[1] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu. Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks. In *International Conference on Learning Representations*, 2020.

[2] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[3] P. L. Bartlett, A. Montanari, and A. Rakhlin. Deep learning: a statistical viewpoint. *Acta Numerica*, 30:87–201, 2021.

[4] M. Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*, 30:203–248, 2021.

[5] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[6] M. Belkin, D. Hsu, and J. Xu. Two models of double descent for weak features. *Society for Industrial and Applied Mathematics Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020.

[7] A. Daniely, R. F. Frostig, and Y. Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

[8] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv:1903.08560*, 2019.

[9] J. Lee, S. S. Schoenholz, J. Pennington, B. Adlam, L. Xiao, R. Novak, and J. Shol-Dickstein. Finite Versus Infinite Neural Networks: an Empirical Study. In *Advances in Neural Information Processing Systems*, 2020.

[10] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference in Learning Representations*, 2020.

[11] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz. Neural Tangents: Fast and easy infinite neural networks in Python. In *International Conference on Learning Representations*, 2020.

[12] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[13] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

A Dual Activation of ReLU

Proposition 3 (NTK for 1 Hidden Layer ReLU)**.** *Assume* $\|x\|_2 = \|x'\|_2 = 1$*, and let* $c = \sqrt{2}$ *and* $\xi = x^T x'$*. Then,*

$$\check{\phi}(\xi) = c^2 \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Sigma)}[\phi(u)\phi(v)] = \frac{1}{\pi}(\xi(\pi - \cos^{-1}(\xi)) + \sqrt{1-\xi^2})$$
$$\check{\phi}'(\xi) = c^2 \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Sigma)}[\phi'(u)\phi'(v)] = \frac{1}{\pi}(\pi - \cos^{-1}(\xi))$$

Proof. Below is the evaluation of $\check{\phi}(\xi)$ for the function $\phi(x) = \mathbf{1}_{x>0}$ (the indicator function that is 1 for $x > 0$ and 0 for $x \leq 0$). One can proceed analogously to derive the dual activation of piece-wise polynomial functions. Consider $\xi \in (-1, 1)$:

$$\check{\phi}(\xi) = 2\mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Sigma)}[\phi(u)\phi(v)] = \frac{1}{\pi\sqrt{1-\xi^2}} \int_0^\infty \int_0^\infty \exp\left(-\frac{u^2+v^2-2uv\xi}{2(1-\xi^2)}\right) dudv$$

Now, we make the substitution $u = r\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right)$ and $v = r\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right)$. We use the following trignometric identity to simplify uv:

$$2uv = 2r^2\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right)\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) = 2\frac{r^2}{2}\left(\sin\left(\theta + \frac{\pi}{2}\right) - \sin(0)\right) = r^2\cos(\theta)$$

Thus, we have that:

$$\check{\phi}(\xi) = \frac{1}{2\pi\sqrt{1-\xi^2)}}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^\infty r\exp\left(-\frac{r^2 - r^2\cos(\theta)\xi}{2(1-\xi^2)}\right)drd\theta$$

We now evaluate the integral corresponding to r and to simplify notation, let $c = \frac{1-\cos(\theta)\xi}{(1-\xi^2)}$. We then have:

$$\int_0^\infty r\exp\left(-\frac{r^2c}{2}\right)dr = \int_0^\infty \exp\left(-uc\right)du = \frac{1}{c} = \frac{1-\xi^2}{1-\cos(\theta)\xi}$$

Thus,

$$\check{\phi}(\xi) = \frac{\sqrt{1-\xi^2}}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{1}{1-\cos(\theta)\xi}d\theta = \frac{\sqrt{1-\xi^2}}{\pi}\int_0^{\frac{\pi}{2}}\frac{1}{1-\cos(\theta)\xi}d\theta \tag{3}$$

We can evaluate this integral directly using the following substitutions:

$$u = \tan\left(\frac{\theta}{2}\right) \qquad d\theta = \frac{2}{u^2+1}du \qquad \cos(\theta) = \frac{1-u^2}{u^2+1}$$

We then get:

$$\check{\phi}(\xi) = \frac{\sqrt{1-\xi^2}}{\pi}\frac{2}{\sqrt{1-\xi^2}}\tan^{-1}\left(\frac{\sqrt{1+\xi}}{\sqrt{1-\xi}}\right) = \frac{1}{\pi}(\pi - \cos^{-1}(\xi))$$

The last equality follows from the identities below (assuming $\xi = \cos(\phi)$):

$$\tan^{-1}\left(\frac{1+\cos(\phi)}{\sin(\phi)}\right) = \tan^{-1}\left(\frac{1}{\tan\left(\frac{\phi}{2}\right)}\right) = \frac{\pi}{2} - \tan^{-1}\left(\tan\left(\frac{\phi}{2}\right)\right) = \frac{\pi}{2} - \frac{\phi}{2}$$

□

B Empirical Evaluation with ReLU NNGP

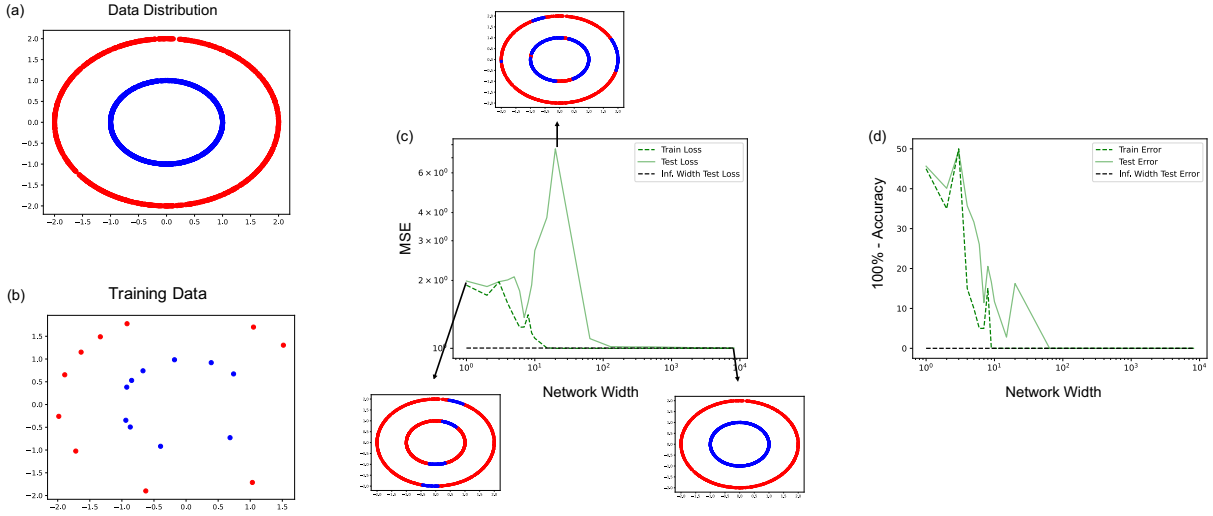

Figure 4: A comparison between using the NNGP for a 1-hidden layer fully connected network with ReLU activation function and training the last layer of corresponding finite width networks. The data distribution in (a), (b) are the same as those of Fig. 2. We again see the same trends as in Fig. 2. Namely, in (c) and (d), we observe that, in the interpolating regime, increasingly larger models lead to lower loss and error, which both limit to that of the infinite width model corresponding to the NNGP. We also observe in (c) that the loss is again maximized when the number of examples equals the width.