# Notes on Random Feature Regression and Wide Neural Networks

Mihai Nica

February 17, 2021

## Contents

# 1 Feature Regression and the Random Feature Model

| Notation | Space object lives in | Definition |
|---|---|---|
| $n_{ex}$ | $\mathbb{N}$ | $n_{ex}$ = Number of examples |
| $n_{in}$ | $\mathbb{N}$ | $n_{in}$ = Input dimension |
| $n_{par}$ | $\mathbb{N}$ | $n_{par}$ = Number of parameters, |
| $x^{(i)}, y^{(i)}$ | $\mathbb{R}^{n_{in}} \times \mathbb{R}^1$ | $x^{(i)}, y^{(i)} = i^{\text{th}}$training example for $1 \leq i \leq n_{ex}$ (Note: The output dimension here is $y^{(i)} \in \mathbb{R}^1$ but this can be generalized to $\mathbb{R}^{n_{out}}$ without much trouble. I am keeping it $\mathbb{R}^1$ to keep things a bit easier to understand) |
| $\mathcal{X}, \mathcal{Y}$ | $(\mathbb{R}^{n_{in}})^{n_{ex}} \times \mathbb{R}^{n_{ex}}$ | Notation used for all the training examples together $\mathcal{X}, \mathcal{Y} = \left(x^{(i)}, y^{(i)}\right)_{i=1}^{n_{ex}}$ |
| $x$ | $\mathbb{R}^{n_{in}}$ | Generic point in the input space $\mathbb{R}^{n_{in}}$ |
| $\theta$ | $\mathbb{R}^{n_{par}}$ | Vector of all the parameters of the model (e.g. all the weights and biases together) |
| $\theta_t$ | $\mathbb{R}^{n_{par}}$ | Parameters at time $t$ along some sequence of parameters (e.g. how parameters evolve under gradient descent) |
| $f(x;\theta)$ | $f : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{par}} \to \mathbb{R}$ | Model function at generic input $x$ and parameters $\theta$ (e.g. a regression model or a neural network) |
| $\nabla_\theta f(x;\theta)$ | $\nabla_\theta f : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{par}} \to \mathbb{R}^{n_{par}}$ | Gradient of the model w.r.t the parameters thought of as a vector in $\mathbb{R}^{n_{par}}$, $(\nabla_\theta f)_i (x;\theta) = \frac{\partial f}{\partial \theta_i}(x;\theta)$ for $1 \leq i \leq n_{par}$ |
| $f(\mathcal{X};\theta)$ | $f(\mathcal{X};\cdot) : \mathbb{R}^{n_{par}} \to \mathbb{R}^{n_{ex}}$ | Stacked vector of the output for all the examples $f(\mathcal{X};\theta) = \begin{bmatrix} f(x^{(1)};\theta) \\ \vdots \\ f(x^{(n_{ex})};\theta) \end{bmatrix} \in \mathbb{R}^{n_{ex}}$ |
| $K(x, x')$ | $K : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ | A "kernel". Can be thought of a generalization of a dot product $K(x, x') = \langle x, x' \rangle$ |
| $K(\mathcal{X}, \mathcal{X})$ | $\mathbb{R}^{n_{ex}} \times \mathbb{R}^{n_{ex}}$ | The matrix you get by applying the $K(\cdot,\cdot)$ to all the input examples from $\mathcal{X}$ $K(\mathcal{X}, \mathcal{X}) = \left[K\left(x^{(i)}, x^{(j)}\right)\right]_{i,j=1}^{n_{ex}} \in \mathbb{R}^{n_{ex}} \times \mathbb{R}^{n_{ex}}$ |
| $\phi(x)$ | $\phi : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{par}}$ | Feature map with $n_{par}$ features in it$\phi(x) = \left(\phi_1(x), \ldots, \phi_{n_{par}}(x)\right) \in \mathbb{R}^{n_{par}}$ |
| $\phi(\mathcal{X})$ | $\mathbb{R}^{n_{par}} \times \mathbb{R}^{n_{ex}}$ | Feature map applied to the training examples $\phi(\mathcal{X}) = \begin{bmatrix} \phi(x^{(1)}) \\ \vdots \\ \phi\left(x^{(n_{ex})}\right) \end{bmatrix} \in \mathbb{R}^{n_{par}} \times \mathbb{R}^{n_{ex}}$ |
| $\mathcal{N}(\mu, \Sigma)$ | $\mathbb{R}^{dim(\mu)}$ | Notation for a Gaussian random vector with mean $\mu \in \mathbb{R}^{dim(\mu)}$ and covariance matrix $\Sigma \in \mathbb{R}^{dim(\mu) \times dim(\mu)}$ |

Other things, I sometimes write the sizes of the matrices during matrix multiplication to make sure it all works out! e.g.

$$\underset{n\times m}{A} = \underset{n\times p}{B}\underset{p\times m}{C}$$

means that $A$ is an $n \times m$ matrix, $B$ is an $n \times p$ matrix, and $C$ is a $p \times m$ matrix.

## 1.1 Definition of the model and main results

**Definition 1.** *[Feature regression model with Gaussian initialization]* As is typical in ***supervised learning***, suppose we are given a list of $n_{ex}$ input examples $x^{(i)} \in \mathbb{R}^{in}$ and the corresponding outputs examples $y^{(i)} \in \mathbb{R}^1$ for $1 \leq i \leq n_{ex}$.[1] Our goal is to find a function $f : \mathbb{R}^{in} \to \mathbb{R}^1$ that fits these given examples well and hopefully generalizes to other unseen test points $x \in \mathbb{R}^{n_{in}}$,$y \in \mathbb{R}^1$ from the data distribution.

One way to do this is the following 2 step procedure:

[1] In the discussion here, the output space can be generalized to $y \in \mathbb{R}^{n_{out}}$ for some $n_{out} \in \mathbb{N}$ without any significant changes. We are using the output dimension $n_{out} = 1$ here for simplicity.

**Step 1.** Choose $n_{par}$ ***feature maps*** $\phi_i : \mathbb{R}^{in} \to \mathbb{R}$ for $1 \leq i \leq n_{par}$ that capture different features of the inputs $x \in \mathbb{R}^{in}$. This creates a feature vector $\phi(x) \in \mathbb{R}^{n_{par}}$ for each input $x \in \mathbb{R}^{n_{in}}$.

**Step 2a.** Set up the ***linear regression*** with $n_{par}$ coefficients using the features from Step 1. This is the parameterized model $f : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{par}} \to \mathbb{R}$ given by:

$$f(x;\theta) = \frac{1}{\sqrt{n_{par}}} \underset{1\times n_{par}}{\theta^T} \underset{n_{par}\times 1}{\phi(x)} = \frac{1}{\sqrt{n_{par}}} \sum_{i=1}^{n_{par}} \theta_i \phi_i(x) \tag{1}$$

**Step 2b.** Initialize the coefficients to be a ***Gaussian random vector*** with independent standard mean zero entries of variance 1: i.e. start $\theta_0$ to be the random vector $\theta_0 \sim \mathcal{N}(0, I^{n_{par}})$. Then optimize the parameters $\theta$ by gradient descent on the loss function[2]

$$L(\theta) = \frac{1}{2} \left\| f(\mathcal{X};\theta) - \mathcal{Y} \right\|^2 = \frac{1}{2} \sum_{i=1}^{n_{ex}} \left( f(x^{(i)};\theta) - y^{(i)} \right)^2$$

For the theoretical discussion here, we will consider updating the parameters $\{\theta_t\}_{t\in\mathbb{R}^+}$ with $t$ being a ***continuous time parameter*** evolving according to the ***gradient flow differential equation***[3]

$$\frac{\mathrm{d}}{\mathrm{d}t}\theta_t = -\alpha \nabla_\theta L(\theta_t) \tag{2}$$

*Remark* 2. The choice of the normalization constant $\frac{1}{\sqrt{n_{par}}}$ ensures that even if $n_{par}$ grows large, the function $f(x;\theta)$ does not grow larger and larger. For example, since the $\theta_i$ are initialized to independent Gaussian random vectors, the $f(x;\theta_0)$ is a Gaussian too with variance $\mathbf{Var}\left[f(x;\theta)\right] = \frac{1}{n_{par}}\phi(x)^T\phi(x) = \frac{1}{n_{par}}\sum_{i=1}^{n_{par}} \phi_i^2(x)$ is proportional to the *average* value of $\phi_i(x)^2$ over all features $1 \leq i \leq n_{par}$. Without the normalization constant in front, this variance would tend to infinity as we add more and more features.

**Example 3.** Depending on the feature maps $\phi$ that are chosen, this model may be very simple or more complicated.

- If $n_{par} = n_{in}$ and $\phi(x) = x$ is the identity, than this model is the classic linear regression model.
- If $n_{in} = 1$ and $\phi$ contains the monomials $\phi(x) = \left[1, x, x^2, \ldots, x^{n_{par}-1}\right]$ then this model is just polynomial regression.
- A model that has received some attention recently is the ***random features model*** where we set $\phi_i(x) = w_i \cdot x$ for some independent random variables $w_i \in \mathbb{R}^{n_{in}}$, $1 \leq i \leq n_{par}$
- One can set $\phi(x)$ to be the output of a pretrained neural network for a related task. For example, if you are doing image recognition, and have already trained $\phi(x)$ from a large datatset, you can now do this regression model to train a model that works on a new dataset more quickly.

**Problem 4.** Is there a way to analyze the resulting function we get if we run gradient flow long enough? i.e. Can we understand the limiting function

$$f(x;\theta_\infty) := \lim_{t\to\infty} f(x;\theta_t).$$

Some things we would like to know:

- Can we say anything about how the model generalizes on unseen data $x$?
- How does the choice of function $\phi$ effect the final output?

[2] This loss function uses *all* the given examples so it sometimes called "full batch gradient descent". In practice, one often uses a subset of the examples rather than all of them.

[3] In practice, it is more common to see the discrete time $t \in \mathbb{N}$ gradient descent update rule $\theta_{t+1} = \theta_t - \eta\nabla_\theta L(\theta_t)$ rather than gradient flow. It is not difficult to show that under mild assumptions that if the learning rate $\eta$ is small enough then discrete time gradient descent and continuous time gradient flow are almost the same. We will stick to gradient flow as the continous time nature of it makes the calculus easier!

**Solution 5.** Yes we can! It turns out that this model has an analytical solution can be written entirely in terms of the ***kernel*** $K : \mathbb{R}^{in} \times \mathbb{R}^{in} \to \mathbb{R}$ given by the scaled dot product of the ***features***:

$$K(x, x') = \frac{1}{n_{par}} \underset{1\times n_{par}}{\phi(x)^T} \underset{n_{par}\times 1}{\phi(x')} = \frac{1}{n_{par}} \sum_{i=1}^{n_{par}} \phi_i(x)\phi_i(x') \tag{3}$$

This kernel can be thought of as a generalization of the inner product $x \cdot x'$and it depends only on the ***feature maps*** $\phi$ that were chosen in Step 1. The given $n_{ex}$ examples $\mathcal{X} = \left(x^{(1)}, \ldots, x^{(n_{ex})}\right)$ can be fed into the kernel to get an $n_{ex} \times n_{ex}$ **matrix** $K(\mathcal{X}, \mathcal{X})$:

$$K(\mathcal{X}, \mathcal{X}) = \frac{1}{n_{par}} \underset{n_{ex}\times n_{par}}{\phi(\mathcal{X})^T} \underset{n_{par}\times n_{ex}}{\phi(\mathcal{X})} = \left[K(x^{(i)}, x^{(j)})\right]_{i,j=1}^{n_{ex}} \tag{4}$$

It turns out that as long as this matrix is invertible[4], then many questions about the limit $f(x, \theta_\infty)$ can nicely be written in terms of the inverse $K^{-1}(\mathcal{X}, \mathcal{X}) := (K(\mathcal{X}, \mathcal{X}))^{-1}$.

**Theorem 6.** *[Training dynamics of the model] Let $K(x, x') = \frac{1}{n_{par}}\phi(x)^T\phi(x')$ be the kernel from (3) and let $K(\mathcal{X}, \mathcal{X})$ be the $n_{ex} \times n_{ex}$ matrix from (4). Because the gradient of the model $\nabla_\theta f(x;\theta)^T \nabla_\theta f(x';\theta) = K(x, x')$ **does not depend on** $\theta$ the following happens, For an arbitrary test point $x \in \mathbb{R}^{n_{in}}$, the evolution of $f(x;\theta_t)$ can be written in term of the kernel$K(\cdot,\cdot)$, and the training data $\mathcal{X}, \mathcal{Y}$ as*

$$\frac{d}{dt} f(x;\theta_t) = \alpha \underset{1\times n_{ex}}{K(x, \mathcal{X})}(\mathcal{Y} - \underset{n_{ex}\times 1}{f(\mathcal{X};\theta_t)})$$

*where $K(x, \mathcal{X})$ is the row vector $\underset{1\times n_{ex}}{K(x, \mathcal{X})} = \frac{1}{n_{par}} \underset{1\times n_{par}}{\phi(x)^T} \underset{n_{par}\times n_{ex}}{\phi(\mathcal{X})} = \left[K(x, x^{(i)})\right]_{i=1}^{n_{ex}} \in \mathbb{R}^{1\times n_{ex}}$.*

*Suppose in addition that the matrix $K(\mathcal{X}, \mathcal{X})$ is invertible. Then the training data will be perfectly fit with zero training error:*

$$f(\mathcal{X};\theta_\infty) = \mathcal{Y}$$

*and moreover we can explicitly solve to get the solution at any test point $x$ as*[5]

$$f(x;\theta_\infty) - f(x;\theta_0) = \underset{1\times n_{ex}}{K(x, \mathcal{X})} \underset{n_{ex}\times n_{ex}}{K^{-1}(\mathcal{X}, \mathcal{X})}(\mathcal{Y} - \underset{n_{ex}\times 1}{f(\mathcal{X};\theta_0)})$$

$$\theta_\infty - \theta_0 = \frac{1}{\sqrt{n_{par}}} \phi(\mathcal{X}) K^{-1}(\mathcal{X}, \mathcal{X}) \left(\mathcal{Y} - f(\mathcal{X};\theta_0)\right)$$

**Proposition 7.** *[Initial distribution of the model] Recall that the initial parameters $\theta_0$ are independent Gaussian with $\theta_0 \sim \mathcal{N}(0, \sigma I^{n_{par}})$. For any fixed point $x \in \mathbb{R}^{n_{in}}$, the random variable $f(x;\theta_0)$ is Gaussian with mean and variance given as follows*

$$\mathbf{E}\left[f(x;\theta_0)\right] = 0, \ \mathbf{Var}\left[f(x;\theta_0)\right] = K(x, x)$$

*Moreover, for any finite collection of test points $\mathcal{X}_{test} = \left(x_{test}^{(1)}, \ldots, x_{test}^{(n_{test})}\right)$, the vector $f(\mathcal{X}_{test};\theta_0) = \left[f(x_{test}^{(i)};\theta_0)\right]_{i=1}^{n_{test}} \in \mathbb{R}^{n_{test}}$ is Gaussian with mean $\mathbf{E}\left[f(\mathcal{X}_{test};\theta_0)\right] = 0$ and covariance matrix*

$$\mathbf{Cov}\left[f(\mathcal{X}_{test};\theta_0)\right] = K(\mathcal{X}_{test}, \mathcal{X}_{test}) = \left[K(x_{test}^{(i)}, x_{test}^{(j)})\right]_{i,j=1}^{n_{test}}$$

*Remark* 8. Sometimes people say "$f(x;\theta_0)$ is a Gaussian process of mean 0 and kernel $K(x, x')$" or write $f(x;\theta_0) \sim GP(0, K(x, x'))$ to mean exactly the statement of Proposition 17. This just means that any finite collection of points gives a Gaussian vector with covariance matrix given by the kernel $K$.

Combining Proposition 7 and Theorem 6 gives us the distribution after training.

[4]Note that $K(\mathcal{X}, \mathcal{X}) = \phi(\mathcal{X})^T\phi(\mathcal{X})$ is invertible if and only if the $n_{ex}$ feature vectors $\phi(x^{(1)}), \ldots, \phi(x^{(n_{ex})})$ are an independent set of vectors in the space $\mathbb{R}^{n_{par}}$. This can only happen in the overparametrized regime where $n_{par} \geq n_{in}$. This is the regime we will focus on here!

[5]A similar formula exists for intermediate times $f(x;\theta_t)$.

**Theorem 9.** *For any test point $x \in \mathbb{R}^{n_{in}}$, $f(x;\theta_\infty)$ is a Gaussian random variable with mean and variance given by*

$$\begin{aligned} \mathbf{E}\left[f(x;\theta_\infty)\right] &= \underset{1\times n_{ex}}{K(x,\mathcal{X})}\underset{n_{ex}\times n_{ex}}{K^{-1}(\mathcal{X},\mathcal{X})}\underset{n_{ex}\times 1}{\mathcal{Y}} \\ \mathbf{Var}\left[f(x;\theta_\infty)\right] &= K(x,x) - \underset{1\times n_{ex}}{K(x,\mathcal{X})}\underset{n_{ex}\times n_{ex}}{K^{-1}(\mathcal{X},\mathcal{X})}\underset{n_{ex}\times 1}{K(\mathcal{X},x)} \end{aligned} \tag{5}$$

*Moreover, for any set of test points $\mathcal{X}_{test} = \left(x_{test}^{(1)},\ldots,x_{test}^{(n_{test})}\right)$ the joint distribution of the vector $f(\mathcal{X}_{test};\theta_\infty)$ is also Gaussian with mean and covariance given by formulas similar to (5). Similar explicit formulas can be given for $f(x;\theta_t)$ at any intermediate time $0<t<\infty$. The random variable $f(x;\theta_\infty)$ has the following equivalent interpretation as a conditional probability: $f(x;\theta_\infty)$ is equal in distribution to $f(x;\theta_0)$ conditioned on the event $f(\mathcal{X};\theta_0)=\mathcal{Y}$, i.e.*

$$\mathbf{P}\left(f(x;\theta_\infty)\in A\right) = \mathbf{P}\left(f(x;\theta_0)\in A \,|\, f(\mathcal{X};\theta_0)=\mathcal{Y}\right)\ \forall A\subset\mathbb{R},$$

In the next few sections I will show the proofs, which are not too difficult. First some remarks!

## 1.2 Random Feature Regression

### 1.2.1 Random Feature Model

**Definition 10.** [Random feature model] One interesting choice of features $\phi$ is to choose them randomly. For example, if we fix a non-linearity $\varphi:\mathbb{R}\to\mathbb{R}$, one can make the features$\phi_i$ independent random functions by setting

$$\phi_i(x) = \varphi\left(\langle y_i, x\rangle\right)$$

where $y_i\in\mathbb{R}^{n_{in}}$ are independently randomly chosen vectors and $\varphi:\mathbb{R}\to\mathbb{R}$ is some fixed non-linearity function. Note that in this case the kernel $K$ is given by:

$$K(x,x') = \frac{1}{n_{par}}\sum_{i=1}^{n_{par}}\varphi\left(\langle y_i,x\rangle\right)\varphi\left(\langle y_i,x'\rangle\right)$$

Since the various $y_i$ are independent, we are looking at an average of independent random variables! By the law of large numbers,

$$K(x,x') \approx \mathbf{E}_Y\left[\varphi\left(\langle Y,x\rangle\right)\varphi\left(\langle Y,x'\rangle\right)\right]$$

and this approximation will improve as $n_{par}\to\infty$.

### 1.2.2 "Train-Only-the-Last-Layer" Neural Network

Another example of a random feature model is to train a neural network where ONLY the weights on the last layer are updated through training, and the weights on the first part of the network remain frozen to their values of initialization. Imagine a network with hidden layer widths $n_1,\ldots,n_{L-1}$ and non-linearity $\varphi$. The output of the network can be written as a simple function of the last hidden layer $f^{L-1}$:

$$f(x;W,b) = \frac{\sigma_W}{\sqrt{n_\ell}}W\varphi\left(f^{L-1}(x;\theta_0^{hid})\right)+\sigma_b b$$

, where the non-linearity $\varphi$ is applied entry-wise. The trainable parameters of the model are the weights $W$ and bias $b$ of the last layer, while the output of the previous layer $f^{L-1}(x;\theta_0^{hid})$ is always fixed to whatever it is at initialization. This is precisely a random feature model with features given by the components of $f^{L-1}(x;\omega_0)$. The relevant kernel is:

$$K(x,x') = \nabla_{\{W,b\}}f^L(x;W,b)^T\nabla_{\{W,b\}}f^L(x;W,b) = \frac{\sigma_W^2}{n_\ell}\varphi\left(f^{L-1}(x;\omega_0)\right)^T\varphi\left(f^{L-1}(x';\omega_0)\right)+\sigma_b^2$$

and in the limit that the size of the last hiddne layer $n_{L-1}\to\infty$ tends to infinitiy, we will again have by the law of large numbers the convergence:

$$K(x,x') \approx \sigma_W^2\mathbf{E}_{\theta_0^{hid}}\left[\varphi\left(f^{L-1}(x;\theta_0^{hid})\right)^T\varphi\left(f^{L-1}(x';\theta_0^{hid})\right)\right]+\sigma_b^2$$

So this model can be understood only through understanding this kernel! We will compute what this is in the next section on neural networks.

The more interesting case is the full network case where ALL the layers are trained. You can think of this as a random feature model where we train the features as we go!

### 1.2.3 Double Descent Curve

In simple example cases, by using probability theory to theoretically compute $K$, we can investigate how the model performance depends on $n_{par}$. Imagine the case where the training data is given as $y^{(i)} = \langle \beta, x^{(i)} \rangle$ for an unknown vector $\beta$. Then train the model using random features with $\varphi(x) = \max\{x, 0\}$, the ReLU nonlinearity. Here is a plot where they plot the test error for an unseen point $\mathbf{E}_{(x,y)\sim\mu}\left[\left(\mathbf{E}\left[f(x;\theta_\infty)\right] - y\right)^2\right]$ on the y-axis vs the ratio $\frac{n_{ex}}{n_{par}}$on the x-axis (They compute the error in the limit that both $n_{ex}$, $n_{par}, n_{in}$ all grow to infinity with the ratio being fixed.) The left plot is exactly the model we discussed (no regularization) while the right plot has a regularization term proportional to $\|\theta\|^2$ added to the loss function. Both plots are an example of the **double descent phenomenon** where the test error first goes down, then goes up, and then goes down again. This kind of phenomenon shows that in the extremely overparameterized case, the performance of the model can actually be very good. Unlike other statistical models, it seems that overfitting does not seem to be a big issue with these models. [MM19, HMRT19] have analysis of the random features regression model in this high dimensional setting. Here is a beautiful figure from [MM19]showing the double descent curve.

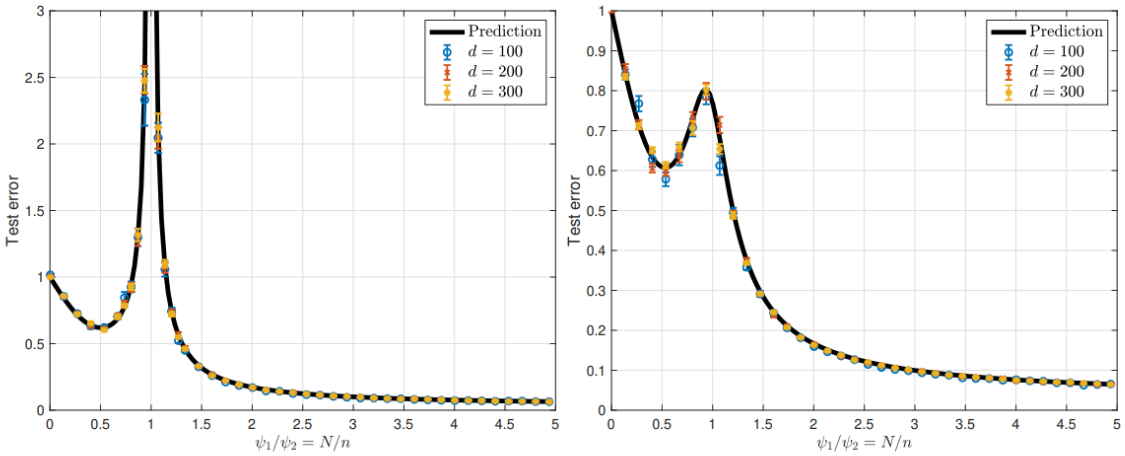


Figure 1: Random features ridge regression with ReLU activation ($\sigma = \max\{x, 0\}$). Data are generated via $y_i = \langle \boldsymbol{\beta}_1, \boldsymbol{x}_i \rangle$ (zero noise) with $\|\boldsymbol{\beta}_1\|_2^2 = 1$, and $\psi_2 = n/d = 3$. Left frame: regularization $\lambda = 10^{-8}$ (we didn't set $\lambda = 0$ exactly for numerical stability). Right frame: $\lambda = 10^{-3}$. The continuous black line is our theoretical prediction, and the colored symbols are numerical results for several dimensions $d$. Symbols are averages over 20 instances and the error bars report the standard error of the means over these 20 instances.

## 1.3 Remarks and Extensions

### 1.3.1 The Kernel Trick

*Remark* 11. Note that the results depends only on the kernel $K(\cdot, \cdot)$, but not actually on the features $\phi(\cdot)$ themselves. This means that there is an alternate method to do the regression that replaces the step of choosing feature vectors $\phi$ with only choosing a kernel $K(\cdot, \cdot)$. This kind of technique of working only with an abstract $K$ and forgetting about where it came from is sometimes called "the kernel trick". Here is what that means in our setting:

**Step 1:** Choose a kernel function $K : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ [6]

**Step 2:** For a given training set $\mathcal{X}, \mathcal{Y}$, define the estimator $\hat{f}(x) = \mathbf{E}\left[f(x;\theta_\infty)\right]$ from 5, which is given in terms of $K(\cdot, \cdot)$ only

This has the potential to be advantageous because it avoids dealing with the high dimension $n_{par}$ and everything is now instead in the "low" dimensions $n_{ex}$ and $n_{in}$. In the same vein, we have replaced the task of choosing $n_{par}$ features $\phi_i : \mathbb{R}^{n_{in}} \to \mathbb{R}$ with choosing a single kernel function $K : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$. Note that you still have to store and invert the matrix $K(\mathcal{X}, \mathcal{X})$ which might have other types of issues. Depending on the application and implementation, it might be better to compute only with $K(\cdot, \cdot)$ or by using the original features.

[6] There are some terms and conditions that the kernel $K$ must satisfy to guarantee everything will work like $K(x, x') = K(x', x)$, $K(x, x) \geq 0$. Essentially, $K(x, x')$ must satisfy all the axioms to make $K(x, x')$ a valid inner product. There is a theoretical result, known as Mercer's theorem, which says that every kernel $K$ that satisfies these reasonable conditions actually arises as $K(x, x') = \phi(x)^T \phi(x')$ for some (possibly complicated) set of features $\phi$. I won't go into the details here!

*Remark* 12. Even if one doesn't carry out the computation using $K(\cdot,\cdot)$, the explicit formulas from Theorem6 can be extremely useful for theoretical analysis. For example, we can measure how well the method overfits by examinging $f(x;\theta_\infty)$ for points $x$ outside the training set. As a concrete setup, one can suppose that $\mathcal{X},\mathcal{Y}$ are drawn independently from some distribution $\mu$, and then the generalization error $\mathbf{E}_{(x,y)\sim\mu}\left[\left(\mathbf{E}\left[f(x;\theta_\infty)\right]-y\right)^2\right]$ measures the average loss of the network to a test point outside of $\mathcal{X},\mathcal{Y}$. In some cases the formulas from Theorem 6 can be used to explicitly calculate this and understand generalization error.

*Remark* 13. There is also an equivalent Bayesian point of view on this where training the model is thought of as a Bayesian update of the weights according to the information given by $\mathcal{X},\mathcal{Y}$. I won't go into this here. You can find some of this in the book Gaussian Processes for Machine Learning by Rasmussen & Williams.

### 1.3.2 Extensions - Weights/Biases and Multi-dimensional output

*Remark* 14. One can also consider a model of which includes weights and biases so the model is now

$$f(x;\theta)=\frac{\sigma_W}{\sqrt{n_W}}W^T\phi\left(x\right)+\sigma_b b$$

where now $\theta=\{W\}\cup\{b\}$ includes all the weights and biases together, and where the initialization is now $W\sim\mathcal{N}(0,1)$ and $b\sim\mathcal{N}(0,1)$ are still independent standard Gaussians. This fits into the previous setup by scaling the features by $\sigma_W$ making one of the features a constant $\sigma_b$. The kernel in this case becomes:

$$K(x,x')=\nabla_\theta f(x;\theta)^T\nabla_\theta f(x;\theta)=\mathbf{E}\left[f(x;\theta_0)f(x';\theta_0)\right]=\frac{\sigma_W^2}{n_{feat}}\phi(x)^T\phi(x')+\sigma_b^2$$

Note that for everything to work out without any other changes, its important that the parameters are always initialized to standard Gaussians $\mathcal{N}(0,1)$, and the scaling by $\sigma_W,\sigma_b$ is put in the definition of the function $f(x;\theta)$ (as opposed to having $W\sim\mathcal{N}(0,\sigma_W^2)$ and $b\sim\mathcal{N}(0,\sigma_b^2)$ and omitting the constants $\sigma_W,\sigma_b$ from the definition of $f(x;\theta)$). The reason this is important is because there are **two parallel parts of the puzzle.** One part is about the gradient $\nabla_\theta f(x;\theta)^T\nabla_\theta f(x;\theta)$, and the other part is about the covariance $\mathbf{E}\left[f(x;\theta_0)f(x';\theta_0)\right]$. We want these two different pieces to gel together, which means we must always have the weights be $\mathcal{N}(0,1)$; changing the variance to something else scales the covariance term but not the gradient term.

*Remark* 15. One can also consider the output dimension to be a general $n_{out}$, rather than just 1 dimensional. In this case one needs to think of the kernel $K(x,x')=\phi^T(x)\phi(x')$ as as a map $K:\mathbb{R}^{n_{in}}\times\mathbb{R}^{n_{in}}\to\mathbb{R}^{n_{out}}\times\mathbb{R}^{n_{out}}$ [7]. Equally well, you can look at the individual components of the model $f(x;\theta)=[f_1(x;\theta),\ldots,f_{n_{out}}(x;\theta)]^T$ and set:

$$\begin{aligned}K_{ij}(x,x')&=\nabla_\theta f_i(x;\theta)^T\nabla_\theta f_j(x';\theta)\\&=\mathbf{E}\left[f_i(x;\theta)f_j(x';\theta)\right]\end{aligned}$$

For the application that we are going to look at it will turn out that the only elements which will be non-zero are going to be the diagonal elements $K_{ii}$. For this reason I'm always going to stick to thinking of kernels as being $\mathbb{R}$-valued and I'll use components. Note that some authors think of these things as being matrix valued with is why they sometimes have formulas involving the tensor product $\otimes$ in their formulas; I'm going to avoid this notation.

## 1.4 Proofs

### 1.4.1 Gradient Descent Dynamics - Proof of Theorem 6 Part 1

The first observation is to calculate the gradient $\nabla_\theta f(x;\theta)$ for this model using the definition (1). In this case, this is[8]

$$\nabla_\theta f(x;\theta)=\frac{1}{\sqrt{n_{par}}}\underset{n_{par}\times 1}{\phi(x)}\tag{6}$$

[7] You can see this from the formula $K(x,x')=\underset{n_{out}\times n_{ex}}{\phi(x)^T}\underset{n_{ex}\times n_{out}}{\phi(x)}$

[8] Note that I'm doing all the calculation as vectors/with matrix multiplication because the final answer can be cleanly written that way. It is a good exercise (and often how I actually did the computation!) to do it all explicitly in coordinates. E.g. write out $f(x;\theta)=\sum_{i=1}^{n_{par}}\theta_i\phi_i(x)$ so you can directly verify $\frac{\partial}{\partial\theta_i}f(x;\theta)=\phi_i(x)$

The miracle here is that the gradient does *not* depend on $\theta$ at all! This is the key feature that will eventually allow us to solve the model. The gradient flow dynamics can then be written using chain rule as:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\theta_t &= -\alpha \underset{n_{par}\times 1}{\nabla_\theta L(\theta_t)} \\ &= -\alpha \underset{n_{par}\times n_{ex}}{\nabla_\theta f(\mathcal{X};\theta)} \underset{n_{ex}\times 1}{(f(\mathcal{X};\theta_t)-\mathcal{Y})} \\ &= -\frac{\alpha}{\sqrt{n_{par}}} \underset{n_{par}\times n_{ex}}{\phi(\mathcal{X})} \underset{n_{ex}\times 1}{(f(\mathcal{X};\theta_t)-\mathcal{Y})}\end{aligned} \tag{7}$$

It is not clear how to solve this differential equation as the evolution of $\theta_t$ depends on the values $f(\mathcal{X};\theta_t)$, which we don't know because we need to solve for $\theta_t$! The trick to solving this is to look directly at how the function $f(x;\theta_t)$ evolves and solve for that first *without* directly solving for the parameters $\theta_t$. We can compute as follows by chain rule again and plugging in (6) and (7) to get

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}f(x;\theta_t) &= \underset{1\times n_{par}}{\nabla_\theta f(x;\theta_t)^T} \underset{n_{par}\times 1}{\frac{\mathrm{d}}{\mathrm{d}t}\theta_t} \\ &= -\frac{\alpha}{n_{par}} \underset{1\times n_{par}}{\phi(x)^T} \underset{n_{par}\times n_{ex}}{\phi(\mathcal{X})} \underset{n_{ex}\times 1}{(f(\mathcal{X};\theta_t)-\mathcal{Y})} \qquad (8) \\ &= -\alpha K(x,\mathcal{X}) \underset{n_{ex}\times 1}{(f(\mathcal{X};\theta_t)-\mathcal{Y})} \qquad (9)\end{aligned}$$

### 1.4.2 Solution to Dynamics - Proof of Theorem 6 Part 2

The nice formula for $\frac{\mathrm{d}}{\mathrm{d}t}f(x;\theta_t)$ is solveable to give the exact formula during training. As it is written now the evolution of $f(x;\theta_t)$ depends on $f(\mathcal{X};\theta_t)$. To "close the loop" and make this into something we can solve, we are going to plug in the training examples we have $x = x^{(1)}, \ldots, x = x^{(n_{ex})}$. You can think of doing this one at a time and stacking up the result into a big vector. In vector form this can be written succinctly in terms of the vector $f(\mathcal{X};\theta_t)$ which is the vector with $f(x^{(1)};\theta_t)\ldots, f(x^{(n_{ex})};\theta_t)$ to get

$$\begin{aligned}\underset{n_{ex}\times 1}{\frac{\mathrm{d}}{\mathrm{d}t}f(\mathcal{X};\theta_t)} &= -\frac{\alpha}{n_{par}} \underset{n_{ex}\times n_{par}}{\phi(\mathcal{X})^T} \underset{n_{par}\times n_{ex}}{\phi(\mathcal{X})} \underset{n_{ex}\times 1}{(f(\mathcal{X};\theta_t)-\mathcal{Y})} \\ &= -\alpha \underset{n_{ex}\times n_{ex}}{K(\mathcal{X},\mathcal{X})} \underset{n_{ex}\times 1}{(f(\mathcal{X};\theta_t)-\mathcal{Y})}\end{aligned}$$

where we recall the matrix $K(\mathcal{X},\mathcal{X}) = \frac{1}{n_{par}}\phi(\mathcal{X})^T\phi(\mathcal{X}) = \left[K(x^{(i)},x^{(j)})\right]_{i,j=1}^{n_{ex}}$. Finally, our differential is a closed system where the evolution of the vector $f(\mathcal{X};\theta_t)$ depends on $f(\mathcal{X};\theta_t)$ itself. Because the matrix $K(\mathcal{X},\mathcal{X})$ is constant and doesn't depend on $t$, this type of equation is easily solved. To make the solution more transparent, let us do the substitution $v_t = f(\mathcal{X};\theta_t) - \mathcal{Y}$. Notice that $\frac{\mathrm{d}}{\mathrm{d}t}f(\mathcal{X};\theta_t) = \frac{\mathrm{d}}{\mathrm{d}t}v_t$. So the differential equation for $v_t$ is simply

$$\frac{\mathrm{d}}{\mathrm{d}t}v_t = -\alpha K(\mathcal{X},\mathcal{X})v_t$$

which is just a constant coefficient vector differential equation[9]. This has solution given by the matrix exponential

$$v_t = e^{-\alpha t K(\mathcal{X},\mathcal{X})}v_0$$

Substituting $v_t$ back now gives the solution for $f(\mathcal{X};\theta_t)$

$$f(\mathcal{X};\theta_t) = \mathcal{Y} + e^{-\alpha t K(\mathcal{X},\mathcal{X})}\left(f\left(\mathcal{X};\theta_0\right) - \mathcal{Y}\right) \tag{10}$$

Right away we can notice that as long as $K(\mathcal{X},\mathcal{X})$ is not degenerate, then $\lim_{t\to\infty} e^{-\alpha t K(\mathcal{X},\mathcal{X})} = 0$ and so $\lim_{t\to\infty} f(\mathcal{X};\theta_t) = \mathcal{Y}$, or in other words that our training data is fit perfectly.

We can finally plug the solution (10) into the evolution of $f(x;\theta_t)$ from (8) to get

$$\frac{\mathrm{d}}{\mathrm{d}t}f(x;\theta_t) = -\alpha K(x,\mathcal{X})e^{-\alpha t K(\mathcal{X},\mathcal{X})}\left(f\left(\mathcal{X};\theta_0\right) - \mathcal{Y}\right)$$

[9] This differential equation is the vector/matrix version of the scalar differential equation $y' = -\alpha y$ which has solution $y(t) = e^{-\alpha t}y(0)$

where $K(x, \mathcal{X}) = \frac{1}{\sqrt{n_{par}}}\phi(x)^T\phi(\mathcal{X})$. This gives $\frac{\mathrm{d}}{\mathrm{d}t}f(x;\theta_t)$ explicitly so we only have to integrate with respect to $t$ to solve[10] to get

$$f(x;\theta_t) = f(x;\theta_0) + K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\left(e^{-\alpha t K(\mathcal{X},\mathcal{X})} - I\right)\left(f\left(\mathcal{X};\theta_0\right) - \mathcal{Y}\right) \tag{11}$$

We can now easily compute the limit assuming that $K(\mathcal{X},\mathcal{X})$ is non-degenerate so that $\lim_{t\to\infty} e^{-\alpha t K(\mathcal{X},\mathcal{X})} = 0$

$$\begin{aligned} f(x;\theta_\infty) &= \lim_{t\to\infty} f(x;\theta_t) \\ &= f(x;\theta_0) - K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\left(f\left(\mathcal{X};\theta_0\right) - \mathcal{Y}\right) \end{aligned} \tag{12}$$

This gives the final solution $f(x;\theta_\infty)$ explicitly in terms of the initialization $f(x;\theta_0)$.

Note that we can get an explicit solution for $\theta_t$ in the same way by plugging in the solution $f(\mathcal{X};\theta_t)$ into the differential (7) to get the solution

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\theta_t &= -\alpha\frac{1}{\sqrt{n_{par}}}\phi(\mathcal{X})e^{-\alpha t K(\mathcal{X},\mathcal{X})}\left(f\left(\mathcal{X};\theta_0\right) - \mathcal{Y}\right) \\ \implies \theta_t - \theta_0 &= \frac{1}{\sqrt{n_{par}}}\phi(\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\left(e^{-\alpha t K(\mathcal{X},\mathcal{X})} - I\right)\left(f\left(\mathcal{X};\theta_0\right) - \mathcal{Y}\right) \\ \implies \theta_\infty - \theta_0 &= \frac{1}{\sqrt{n_{par}}}\phi(\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\left(\mathcal{Y} - f(\mathcal{X};\theta_0)\right) \end{aligned} \tag{13}$$

Note that the amount any individual weights change scales like $O\left(\frac{1}{\sqrt{n_{par}}}\right)$: this means that as we have more and more features, the amount each parameters need to change during training shrinks. Note also by computing $\left(\theta_\infty - \theta_0\right)^T\left(\theta_\infty - \theta_0\right)$ using 13 we get that:

$$\underset{1\times 1}{\|\theta_\infty - \theta_0\|^2} = \underset{1\times n_{ex}}{\left(\mathcal{Y} - f(\mathcal{X};\theta_0)\right)^T}\underset{n_{ex}\times n_{ex}}{K^{-1}(\mathcal{X},\mathcal{X})}\underset{n_{ex}\times 1}{\left(\mathcal{Y} - f(\mathcal{X};\theta_0)\right)}$$

meaning that the total norm of the change stays $O(1)$ even as the number of parameters grows.

### 1.4.3 Distribution at initialization - Proof of Proposition 7

The key to part 3 is to notice that $f(x;\theta_0)$ is a mean zero Gaussian $\theta_0$ which is simply multiplied by the features $\phi(x)$. Gaussian are preserved under this type of multiplication.

**Fact 16.** *If $X$ is Gaussian random variable $X \sim \mathcal{N}(\mu_X, \Sigma_X)$ and if $Y = a + BX$ then $Y$ is Gaussian too with $Y \sim \mathcal{N}(a + B\mu_X, B\Sigma_X B^T)$*

**Proposition 17.** *Recall that the initial parameters $\theta_0$ are independent Gaussian with $\theta_0 \sim \mathcal{N}(0, \sigma I^{n_{par}})$. For any fixed point $x \in \mathbb{R}^{n_{in}}$, the random variable $f(x;\theta_0)$ is Gaussian with mean and variance given as follows*

$$\mathbf{E}\left[f(x;\theta_0)\right] = 0,\ \mathbf{Var}\left[f(x;\theta_0)\right] = K(x,x)$$

*Moreover, for any finite collection of test points $\mathcal{X}_{test} = \left(x_{test}^{(1)}, \ldots, x_{test}^{(n_{test})}\right)$, the vector $f(\mathcal{X}_{test};\theta_0) = \left[f(x_{test}^{(i)};\theta_0)\right]_{i=1}^{n_{test}} \in \mathbb{R}^{n_{test}}$ is Gaussian with mean $\mathbf{E}\left[f(\mathcal{X}_{test};\theta_0)\right] = 0$ and covariance matrix*

$$\mathbf{Cov}\left[f(\mathcal{X}_{test};\theta_0)\right] = K(\mathcal{X}_{test},\mathcal{X}_{test}) = \left[K(x_{test}^{(i)}, x_{test}^{(j)})\right]_{i,j=1}^{n_{test}}$$

*Proof.* This follows directly from Fact 16 since $\theta_0 \sim \mathcal{N}(0, \sigma I^{n_{par}})$ and $f(x;\theta_0) = \frac{1}{\sqrt{n_{par}}}\phi(x)^T\theta_0$ is a linear transformation of $\theta_0$. Note that the variance of $f(x;\theta_0)$ arises from $K(x,x) = \frac{1}{n_{par}}\phi(x)^T\phi(x)$. Similarly, $f(\mathcal{X}_{test};\theta_0) = \frac{1}{\sqrt{n_{par}}}\phi(\mathcal{X}_{test})^T\theta_0$ is also linear transformation of the original weights. In this case the covariance matrix for $\phi(\mathcal{X}_{test})^T\theta_0$ is now given by $\frac{\sigma}{n_{par}}\phi(\mathcal{X}_{test})^T\phi(\mathcal{X}_{test}) = K(\mathcal{X}_{test},\mathcal{X}_{test})$ as desired. □

[10] Once again, the vector/matrix integration we are doing looks slightly complicated, but it is the analog of the scalar integration $y' = e^{Ct} \implies y(t) = \frac{1}{C}\left(e^{Ct} - 1\right) + y(0)$

### 1.4.4 Distribution after training - Proof of Theorem 9 Part 1

*Proof.* Since $f(x;\theta_\infty)$ is an explicit linear transformation of $f(x;\theta_0)$ this will be Gaussian too, and we can compute its mean and covariance. The calculation that $\mathbf{E}[f(x;\theta_\infty)] = K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\mathcal{Y}$ follows immediately by taking expectation of both side of 12 since $\mathbf{E}[f(x,\theta_0)] = 0$ and $\mathbf{E}[f(\mathcal{X};\theta_0)] = 0$. Theorem 9 now follows by considering the test set of points $\{x\} \cup \mathcal{X}$ which consists of a single test point $x$ and the training data $\mathcal{X}$. By Proposition 17, this vector can be understood as a mean zero Gaussian vector whose covariance has a nice block structure:

$$\mathbf{E}\left(\begin{bmatrix} f(x;\theta_0) \\ f(\mathcal{X};\theta_0) \end{bmatrix}\right) = 0 \ \mathbf{Cov}\left(\begin{bmatrix} f(x;\theta_0) \\ f(\mathcal{X};\theta_0) \end{bmatrix}\right) = \begin{bmatrix} K(x,x) & K(x,\mathcal{X}) \\ 1\times 1 & 1\times n_{ex} \\ K(\mathcal{X},x) & K(\mathcal{X},\mathcal{X}) \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \tag{14}$$

Now the result of Part 1 and Part 2 of Theorem 6 (e.g. 12) can be interpreted as giving the vector $\begin{bmatrix} f(x;\theta_\infty) \\ f(\mathcal{X};\theta_\infty) \end{bmatrix}$ as a linear transformation $\begin{bmatrix} f(x;\theta_0) \\ f(\mathcal{X};\theta_0) \end{bmatrix}$ which is nicely written in block matrix form as follows:

$$\begin{bmatrix} f(x;\theta_\infty) \\ f(\mathcal{X};\theta_\infty) \end{bmatrix} = \begin{bmatrix} 1 & -K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X}) \\ 1\times 1 & 1\times n_{ex} \quad n_{ex}\times n_{ex} \\ 0 & 0 \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \begin{bmatrix} f(x;\theta_0) \\ f(\mathcal{X};\theta_0) \end{bmatrix} + \begin{bmatrix} K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\mathcal{Y} \\ 1\times n_{ex} \quad n_{ex}\times n_{ex} \quad n_{ex}\times 1 \\ 0 \\ n_{ex}\times 1 \end{bmatrix} \tag{15}$$

Hence by Fact 16, we can compute the mean and covariance of $\begin{bmatrix} f(x;\theta_\infty) \\ f(\mathcal{X};\theta_\infty) \end{bmatrix}$ from the mean and covariance of $\begin{bmatrix} f(x;\theta_0) \\ f(\mathcal{X};\theta_0) \end{bmatrix}$ given in 14 as follows:

$$\mathbf{E}\left(\begin{bmatrix} f(x;\theta_\infty) \\ f(\mathcal{X};\theta_\infty) \end{bmatrix}\right) = \begin{bmatrix} K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})\mathcal{Y} \\ 1\times n_{ex} \quad n_{ex}\times n_{ex} \quad n_{ex}\times 1 \\ \mathcal{Y} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{Cov}\left(\begin{bmatrix} f(x;\theta_\infty) \\ f(\mathcal{X};\theta_\infty) \end{bmatrix}\right) &= \begin{bmatrix} 1 & -K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X}) \\ 1\times 1 & 1\times n_{ex} \quad n_{ex}\times n_{ex} \\ 0 & 0 \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \begin{bmatrix} K(x,x) & K(x,\mathcal{X}) \\ 1\times 1 & 1\times n_{ex} \\ K(\mathcal{X},x) & K(\mathcal{X},\mathcal{X}) \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1\times 1 & 1\times n_{ex} \\ -K^{-1}(\mathcal{X},\mathcal{X})K(\mathcal{X},x) & 0 \\ n_{ex}\times n_{ex} \quad n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \\
&= \begin{bmatrix} 1 & -K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X}) \\ 1\times 1 & 1\times n_{ex} \quad n_{ex}\times n_{ex} \\ 0 & 0 \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \begin{bmatrix} K(x,x) - K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})K(\mathcal{X},x) & 0 \\ 1\times 1 \quad 1\times n_{ex} \quad n_{ex}\times n_{ex} \quad n_{ex}\times 1 & 1\times n_{ex} \\ 0 & 0 \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix} \\
&= \begin{bmatrix} K(x,x) - K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})K(\mathcal{X},x) & 0 \\ 1\times 1 \quad 1\times n_{ex} \quad n_{ex}\times n_{ex} \quad n_{ex}\times 1 & 1\times n_{ex} \\ 0 & 0 \\ n_{ex}\times 1 & n_{ex}\times n_{ex} \end{bmatrix}
\end{aligned}$$

(Note that its not a surprise that the covariance matrix blocks involving $f(\mathcal{X};\theta_\infty)$ are zero as because we know that $f(\mathcal{X};\theta_\infty) = \mathcal{Y}$ exactly so it has no variance at all!) From this we can read off the variance of $f(x;\theta_\infty) = K(x,x) - K(x,\mathcal{X})K^{-1}(\mathcal{X},\mathcal{X})K(\mathcal{X},x)$ from the top-left block. The same technique works to compute the covariance matrix for any arbitrary test set $\mathcal{X}_T$ or for $f(x;\theta_t)$ at any arbitrary time $0 < t < \infty$ using (11). □

### 1.4.5 Conditioning Gaussian Vectors - Proof of Theorem 9 Part 2

Finally, the result about conditioning follows from another standard fact about conditional Gaussian distributions. The mean and covariance of our final model $f(x;\theta_\infty)$ happens to be precisely that of the conditional Gaussian distribution. The mean and variance of the conditional Gaussian model comes from the following calculation:

**Fact 18.** *Let $n_1$ and $n_2$ be integers, and consider the Gaussian vector of dimension $n_1+n_2$, $X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \in \mathbb{R}^{n_1+n_2}$, where $X_1 \in \mathbb{R}^{n_1}$ and $X_2 \in \mathbb{R}^{n_2}$ which has means*

$$\mathbf{E}[X] = \mu = \begin{bmatrix} \mathbf{E}[X_1] \\ \mathbf{E}[X_2] \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

*with $\mu \in \mathbb{R}^{n_1+n_2}$ and $\mu_1 \in \mathbb{R}^{n_1}$, $\mu_2 \in \mathbb{R}^{n_2}$, and $(n_1+n_2)\times(n_1+n_2)$ covariance matrix $\Sigma$ which we decompose into blocks*

$$\mathbf{Cov}[X] = \underset{(n_1+n_2)\times(n_1+n_2)}{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ n_1\times n_1 & n_1\times n_2 \\ \Sigma_{21} & \Sigma_{22} \\ n_2\times n_1 & n_2\times n_2 \end{bmatrix}$$

*Then, for any fixed vector $x \in \mathbb{R}^{n_2}$, conditioned on the event $\{X_2 = x\}$, the conditional distribution of $X_1 \in \mathbb{R}^{n_1}$ is a Gaussian with mean and covariance structure given by:*

$$\underset{n_1 \times 1}{\mathbf{E}}\left[X_1 \,|X_2 = x\right] = \underset{n_1 \times 1}{\mu_1} + \underset{n_1 \times n_2}{\Sigma_{12}} \underset{n_2 \times n_2}{\Sigma_{22}^{-1}} \underset{n_2 \times 1}{(x - \mu_2)}$$

$$\underset{n_1 \times 1}{\mathbf{Cov}}\left[X_1 \,|X_2 = x\right] = \underset{n_1 \times 1}{\Sigma_{11}} - \underset{n_1 \times n_2}{\Sigma_{12}} \underset{n_2 \times n_2}{\Sigma_{22}^{-1}} \underset{n_2 \times n_1}{\Sigma_{21}}$$

# 2 Neural Net Gaussian Processes and the Neural Tangent Kernel

| Notation | Space object lives in | Definition |
|---|---|---|
| $L$ | $\mathbb{N}$ | $L$= Number of hidden layers in the neural network |
| $n_1, \ldots, n_L$ | $\mathbb{N}$ | $n_i$ =Width of the $i$-th hidden layer |
| $n_{in} = n_0$ | $\mathbb{N}$ | $n_{in} = n_0$ =input dimension, also sometimes known as the 0-th layer of the network |
| $n_{out} = n_{L+1}$ | $\mathbb{N}$ | $n_{out} = n_{L+1}$ =output dimension, also sometimes known as the $L+1$-st layer of the network |
| $W^\ell$ | $\mathbb{R}^{n_\ell \times n_{\ell-1}}$ | Weight matrix connecting layer $\ell$ to $\ell - 1$ |
| $b^\ell$ | $\mathbb{R}^{n_\ell}$ | Bias vector for layer $\ell$ |
| $\sigma_W$ | $\mathbb{R}$ | Standard deviation of weight matrix entries |
| $\sigma_b$ | $\mathbb{R}$ | Standard deviation of bias entries |
| $\varphi(x)$ | $\varphi : \mathbb{R} \to \mathbb{R}$ OR $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ | Non-linear activation function of the network. Note that we apply this function entry-wise when it is applied to vectors. |
| $x$ | $\mathbb{R}^{n_{in}}$ | Generic point in the input space $\mathbb{R}^{n_{in}}$ |
| $\theta$ | $\mathbb{R}^{n_{par}}$ | Vector of all the parameters of the model (e.g. all the weights and biases together) |
| $\theta_t$ | $\mathbb{R}^{n_{par}}$ | Parameters at time $t$ along some sequence of parameters (e.g. how parameters evolve under gradient descent) |
| $f(x;\theta)$ | $f : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{par}} \to \mathbb{R}$ | Model function at generic input $x$ and parameters $\theta$ (e.g. a regression model or a neural network) |
| $\nabla_\theta f(x;\theta)$ | $\nabla_\theta f : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{par}} \to \mathbb{R}^{n_{par}}$ | Gradient of the model w.r.t the parameters thought of as a vector in $\mathbb{R}^{n_{par}}$, $(\nabla_\theta f)_i (x;\theta) = \frac{\partial f}{\partial \theta_i}(x;\theta)$ for $1 \leq i \leq n_{par}$ |
| $f(\mathcal{X};\theta)$ | $f(\mathcal{X};\cdot) : \mathbb{R}^{n_{par}} \to \mathbb{R}^{n_{ex}}$ | Stacked vector of the output for all the examples $f(\mathcal{X};\theta) = \begin{bmatrix} f(x^{(1)};\theta) \\ \vdots \\ f(x^{(n_{ex})};\theta) \end{bmatrix} \in \mathbb{R}^{n_{ex}}$ |
| $\Sigma^\ell(x, x')$ | $\Sigma : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ | The Neural Net Gaussian Process kernel. Encodes the variance of the $\ell$-th layer of the network. |
| $\Theta^\ell(x, x')$ | $\Sigma : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ | The Neural Tangent Kernel. Encodes the evolution of the gradient descent for the $\ell$-th layer of the network. |

## 2.1 Main idea - Linear Approximation of a Neural Network

The point of this section is to argue that **in this limit, where the intermediate layer widths become very wide, neural networks behave a lot like the random feature regression model**. This might be surprising because from their definition neural networks seem a lot more complicated than the feature regression model....the most important difference being that the feature regression is **linear** in the parameters, while the neural network is definitely **non-linear**.

One way to see why a wide neural network behaves like a linear feature regression model is to take the Taylor series expansion of the neural network with respect to the weights. For a given initialization $\theta_0$, define the linearization $f^{lin}(x;\theta)$ by

$$f^{lin}(x;\theta) := f(x;\theta_0) + \nabla_\theta f(x;\theta_0)^T (\theta - \theta_0)$$

Clearly $f(x;\theta_0) = f^{lin}(x;\theta_0)$ and in fact, the suprising truth is that $f(x;\theta_t) \approx f^{lin}(x;\theta_t)$ for all times $t$ in wide neural neworks! The approximation $f(x;\theta) \approx f^{lin}(x;\theta)$ turns out to be justified because during training each individual

parameter of $\theta_t - \theta_0 = O\left(\frac{1}{\sqrt{n}}\right)$, where $n$ is the width of the intermediate layers. (This is very similar to the feature regression model where we showed that $\theta_t - \theta_0$ was proportional to $\frac{1}{\sqrt{n_{par}}}$.) This is useful because see can see from its definition that $f^{lin}(x;\theta)$ is a feature regression model (plus some random initialization) with features given by:

$$\phi(x;\theta_0) = \nabla_\theta f(x;\theta_0)$$

Note that these features are **random features** because they depend on the initialization $\theta_0$. Even though the individual features are random, the thing that actually matters is the kernel $\phi(x,\theta_0)^T\phi(x,\theta_0)$. By our analysis of the random feature model, we know that as the number of parameters gets large, they tend to be averaged out. The relevant kernel in the limit will be

$$\mathbf{E}\left[\nabla_\theta f(x;\theta_0)^T \nabla_\theta f(x';\theta_0)\right]$$

This is precisely the kernel that drives the evolution of a neural network in the infinite width limit! One of the main results of [LXS+19] is that this approximation becomes more and more accurate as the size of the networks increase in size. You can also understand the size of the error in terms of the difference between the function $f$ and the linear approximation $f^{lin}$.

**Theorem 19.** *[From [LXS+19]] Suppose the hidden layer widths are all equal so that $n_1 = \ldots = n_L = n$. Let $f^{lin}(x;\theta)$ be the linear approximation to the network. Let $\theta_t$ be the evolution of the parameters under gradient flow. Then the following holds for all time $t \geq 0$ :*

$$\begin{aligned}
\left\|f^{lin}(x;\theta_t) - f(x;\theta_t)\right\| &= O\left(\frac{1}{\sqrt{n}}\right) \\
\|\theta_t - \theta_0\| &= O(1) \\
\left\|\nabla_\theta f(x;\theta_t)^T \nabla_\theta f(x';\theta_t) - \nabla_\theta f(x;\theta_0)^T \nabla_\theta f(x';\theta_0)\right\| &= O\left(\frac{1}{\sqrt{n}}\right)
\end{aligned}$$

## 2.2 Detailed Results

### 2.2.1 Definition of Deep Neural Network

**Definition 20.** [A deep neural network] A deep neural net with input dimension $n_{in}$, output dimension $n_{out}$, with $L$ hidden layers, intermediate layer widths $n_1, \ldots, n_L$, and non-linearity function $\varphi : \mathbb{R} \to \mathbb{R}$ is defined as follows. The output of the intermediate layers of the neural network $f^\ell(x;\theta) \in \mathbb{R}^{n_\ell}$ and $h^\ell(x;\theta) \in \mathbb{R}^{n_\ell}$ for $1 \leq \ell \leq L+1$ are defined recursively using parameters $\theta = \left\{W^\ell, b^\ell\right\}_{\ell=1}^{L+1}$ where weight matrices $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and bias vectors $b^\ell \in \mathbb{R}^{n_\ell}$ by

$$\begin{aligned}
f^1(x;\theta) &= \frac{\sigma_W}{\sqrt{n_1}} \underset{n_1 \times n_{in}}{W^1} \underset{n_{in} \times 1}{x} + \sigma_b \underset{n_1 \times 1}{b^1} \\
h^\ell(x;\theta) &= \varphi\left(h^\ell(x;\theta)\right) \text{ (applied entry-wise)} \\
f^{\ell+1}(x;\theta) &= \frac{\sigma_W}{\sqrt{n_\ell}} \underset{n_{\ell+1} \times n_\ell}{W^{\ell+1}} \underset{n_\ell \times 1}{h^\ell(x;\theta)} + \sigma_b \underset{n_{\ell+1} \times 1}{b^{\ell+1}}
\end{aligned} \tag{16}$$

The $f$ functions are sometimes refereed to as the "pre-activation outputs" and the $h$ functions are the "post-activation outputs" of the layers. The output of the network is just defined to be the last layer output

$$f(x;\theta) = f^{L+1}(x;\theta)$$

with the convention that $n_{L+1} = n_{out}$. As with the regression model, the scaling of $\frac{1}{\sqrt{n_\ell}}$ is chosen so that we can take the limit $n_1, \ldots, n_L \to \infty$ and in this limit $f(x;\theta)$ will makes sense. At initialization, $\theta_0$ is set so that each of the weights and biases are independent Gaussian vectors $W_{ij}^\ell \sim \mathcal{N}(0,1)$ and $b_i^\ell \sim \mathcal{N}(0,1)$ where $\sigma_W$ and $\sigma_b$ are some fixed variance parameters.

### 2.2.2 Neural Network Gaussian Processes

An important difference between the feature regression model and neural networks is that two relevant kernels that appear are NOT equal.

$$\begin{aligned}
\mathbf{E}\left[f(x;\theta_0) f(x';\theta_0)\right] &\approx \Sigma^{L+1}(x,x') \\
\nabla_\theta f(x;\theta_0)^T \nabla_\theta f(x';\theta_0) &\approx \Theta^{L+1}(x,x')
\end{aligned}$$

This means that the final formulas for the variance $f(x;\theta_\infty)$ is quite a bit more messy as there is no nice cancellation, and the interpretation as a conditioned Gaussian model does not work as it did in the feature regression model.

The first observation is that, as with the feature regression model, if the weights are initialized to Gaussians, then the output of the network is also Gaussian. This is very similar to the feature regression model. Moreover, in the limit that the intermediate widths go to infinity, the covariance structure can be calculated in terms of the non-linearity $\varphi$. The kernel $\Sigma$ is sometimes called the Neural Network Gaussian Process "NNGP" kernel.

**Proposition 21.** *In the limit that the hidden layer widths $n_1, n_2, \ldots, n_L \to \infty$,[11] there is a kernel $\Sigma^\ell(x,x')$ for each $1 \leq \ell \leq L+1$ so that for any fixed point $x \in \mathbb{R}^{n_{in}}$, the random variable $f^\ell(x;\theta_0)$ is Gaussian with mean and variance given as*

$$\mathbf{E}\left[f^\ell(x;\theta_0)\right] = 0, \ \mathbf{Var}\left[f^\ell(x;\theta_0)\right] = \Sigma^\ell(x,x)$$

*Moreover, for any finite collection of test points $\mathcal{X}_{test} = \left(x_{test}^{(1)}, \ldots, x_{test}^{(n_{test})}\right)$, the vector $f^\ell(\mathcal{X}_{test};\theta_0) = \left[f^\ell(x_{test}^{(i)};\theta_0)\right]_{i=1}^{n_{test}} \in \mathbb{R}^{n_{test}}$ is Gaussian with mean $\mathbf{E}\left[f^\ell(\mathcal{X}_{test};\theta_0)\right] = 0$ and covariance matrix*

$$\mathbf{Cov}\left[f^\ell(\mathcal{X}_{test};\theta_0)\right] = \Sigma^\ell(\mathcal{X}_{test},\mathcal{X}_{test}) = \left[\Sigma^\ell(x_{test}^{(i)}, x_{test}^{(j)})\right]_{i,j=1}^{n_{test}}$$

*There is a recursive formula that gives $\Sigma^\ell$ in terms $\Sigma^{\ell-1}$ by*

$$\Sigma^1(x,x') = \sigma_W^2 \langle x, x'\rangle + \sigma_b^2$$
$$\Sigma^\ell(x,x') = \sigma_W^2 \mathbf{E}\left[\varphi(Z)^T \varphi(Z')\right] + \sigma_b^2$$

*where $Z, Z'$ are mean 0 Gaussians with covariance structure $\mathbf{Cov}(Z,Z') = \begin{bmatrix} \Sigma^{\ell-1}(x,x) & \Sigma^{\ell-1}(x',x) \\ \Sigma^{\ell-1}(x,x') & \Sigma^{\ell-1}(x',x') \end{bmatrix}$.*

### 2.2.3 The Neural Tangent Kernel

**Theorem 22.** *There is a kernel $\Theta(x,x')$ known as the **neural tangent kernel** so that in the limit that the hidden layer widths $n_1, \ldots, n_L \to \infty$ we have*

1. *We have the convergence*

$$\nabla_\theta f(x;\theta_0)^T \nabla_\theta f(x';\theta_0) \to \Theta(x,x')$$
$$\text{and } \mathbf{E}\left[\nabla_\theta f(x;\theta_0)^T \nabla_\theta f(x';\theta_0)\right] \to \Theta(x,x')$$

2. *The limiting network satisfies the evolution:*

$$\frac{d}{dt} f(x;\theta_t) = -\alpha \Theta(x,\mathcal{X})\left(f(\mathcal{X};\theta) - \mathcal{Y}\right)$$

*whose solution is given by formulas similar to Theorem 6 in terms of the kernel $\Theta$, e.g. $\mathbf{E}\left[f(x;\theta_\infty)\right] = \Theta(x,\mathcal{X})\Theta^{-1}(\mathcal{X},\mathcal{X})\mathcal{Y}$.*

3. *There is a kernel $\Theta^\ell$ for each layer of the network; the kernel $\Theta$ is simply the output for the last layer is simply $\Theta = \Theta^{L+1}$. The recursive formula giving the kernel $\Theta^\ell$ in terms of the previous layer $\Theta^{\ell-1}$ is as follows: Let $\Sigma^\ell(x,x')$ be the limiting covariance kernels of the neural network as in Proposition 21. Also define $\dot{\Sigma}^\ell(x,x')$ by a similar recursion using the derivative $\dot{\varphi}(x) = \frac{d}{dx}\varphi(x)$*

$$\Sigma^\ell(x,x') = \sigma_W^2 \mathbf{E}\left[\varphi(Z)^T \varphi(Z')\right] + \sigma_b^2$$
$$\dot{\Sigma}^\ell(x,x') = \sigma_W^2 \mathbf{E}\left[\dot{\varphi}(Z)^T \dot{\varphi}(Z')\right]$$

*where where $Z, Z'$ are mean 0 Gaussians with covariance structure given by the old kernel $\Sigma^{\ell-1}(x,x')$: $\mathbf{Cov}(Z,Z') = \begin{bmatrix} \Sigma^{\ell-1}(x,x) & \Sigma^{\ell-1}(x',x) \\ \Sigma^{\ell-1}(x,x') & \Sigma^{\ell-1}(x',x') \end{bmatrix}$. The kernels $\Theta^\ell$ are defined by the recursion:*

$$\Theta^1(x,x') = \sigma_W^2 \langle x, x'\rangle + \sigma_b^2$$
$$\Sigma^1(x,x') = \sigma_W^2 \langle x, x'\rangle + \sigma_b^2$$
$$\Theta^\ell(x,x') = \Theta^{\ell-1}(x,x')\dot{\Sigma}^\ell(x,x') + \Sigma^\ell(x,x')$$

[11] More specifically we are doing the limits in order with $n_1 \to \infty$ first, then $n_2 \to \infty$ and so on.

## 2.3 Proofs

### 2.3.1 Proof of Neural Network Gaussian Process

Proposition 21 is proven with induction by showing that the Gaussian process property passes from each layer to the next layer. We give this property a name, and then prove that it propagates through the layers.

**Definition 23.** We say that a random function $f(x) \sim GP(m(x), \Sigma(x, x'))$ is a "Gaussian process" to mean that:

1. For each $x \in \mathbb{R}^{in}$, $f(x)$ is Gaussian with mean and variance given by:

$$\begin{aligned} \mathbf{E}\left[f(x)\right] &= m(x) \\ \mathbf{Var}\left[f(x)\right] &= \Sigma(x, x) \end{aligned}$$

2. For any finite set $\mathcal{X}_{test} = \left(x_{test}^{(1)}, \ldots, x_{test}^{(n_{test})}\right)$ the vector $f(\mathcal{X}_{test}) \in \mathbb{R}^{n_{test}}$ is a Gaussian vector with covariance matrix

$$\begin{aligned} \mathbf{E}\left[f(\mathcal{X}_{test})\right] &= m\left(\mathcal{X}_{test}\right) \\ \mathbf{Var}\left[f(\mathcal{X}_{test})\right] &= \Sigma(\mathcal{X}_{test}, \mathcal{X}_{test}) = \left[\Sigma(x_{test}^{(i)}, x_{test}^{(j)})\right]_{i,j=1}^{n_{test}} \end{aligned}$$

**Proposition 24.** *[Propogation of the NNGP kernel] Let $x \in \mathbb{R}^{n_{in}}$. Suppose $f^{old} : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{old}}$ is a random function so that each of its components satisfy $f_i^{old}(x) \sim GP(0, \Sigma^{old}(x, x'))$ and the components $f_1^{old}, \ldots, f_{n_{old}}^{old}$ are independent. Define the random output function $f^{new} : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{new}}$ by*

$$f^{new}(x) = \frac{\sigma_W}{\sqrt{n_{old}}} \underset{n_{new} \times n_{old}}{W} \varphi\left(\underset{n_{hid} \times 1}{f^{hid}(x)}\right) + \sigma_b \underset{n_{old} \times 1}{b},$$

*Suppose also that $W$ is an $n_{new} \times n_{old}$ random matrix and $b$ is a random vector of size $n_{old}$ whose entries are iid standard Gaussians. Then, in the limit $n_{hid} \to \infty$, each component $f_j^{new}(x) \sim GP(0, \Sigma^{new}(x, x'))$ is a GP of mean 0 and covariance structure:*

$$\begin{aligned} \Sigma^{new}(x, x') &= \sigma_W^2 \mathbf{E}\left[\varphi\left(f_1^{old}(x)\right)^T \varphi\left(f_1^{old}(x')\right)\right] + \sigma_b^2 \\ &= \sigma_W^2 \mathbf{E}\left[\varphi\left(Z\right)^T \varphi\left(Z'\right)\right] + \sigma_b^2 \end{aligned}$$

*where $Z, Z'$ are mean 0 Gaussians with covariance structure $\mathbf{Cov}\left(Z, Z'\right) = \begin{bmatrix} \Sigma^{old}(x, x) & \Sigma^{old}(x', x) \\ \Sigma^{old}(x, x') & \Sigma^{old}(x', x') \end{bmatrix}$. Moreover, the components $f_i^{old}, f_j^{old}$ are independent random functions for $i \neq j$.*

*Proof.* Conditioned on the values of $f^{old}(x)$, the function each component $f_j^{new}(x)$ is precisely a feature regression model with $n_{hid}$ features $\phi(x) = \varphi\left(f^{hid}(x)\right)$. Therefore, this is a Gaussian process with mean 0 and covariance $K(x, x') = \frac{\sigma_b^2}{n_{hid}} \phi(x)^T \phi(x') + \sigma_b^2 = \frac{\sigma_W^2}{n_{hid}} \varphi\left(f^{hid}(x; \theta)\right)^T \varphi\left(f^{hid}(x'; \theta)\right) + \sigma_b^2$. Now notice that $\frac{1}{n_{old}} \varphi\left(f^{old}(x)\right)^T \varphi\left(f^{old}(x')\right)$ is actually a sum over $n_{old}$ random variables:

$$\frac{1}{n_{old}} \varphi\left(f^{old}(x; \theta)\right)^T \varphi\left(f^{old}(x'; \theta)\right) = \frac{1}{n_{old}} \sum_{i=1}^{n_{old}} \varphi\left(f_i^{old}(x; \theta)\right) \varphi\left(f_i^{old}(x'; \theta)\right)$$

By the assumption on $f^{old}$ these are independent random variables! So by the Law of Large Numbers $\frac{1}{n_{old}} \varphi\left(f^{old}(x; \theta)\right)^T \varphi\left(f^{old}(x'; \theta)\right) \to \mathbf{E}\left[\varphi\left(f_1^{old}(x; \theta)\right)^T \varphi\left(f_1^{old}(x; \theta)\right)\right] = \mathbf{E}\left[\varphi(Z)^T \varphi(Z')\right]$ where $\mathbf{Cov}\left(Z, Z'\right) = \begin{bmatrix} \Sigma^{old}(x, x) & \Sigma^{old}(x', x) \\ \Sigma^{old}(x, x') & \Sigma^{old}(x', x') \end{bmatrix}$ by the assumption on the distribution of $f^{old} \sim GP\left(0, \Sigma^{old}(x, x')\right)$. Thus the kernel converges to the desired limit.[12] The independence of $f_i^{out}$and $f_j^{out}$ follows since $f_i^{out}$ depends only on the $i$-th row of $W$, while $f_j^{out}$ depends only on the $j$-th row of $W$, which are independent rows by the construction of $W$. □

[12] Note that more work would be needed to make this convergence more precise; e.g. if the kernel converges is it clear that the whole process is GP in the limit? We will skip this for now.

### 2.3.2 Proof of Neural Tangent Kernel

**Proposition 25.** *[Propagation of the NTK kernel] Suppose $\Theta^{old} : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ is a kernel, and $f^{old} : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{par}} \to \mathbb{R}^{n_{old}}$ is a function. Suppose that for the initialization $\theta_0^{old}$, that $f_i^{old}(x;\theta_0^{old}) \sim GP(0, \Sigma^{old}(x,x'))$ and that components $f_i^{hid}(x;\theta_0^{old})$, $f_j^{hid}(x;\theta_0^{old})$ are independent for $i \neq j$ at initialization $\theta_0$. is a function such that its gradient satisfies*

$$\nabla_\theta f_i^{old}(x;\theta_0^{old})^T \nabla_\theta f_i^{old}(x';\theta_0^{old}) = \Theta^{old}(x,x')$$
$$\nabla_\theta f_i^{old}(x;\theta_0^{old})^T \nabla_\theta f_j^{old}(x';\theta_0^{old}) = 0 \text{ if } i \neq j$$

*Define now a new function $f^{new} : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n'_{par}} \in \mathbb{R}^{new}$ by*

$$f^{new}(x;\theta^{new}) = \frac{\sigma_W}{\sqrt{n_{old}}} \underset{n_{new}\times n_{old}}{W} \varphi\left(\underset{n_{old}\times 1}{f^{old}(x;\theta^{old})}\right) + \sigma_b \underset{n_{new}\times 1}{b},$$

*where $\theta^{new} = \theta^{old} \cup \{W\} \cup \{b\} \in \mathbb{R}^{n'_{par}}$ are the old parameters with the weight matrix $W \in \mathbb{R}^{n_{new}\times n_{old}}$ and bias vector $b \in \mathbb{R}^{n_{new}}$ appended to them (*[13]*. Define the kernels $\Sigma^{new} : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ and $\dot{\Sigma}^{new} : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ using the derivative $\dot{\varphi}(x) = \frac{d}{dx}\varphi(x)$ by the formulas:*

$$\Sigma^{new}(x,x') = \sigma_W^2 \mathbf{E}\left[\varphi(Z)^T \varphi(Z')\right] + \sigma_b^2$$
$$\dot{\Sigma}^{new}(x,x') = \sigma_W^2 \mathbf{E}\left[\dot{\varphi}(Z)^T \dot{\varphi}(Z')\right]$$

*where where $Z, Z'$ are mean 0 Gaussians with covariance structure given by the old kernel $\Sigma^{old}(x,x')$: $\mathbf{Cov}(Z,Z') = \begin{bmatrix} \Sigma^{old}(x,x) & \Sigma^{old}(x',x) \\ \Sigma^{old}(x,x') & \Sigma^{old}(x',x') \end{bmatrix}$. Then in the limit $n_{old} \to \infty$, we have that the gradients of the components $\nabla f_i^{new}$ satisfy*

$$\nabla_{\theta^{new}} f_i^{new}(x;\theta_0^{new})^T \nabla_{\theta^{new}} f_i^{new}(x';\theta_0^{new}) \to \Theta^{new}(x,x')$$
$$\nabla_{\theta^{new}} f_i^{new}(x;\theta_0^{new})^T \nabla_{\theta^{new}} f_j^{new}(x';\theta_0^{new}) \to 0 \text{ for } i \neq j$$

*where $\Theta^{new} : \mathbb{R}^{n_{in}} \times \mathbb{R}^{n_{in}} \to \mathbb{R}$ is defined by*

$$\Theta^{new}(x,x') = \Theta^{old}(x,x')\dot{\Sigma}^{new}(x,x') + \Sigma^{new}(x,x')$$

*Proof.* First consider that the gradient product can be split into a sum of two parts, one part with only the "new" parameters $W, b$, and one part with only the "old" parameters $\theta^{old}$

$$\nabla_{\theta^{new}} f_i^{new}(x;\theta^{new})^T \nabla_{\theta^{new}} f_j^{new}(x';\theta^{new}) = \nabla_{\{W,b\}} f_i^{new}(x;\theta^{new})^T \nabla_{\{W,b\}} f_j^{new}(x';\theta^{new}) + \nabla_{\theta^{old}} f_i^{new}(x;\theta^{new})^T \nabla_{\theta^{old}} f_j^{new}(x';\theta^{new} \tag{17}$$

we will show that the first term goes to $\Theta^{old}(x,x')\dot{\Sigma}^{old}(x,x')$ and the second term goes to $\Sigma^{old}(x,x')$ when $i = j$, and that they both go to zero if $i \neq j$.

The first term of 17 is the same argument as the previous proposition for the the NNGP, since the $W, b$ derivatives are precisely the things from the NNGP kernel argument:

$$\begin{aligned}
\nabla_{\{W,b\}} f_i^{new}(x;\theta^{new})^T \nabla_{\{W,b\}} f_i^{new}(x';\theta^{new}) &= \frac{\sigma_W^2}{n_{old}} \varphi\left(\underset{1\times n_{old}}{f^{old}(x;\theta^{old})}\right)^T \varphi\left(\underset{n_{old}\times 1}{f^{old}(x';\theta^{old})}\right) + \sigma_b^2 \\
&\to \sigma_W^2 \mathbf{E}\left[\varphi\left(f^{old}(x;\theta^{old})\right)^T \varphi\left(f^{old}(x';\theta^{old})\right)\right] + \sigma_b^2 \\
&= \Sigma^{new}(x,x')
\end{aligned}$$

which converges as before. When $i \neq j$, notice that $\nabla_{\{W,b\}} f_i^{new}(x;\theta^{new})^T \nabla_{\{W,b\}} f_j^{new}(x';\theta^{new}) = 0$ since the only terms that are non-zero in the vector $\nabla_{\{W,b\}} f_i^{new}(x;\theta^{new})^T$ are the partial derivatives terms with an $i$: $\frac{\partial}{\partial W_{i,\cdot}}$ and $\frac{\partial}{\partial b_i}$, while the only non-zero terms in the vector $\nabla_{\{W,b\}} f_j^{new}(x';\theta^{new})$ are the partial derivative terms with a $j$: $\frac{\partial}{\partial W_{j,\cdot}}$ and $\frac{\partial}{\partial b_j}$. Hence this inner product is 0.

[13] This means that the number of new parameters is $n'_{par} = n_{par} + (n_{new} + 1)n_{old}$

For the second term of 17, we will compute by chain rule, which is where the $\dot{\varphi}(x)$ will appear! Consider

$$\begin{aligned}\underset{n_{new}\times n_{par}}{\nabla_{\theta^{old}} f^{new}(x;\theta^{new})^T} &= \frac{\sigma_W}{\sqrt{n_{old}}} \underset{n_{new}\times n_{old}}{W} \underset{n_{old}\times n_{par}}{\nabla_{\theta^{old}} \varphi\left(f^{old}(x;\theta^{old})\right)^T} \\ &= \frac{\sigma_W}{\sqrt{n_{old}}} \underset{n_{new}\times n_{old}}{W} \underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\varphi'\left(f^{old}(x;\theta^{hid})\right)\right]} \left(\underset{n_{old}\times n_{par}}{\nabla_{\theta^{old}} f^{old}(x;\theta^{old})^T}\right)\end{aligned}$$

Here $\underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\varphi'\left(f^{old}(x;\theta^{hid})\right)\right]}$ is the $n_{old}\times n_{old}$ matrix which has the elements from the size $n_{old}$ vector $\varphi'\left(f^{old}(x;\theta^{hid})\right)$ on it[14] From this structure, we see the two terms that will give rise to the two terms in the limit we want $\dot{\Sigma}^{new}(x,x')$ and $\Theta^{old}(x,x')$. To see this consider that:

$$\begin{aligned}&\nabla_{\theta^{old}} f_i^{new}(x;\theta^{new})^T \nabla_{\theta^{old}} f_j^{new}(x';\theta^{new}) \\ =&\frac{\sigma_W^2}{n_{old}} \underset{1\times n_{old}}{W_{i\cdot}} \underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\varphi'\left(f^{old}(x;\theta^{hid})\right)\right]} \left(\underset{n_{old}\times n_{par}}{\nabla_{\theta^{old}} f^{old}(x;\theta^{old})^T} \underset{n_{par}\times n_{old}}{\nabla_{\theta^{old}} f^{old}(x';\theta^{old})}\right) \left(\underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\varphi'\left(f^{old}(x';\theta^{hid})\right)\right]}\right) \underset{n_{old}\times 1}{W_{j\cdot}^T} \\ =&\frac{\sigma_W^2}{n_{old}} \underset{1\times n_{old}}{W_{i\cdot}} \underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\varphi'\left(f^{old}(x;\theta^{hid})\right)\right]} \underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\Theta^{old}(x,x')\right]} \underset{n_{old}\times n_{old}}{\operatorname{diag}\left[\varphi'\left(f^{old}(x';\theta^{hid})\right)\right]} \underset{n_{old}\times 1}{W_{j\cdot}^T} \\ =&\frac{\sigma_W^2}{n_{old}} \sum_{k=1}^{n_{old}} W_{ik}\varphi'\left(f_k^{old}(x;\theta^{hid})\right)\Theta^{old}(x,x')\varphi'\left(f_k^{old}(x';\theta^{hid})\right)W_{jk} \\ =&\Theta^{old}(x,x')\frac{\sigma_W^2}{n_{old}} \sum_{k=1}^{n_{old}} W_{ik}\varphi'\left(f_k^{old}(x;\theta^{hid})\right)\varphi'\left(f_k^{old}(x';\theta^{hid})\right)W_{jk}\end{aligned}$$

as before, this is a sum of $n_{old}$ independent variables! In the limit $n_{old}\to\infty$ this converges to its mean by the law of large numbers:

$$\begin{aligned}&\to \Theta^{old}(x,x')\sigma_W^2 \mathbf{E}\left[W_{ik}W_{jk}\varphi'\left(f_k^{old}(x;\theta^{hid})\right)\varphi'\left(f_k^{old}(x';\theta^{hid})\right)\right] \\ &= \begin{cases}\Theta^{old}(x,x')\dot{\Sigma}^{new}(x,x') & \text{if } i=j \\ 0 & \text{if } i\neq j\end{cases}\end{aligned}$$

by the definition of $\dot{\Sigma}^{new}(x,x')$ □

# 3 References

I've cited the arXiv versions since these are more recently updated and include all the appendices with proofs.

[RW06] great reference for the regression model

[MM19, HMRT19] analysis of the random features regression model

[LXS+19, COB20] contains discussion about the linearized model and the comparison $f\approx f^{lin}$

[MRH+18, LBN+17] show the Gaussian process behaiviour and the kernel $\Sigma$

[JGH18] orginal paper with the NTK kernel $\Theta$

# References


[COB20] Lenaic Chizat, Edouard Oyallon, and Francis Bach, *On Lazy Training in Differentiable Programming*, arXiv:1812.07956 [cs, math] (2020), arXiv: 1812.07956.

[HMRT19] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani, *Surprises in High-Dimensional Ridgeless Least Squares Interpolation*, arXiv:1903.08560 [cs, math, stat] (2019) (en).

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler, *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*, arXiv:1806.07572 [cs, math, stat] (2018) (en).

[LBN+17] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein, *Deep Neural Networks as Gaussian Processes*, arXiv:1711.00165 [cs, stat] (2017) (en).


[14] This arises because we are thinking of the function $\varphi:\mathbb{R}\to\mathbb{R}$ as being a function $\varphi:\mathbb{R}^{n_{old}}\to\mathbb{R}^{n_{old}}$ by applying it entrywise. Because of the way of extending the map from $\mathbb{R}$ to $\mathbb{R}^{n_{old}}$, when you take the Jacobian matrix of this map, you get a diagonal matrix.

[LXS+19] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington, *Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent*, arXiv:1902.06720 [cs, stat] (2019), arXiv: 1902.06720.

[MM19] Song Mei and Andrea Montanari, *The generalization error of random features regression: Precise asymptotics and double descent curve*, arXiv:1908.05355 [math, stat] (2019) (en).

[MRH+18] Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani, *Gaussian Process Behaviour in Wide Deep Neural Networks*, arXiv:1804.11271 (2018) (en).

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian processes for machine learning*, Adaptive computation and machine learning, MIT Press, Cambridge, Mass, 2006 (en), OCLC: ocm61285753.