

Implicit Regularization in Deep Learning: The Kernel and Rich Regimes

Henry Smith

Advised by Professor Harrison Zhou

Department of Statistics & Data Science
Yale University

May 5, 2022

Abstract

A stream of recent works in the field of theoretical deep learning has demonstrated the equivalence of neural network training and kernel methods under certain conditions on the model and loss function. One such paper by Chizat and colleagues lays the groundwork for what they coin the “lazy training” or “kernel” limit. In particular, the authors show that for a positive homogeneous, differentiable network that satisfies certain assumptions, taking the scale of the gradient flow initialization to be arbitrarily large produces a kernel estimator. Although this result is certainly fruitful for our understanding of neural network training, we argue along the lines of Woodworth and colleagues that it does not produce good generalization properties in the resulting network function. In particular, for high-dimensional problems in which there exists implicit sparsity in the underlying data, we hypothesize that by training the network far away from the kernel limit, near the “rich limit”, we achieve good generalization of the corresponding model. As a first step in our analysis, we provide some intuition for the kernel and rich limits by illustrating what they look like for the linear regression problem posed by Woodworth and colleagues. Thereafter, we (1) reproduce the results for the sparse linear regression problem studied by Woodworth and colleagues and (2) extend our consideration to the more advanced sparse logistic regression problem considered by Wei and colleagues. Our results suggest that, consistent with our hypothesis, training near the rich limit imposes some form of implicit ℓ^1 regularization, at least for the linear and logistic regression problems.

Acknowledgements

Thank you to my family, especially my parents, for their steadfast support over the past four years. I would not have made it here had you not pushed me to wholeheartedly pursue my interest in statistics. To Professor Zhou, thank you for your guidance and insight over the past five months. Your kind mentorship has meant more to me than I can express in words, and it sets a great example for how I hope to interact with my peers (and perhaps my students) in the future. Lastly, I would like to express my gratitude to Dr. Blake Woodworth, Dr. Tengyu Ma, and Colin Wei for generously sharing their code and talking with me about their research over the past few months.

Contents

1 Overview	4
2 Preliminaries	5
3 The Kernel and Rich Regimes, An Introduction	6
3.1 Defining the Linearized Model	6
3.2 Gradient Flow as a Kernel Method	6
3.3 Relating the Original and Linearized Models	7
3.4 Defining the Kernel and Rich Regimes	9
4 Visualizing the Kernel and Rich Regimes	10
4.1 Linear Regression	10
4.1.1 Explicit Solutions for the Kernel and Rich Limits	11
4.2 The Linearized Model	13
4.3 The Neural Tangent Kernel	14
4.4 The Model Weights	19
5 Rich Training and Sparse Generalization	20
5.1 The Sparse Linear Regression Problem	20
5.1.1 Computational Limitations of Rich Training	24
5.2 A New Problem: Sparse Logistic Regression	24
5.2.1 Logistic Regression	25
5.2.2 Computational Experiments Near the Rich Limit	27
6 Conclusion	31

1 Overview

Without a doubt, neural networks have taken the field of machine learning by storm. Indeed, over the couple of decades, deep learning models have climbed in performance to surpass tree-based methods (random forests, boosting), support vector machines (SVMs), and other predictors once regarded as top-of-the-line. Especially salient are the generalization properties of contemporary neural networks: not only can they fit the data on which they are trained nearly perfectly, but their performance on unseen data is unparalleled by rival predictors. In spite of these contemporary successes of neural networks in the applied realm, their desirable generalization properties remain nebulous from a theoretical perspective. That is to say, practitioners' development and application of deep learning models has drastically outpaced mathematicians' understanding of why these model work so well.

In their paper “On Lazy Training in Differentiable Programming”, Chizat, Oyallon, and Bach shine some light on the theory that dictates network training. In particular, they introduce a phenomenon in network training, the “kernel limit”, in which a differentiable model behaves as a linearization around its initialization [7]. That is to say, the authors show that in the kernel limit, training a model which is highly nonconvex in its parameter space is simplified to training an affine model. More rigorously, one can show that in this kernel limit, subject to certain conditions on the model and loss function, training a neural network via gradient flow on the corresponding objective function is equivalent to a kernel method.

While the kernel limit formalized by Chizat and colleagues is certainly of merit insofar as it imparts a better understanding of neural network training, it also begs the question of whether or not the good generalization properties of contemporary deep learning models can be attributed to training near the kernel limit. More succinctly, we ask whether or not training a neural network near the kernel limit achieves a model with good generalization on unseen data. Woodworth and colleagues broach this question in their paper “Kernel and Rich Regimes in Overparametrized Model”. Specifically, they show that for the case of linear regression, training the model with gradient flow near the kernel limit corresponds to implicit ℓ^2 regularization of the solution vector [13]. Woodworth and colleagues also consider what happens far away from the kernel limit, near the “rich limit”, in which the model behaves very differently from the affine model about its initialization [13]. Contrasting with their result for the kernel limit, they show that training near the rich limit for the linear regression problem corresponds to implicit ℓ^1 regularization of the solution vector.

This result by Woodworth and colleagues, while seemingly limited in scope, points to some significant advances in our knowledge of neural network training. Namely, it implies that, at least in the case of linear regression, training in the rich limit, rather than the kernel limit, leads to good generalization in sparse problems. Considering the analogy of LASSO and ridge regression, we know that ℓ^1 regularization performs much better at enforcing sparsity in the solution than does its ℓ^2 counterpart. Accordingly, we would expect that when the data is drawn from a sparse distribution, the implicit ℓ^1 regularization present near the rich limit would result in a model that generalizes well on unseen data from the same distribution.

In order to study the “lazy” and “rich” limits in neural network training that we have briefly described above, we will first introduce the theory motivating these limits and subsequently consider their implications. In Section 2, we start by introducing some mathematical objects of interest that will be crucial for our discussion of the kernel and rich limits. Thereafter, in Section 3 we detail the formal theory of the kernel and rich limits as it is laid out by Chizat and colleagues [7]. Following our formal characterization of these two limits, in Section 4 we demonstrate empirically that the kernel and rich limits hold for the case of the linear regression model from [13]. And in Section 5 we dive into the most stimulating portion of our paper, considering the generalization properties of networks trained near the kernel and rich limits.

Prior to beginning into our analysis, we point the reader in the direction of our project Github repository, which contains all of the code we wrote to produce our visualizations and conduct our simulations. Furthermore, we advise our reader to open this PDF document in some PDF reader, such as Adobe Acrobat. A number of our visualizations take the form of GIFs and cannot be viewed if opened in a normal web browser.

2 Preliminaries

The notation we present mirrors that of [7] and [13], since these two papers provide the main motivation for our research.

In particular, let $h : \mathbb{R}^p \rightarrow \mathcal{F}$ be a model mapping from parameter space \mathbb{R}^p to Hilbert space \mathcal{F} . In the context of neural networks, we take \mathcal{F} to be the Hilbert space consisting of all possible network functions. Then for each weight vector $\mathbf{w} \in \mathbb{R}^p$, our model h specifies an associated network function $f \in \mathcal{F}$. More explicitly, we have the map $\mathbf{w} \mapsto f(\mathbf{w}, \cdot)$, where $f(\mathbf{w}, \cdot) : \mathcal{X} \rightarrow \mathbb{R}$ is the neural network parameterized by weight vector \mathbf{w} . Here, \mathcal{X} denotes the input space of the network f ; for the purposes of our studies, we consider a Euclidean input space $\mathcal{X} \subseteq \mathbb{R}^n$. Of key importance in this notation is the distinction between h , which maps an individual weight vector to a network function, and $f(\mathbf{w}, \cdot)$, which maps an input point to a response value.

Since this notation is admittedly quite difficult to parse, we provide two illustrative examples which are of interest to our research. First, we consider the case in which $\mathcal{F} = \mathcal{X}^*$, the dual space of \mathcal{X} . Then for each weight vector $\mathbf{w} \in \mathbb{R}^p$, we get the corresponding network function $f(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^n (\beta_{\mathbf{w}})_i \mathbf{x}_i = \langle \beta_{\mathbf{w}}, \mathbf{x} \rangle$ for some $\beta_{\mathbf{w}} \in \mathbb{R}^n$. That is, we associate with each weight vector in \mathbb{R}^p a linear function in the input space \mathcal{X} . Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ be the training points for our neural network, where each $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathbb{R}$. Moreover, let \mathcal{D} be the empirical distribution determined by the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. The keen reader will recognize that minimizing the empirical risk with respect to $\mathbf{w} \in \mathbb{R}^p$, $\arg \min_{\mathbf{w} \in \mathbb{R}^p} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [(y - f(\mathbf{w}, \mathbf{x}))^2] = \arg \min_{\mathbf{w} \in \mathbb{R}^p} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [(y - \langle \beta_{\mathbf{w}}, \mathbf{x} \rangle)^2]$, gives an alternative parameterization of the linear regression problem where $\beta_{\mathbf{w}}$ is the coefficient vector. As for the intercept term, one can suppose that the training data $\{\mathbf{x}_i\}_{i=1}^N$ satisfies $(\mathbf{x}_i)_1 = 1$. This linear regression example is that which is studied in [13].

More generally, though, the network functions $f(\mathbf{w}, \cdot) \in \mathcal{F}$ need not be linear in the input space \mathcal{X} . And so to broaden our function class, we consider $\mathcal{F} = L^2(\mathcal{D}_{\mathbf{x}}, \mathcal{X})$, where $\mathcal{D}_{\mathbf{x}}$ is the \mathbf{x} marginal distribution of \mathcal{D} [7]. That is, we specify that for each vector $\mathbf{w} \in \mathbb{R}^p$, the corresponding network function $f(\mathbf{w}, \cdot)$ must be square integrable with respect to the distribution of input samples. But the distribution \mathcal{D} is in no way special, and so we could take $\mathcal{F} = L^2(\mathcal{P}, \mathcal{X})$ where \mathcal{P} is any probability measure on the input space. Clearly, this is a broad function class \mathcal{F} which encompasses many popular examples in contemporary deep learning.

Now that we have cleared up the meaning of h and $f(\mathbf{w}, \cdot)$, we consider a certain property on h which will be essential for our paper. In particular, we are interested in models h which are *D-positive homogeneous*. This means that for any $\mathbf{w} \in \mathbb{R}^p$ and any $\alpha \in \mathbb{R}_{++}$, then $h(\alpha \mathbf{w}) = \alpha^D h(\mathbf{w})$. Intuitively, h D-positive homogeneous tells us that scaling the output of h by a factor of α is equivalent to scaling the input by a factor of $\alpha^{1/D}$.

To conclude the setup of our paper, we consider how one would find an optimal weight vector \mathbf{w} which minimizes some loss objective on \mathcal{F} . In particular, let $L : \mathcal{F} \rightarrow \mathbb{R}_+$ be the loss function mapping each element f in our Hilbert space \mathcal{F} to a nonnegative real number. An example of such a function L that we mentioned above is the mean-squared error, which is equivalent to the empirical risk corresponding to the square loss $(y - f(\mathbf{x}))^2$:

$$L(f) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [(y - f(\mathbf{x}))^2] = \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2. \quad (1)$$

Given a model $h : \mathbb{R}^p \rightarrow \mathcal{F}$ and corresponding loss function $L : \mathcal{F} \rightarrow \mathbb{R}_+$, we would like to use a gradient-based method to minimize the objective $L(h(\mathbf{w}))$. Surely, though, this would necessitate that we restrict our attention to h and L are everywhere differentiable on their domains. We formally state this assumption:

Assumption (from [7]). *The model $h : \mathbb{R}^p \rightarrow \mathcal{F}$ is differentiable with a locally Lipschitz differential Dh . When we specify that Dh is locally Lipschitz, we are referring to the map $\mathbf{w} \mapsto Dh(\mathbf{w})$, and so the Lipschitz*

constant is defined with respect to the operator norm. Moreover, $L : \mathcal{F} \rightarrow \mathbb{R}_+$ is differentiable with a Lipschitz gradient.

As a consequence, we are able to define the *gradient flow* dynamics on objective function $L(h(\mathbf{w}))$ with respect to $\mathbf{w} \in \mathbb{R}^p$. That is, we suppose that the network weights $\mathbf{w}(t)$ evolve in time $t \in \mathbb{R}_+$ according to the differential equation

$$\mathbf{w}'(t) = -\nabla F(t) = -Dh(\mathbf{w}(t))^T \nabla L(h(\mathbf{w}(t))), \quad \mathbf{w}(0) = \mathbf{w}_0.$$

Here, $F(t) = L(h(\mathbf{w}(t)))$ is the objective function evaluated at $\mathbf{w}(t)$ and $\mathbf{w}_0 \in \mathbb{R}^p$ is some starting point. Also, just as in [7], we use $Dh(\mathbf{w}(t))^T$ to denote the adjoint of $Dh(\mathbf{w}(t)) : \mathbb{R}^p \rightarrow \mathcal{F}$. Practitioners of deep learning should understand the gradient flow as a continuous time analogue to gradient descent [12]. Rather than choosing some positive stepsize with which we update our weights at $t \in \{0, 1, 2, \dots\}$, gradient flow specifies the instantaneous direction in which we should update our weights at every time $t \in \mathbb{R}_+$.

3 The Kernel and Rich Regimes, An Introduction

3.1 Defining the Linearized Model

As we detailed in the previous section, suppose we have some model $h : \mathbb{R}^p \rightarrow \mathcal{F}$ mapping a weight vector to a predictor function. And let $L : \mathcal{F} \rightarrow \mathbb{R}_+$ be a loss function which computes the misfit for each predictor function. Suppose h and L satisfy the assumptions outlined in Section 2. Moreover, suppose that our model h is D -positive homogeneous. Using the gradient flow on the objective function $L(h(\mathbf{w}))$, we would like to find a \mathbf{w}^* such that $h(\mathbf{w}^*)$ minimizes L . Let us denote the gradient flow on $L(h(\mathbf{w}))$ by $(\mathbf{w}(t))_{t \geq 0}$ with starting point $\mathbf{w}(0) = \mathbf{w}_0$.

Corresponding to this model h , we have a *linearized model* \bar{h} defined

$$\bar{h}(\mathbf{w}) := h(\mathbf{w}_0) + Dh(\mathbf{w}_0)(\mathbf{w} - \mathbf{w}_0). \quad (2)$$

In our particular case of $h(\mathbf{w}) = f(\mathbf{w}, \cdot)$ a neural network with $f(\mathbf{w}, \cdot) : \mathcal{X} \rightarrow \mathbb{R}$, then

$$\begin{aligned} \bar{h}(\mathbf{w}) &= \bar{f}(\mathbf{w}, \mathbf{x}) = f(\mathbf{w}_0, \mathbf{x}) + D_{\mathbf{w}}f(\mathbf{w}_0, \mathbf{x})(\mathbf{w} - \mathbf{w}_0) \\ &= f(\mathbf{w}_0, \mathbf{x}) + \langle \nabla_{\mathbf{w}}f(\mathbf{w}_0, \mathbf{x}), \mathbf{w} - \mathbf{w}_0 \rangle, \quad \mathbf{x} \in \mathcal{X}. \end{aligned} \quad (3)$$

One will notice that the linearized model $\bar{h} = \bar{f}(\mathbf{w}, \cdot)$ is simply equal to the linearization of h around its initialization \mathbf{w}_0 [7]. That is, \bar{h} is an affine model in the weights \mathbf{w} with $\bar{h}(\mathbf{w}_0) = h(\mathbf{w}_0) = f(\mathbf{w}_0, \mathbf{x})$ and $D\bar{h}(\mathbf{w})|_{\mathbf{w}=\tilde{\mathbf{w}}} = Dh(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_0} = \nabla_{\mathbf{w}}f(\mathbf{w}_0, \mathbf{x})$ for every $\tilde{\mathbf{w}} \in \mathbb{R}^p$.

From this definition of the linearized model \bar{h} , we state the gradient flow on $L(\bar{h}(\mathbf{w}))$ as

$$\bar{\mathbf{w}}'(t) = -\nabla \bar{F}(t) = -Dh(\mathbf{w}_0)^T \nabla L(\bar{h}(\bar{\mathbf{w}}(t))), \quad \bar{\mathbf{w}}(0) = \mathbf{w}_0$$

where $\bar{F}(t) = L(\bar{h}(\bar{\mathbf{w}}(t)))$ is the linearized objective function evaluated at $\bar{\mathbf{w}}(t)$. Observe that the term $Dh(\bar{\mathbf{w}}(t))^T = Dh(\mathbf{w}_0)^T$ does not depend on the time $t \in \mathbb{R}_+$, which is not the case for the analogous term $Dh(\mathbf{w}(t))^T$ in the gradient flow on $L(h(\mathbf{w}))$. We deliberately choose $\bar{\mathbf{w}}(0)$ such that $\bar{\mathbf{w}}(0) = \mathbf{w}(0) = \mathbf{w}_0$.

3.2 Gradient Flow as a Kernel Method

Now that we have presented the definition of the linearized model, let us analyze the special case of $h(\mathbf{w}) = f(\mathbf{w}, \cdot)$ identically zero at its initialization \mathbf{w}_0 and $L(h(\mathbf{w})) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}}[\ell(y, f(\mathbf{w}, \mathbf{x}))]$, where $\nabla \ell(y, y')$

depends only on $y - y'$ [4]. As in Section 2, \mathcal{D} denotes the empirical distribution determined by the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. The mean-squared error (1) is an example of such an L with $\ell(y, y') = (y - y')^2$.

Under these assumptions, the corresponding linearized model \bar{h} is

$$\bar{h}(\mathbf{w}) = \bar{f}(\mathbf{w}, \mathbf{x}) = \langle \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}), \mathbf{w} - \mathbf{w}_0 \rangle, \quad \mathbf{x} \in \mathcal{X}.$$

And so since $\nabla \ell(y, y')$ only depends on $y - y'$, then the gradient flow on $L(\bar{h}(\mathbf{w}))$ is given by

$$\begin{aligned} \bar{\mathbf{w}}'(t) &= -D_{\mathbf{w}} \bar{f}(\bar{\mathbf{w}}(t), \mathbf{x})^T \nabla L(\bar{h}(\bar{\mathbf{w}}(t))) \\ &= -\nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x})^T \left(\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\frac{\partial}{\partial y'} \ell(y, y') \Big|_{y'=\langle \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}), \bar{\mathbf{w}}(t) - \mathbf{w}_0 \rangle} \right] \right) \\ &= -\nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x})^T \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\ell_{y'}(y - \langle \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}), \bar{\mathbf{w}}(t) - \mathbf{w}_0 \rangle)], \quad \bar{\mathbf{w}}(0) = \mathbf{w}_0. \end{aligned}$$

We write $\ell_{y'}(y - \langle \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}), \bar{\mathbf{w}}(t) - \mathbf{w}_0 \rangle)$ rather than $\ell_{y'}(y, \langle \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}), \bar{\mathbf{w}}(t) - \mathbf{w}_0 \rangle)$ to indicate that $\ell_{y'}$ is a function of $y - y'$. From this expression for $\bar{\mathbf{w}}'(t)$, we observe that the gradient flow of $L(\bar{h}(\mathbf{w}))$ is equivalent to the gradient flow of a linear model with input variables $\nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x})$ and output variable y [4].

Before we can fully grasp the significance of the prior statement, we must first define the *neural tangent kernel* proposed by Jacot and colleagues [9]. In particular, the neural tangent kernel is the positive-definite kernel function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ corresponding to the feature map $\varphi(\mathbf{x}) = \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x})$:

$$\begin{aligned} K(\mathbf{x}_1, \mathbf{x}_2) &:= \langle \varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2) \rangle \\ &= \langle \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}_1), \nabla_{\mathbf{w}} f(\mathbf{w}_0, \mathbf{x}_2) \rangle, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}. \end{aligned} \tag{4}$$

The neural tangent kernel K defines a Reproducing Kernel Hilbert Space (RKHS) H_φ consisting of those $f \in \mathcal{F}$ for which there exists a $\mathbf{w} \in \mathbb{R}^p$ such that $f(\mathbf{x}) = \langle \varphi(\mathbf{x}), \mathbf{w} \rangle, \forall \mathbf{x} \in \mathcal{X}$.

Since the gradient flow of $L(\bar{h}(\mathbf{w}))$ is the same as that of a linear model in the input variables $\varphi(\mathbf{x})$ and the output y , then it is equivalent to a kernel method with the neural tangent kernel (4). For the case of L the mean-squared error (1), if we assume that the gradient flow limit $\bar{\mathbf{w}}^* = \lim_{t \rightarrow \infty} \bar{\mathbf{w}}(t)$ is a global minimizer of the loss L (see Theorem 2.4), then $f(\bar{\mathbf{w}}^*)$ gives a solution to the kernelized linear regression problem

$$\arg \min_{f \in H_\varphi} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [(y - f(\mathbf{x}))^2].$$

More generally, finding a vector $\bar{\mathbf{w}}^* \in \mathbb{R}^p$ which minimizes $L(\bar{h}(\mathbf{w}))$ is equivalent to finding a function f^* in the RKHS H_φ which minimizes L .

3.3 Relating the Original and Linearized Models

So far, we have characterized the linearized model \bar{h} corresponding to model h and have justified that, under appropriate circumstances, gradient flow on the linearized objective $L(\bar{h}(\mathbf{w}))$ is equivalent to a kernel method. However, we have provided no reason to suggest that the original model h should be at all similar to the linearized model \bar{h} : in general, even the most simple neural networks are highly nonlinear in their weights.

Contrary to what one might expect, [7] rigorously argues that under certain conditions, the gradient flow on $L(h(\mathbf{w}))$, $(\mathbf{w}(t))_{t \geq 0}$, and that on $L(\bar{h}(\mathbf{w}))$, $(\bar{\mathbf{w}}(t))_{t \geq 0}$, remain close for all times $t \in \mathbb{R}_+$ with respect to the Euclidean norm $\|\cdot\|_2$. Furthermore, not only does [7] prove that $\mathbf{w}(t)$ and $\bar{\mathbf{w}}(t)$ are close, but also that the corresponding predictors $h(\mathbf{w}(t))$ and $h(\bar{\mathbf{w}}(t))$ are close with respect to $\|\cdot\|_{\mathcal{F}}$.

In order to fully understand this result and its implications, we must first define the scaled model $\alpha h(\mathbf{w})$ and the corresponding linearized model $\alpha \bar{h}(\mathbf{w})$ for $\alpha \in \mathbb{R}_{++}$. Corresponding to the scaled models, we define the

scaled objective functions $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ and $\frac{1}{\alpha^2}L(\alpha h(\bar{\mathbf{w}}))$. One should note that the multiplicative factor of $\frac{1}{\alpha^2}$ is no more than a positive scaling of the loss and does not affect the set of minimizers of either objective. Let $(\mathbf{w}_\alpha(t))_{t \geq 0}$ be the gradient flow on the objective $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ and $(\bar{\mathbf{w}}_\alpha(t))_{t \geq 0}$ be the gradient flow on $\frac{1}{\alpha^2}L(\alpha \bar{h}(\mathbf{w}))$ satisfying $\mathbf{w}_\alpha(0) = \bar{\mathbf{w}}_\alpha(0) = \mathbf{w}_0$. And so all we have done is scale the output of our model h by a factor of α and considered the gradient flow dynamics on the corresponding objective $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$.

Remarkably, Chizat and colleagues prove that, subject to certain conditions on the model h and the loss L , as the scale of the output $\alpha \rightarrow \infty$, then training the model αh is equivalent to training the linearized model $\alpha \bar{h}$. The following two theorems summarize their results as they pertain to our research:

Theorem 2.2 (from [7]). *Assume that $h(\mathbf{w}_0) = 0$. Given a fixed time horizon $T > 0$, it holds that $\sup_{t \in [0, T]} \|\mathbf{w}_\alpha(t) - \mathbf{w}_0\|_2 = \mathcal{O}(1/\alpha)$,*

$$\sup_{t \in [0, T]} \|\mathbf{w}_\alpha(t) - \bar{\mathbf{w}}_\alpha(t)\|_2 = \mathcal{O}(1/\alpha^2) \quad \text{and} \quad \sup_{t \in [0, T]} \|\alpha h(\mathbf{w}_\alpha(t)) - \alpha \bar{h}(\bar{\mathbf{w}}_\alpha(t))\|_{\mathcal{F}} = \mathcal{O}(1/\alpha).$$

Theorem 2.4 (from [7]). *Consider the M -smooth and m -strongly convex loss L with minimizer f^* and condition number $\kappa := M/m$. Assume that σ_{\min} , the smallest singular value of $Dh(\mathbf{w}_0)^T$, is positive and that the initialization satisfies $\|h(\mathbf{w}_0)\|_{\mathcal{F}} \leq C_0 := \sigma_{\min}^3 / (32\kappa^{3/2} \|Dh(\mathbf{w}_0)\| \text{Lip}(Dh))$, where $\text{Lip}(Dh)$ is the Lipschitz constant of Dh . If $\alpha > \|f^*\|_{\mathcal{F}}/C_0$, then for $t \in \mathbb{R}_+$, it holds*

$$\|ah(\mathbf{w}_\alpha(t)) - f^*\|_{\mathcal{F}} \leq \sqrt{\kappa} \|\alpha h(\mathbf{w}_0) - f^*\|_{\mathcal{F}} \exp(-m\sigma_{\min}^2 t/4).$$

If moreover $h(\mathbf{w}_0) = 0$, it holds as $\alpha \rightarrow \infty$, $\sup_{t \geq 0} \|\mathbf{w}_\alpha(t) - \mathbf{w}_0\|_2 = \mathcal{O}(1/\alpha)$,

$$\sup_{t \geq 0} \|\alpha h(\mathbf{w}_\alpha(t)) - \alpha \bar{h}(\bar{\mathbf{w}}_\alpha(t))\|_{\mathcal{F}} = \mathcal{O}(1/\alpha) \quad \text{and} \quad \sup_{t \geq 0} \|\mathbf{w}_\alpha(t) - \bar{\mathbf{w}}_\alpha(t)\|_2 = \mathcal{O}(\log \alpha / \alpha^2).$$

Starting with Theorem 2.2, we have that for model h satisfying $h(\mathbf{w}_0) = 0$ and fixed time $T > 0$, the ℓ^2 distance between $\mathbf{w}_\alpha(t)$, the gradient flow path of $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$, and \mathbf{w}_0 , the initialization of the gradient flow, for $t \in [0, T]$ is no greater than $\mathcal{O}(1/\alpha)$. This result indicates that in the limit as $\alpha \rightarrow \infty$, the gradient flow path of $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ at any time $t \in \mathbb{R}_+$, $\mathbf{w}_\alpha(t)$, remains fixed at the initialization $\mathbf{w}_\alpha(0)$. Moreover, as we had previously hinted at, the distance between the gradient flow path of the scaled original model, $\mathbf{w}_\alpha(t)$, and that of the scaled linearized model, $\bar{\mathbf{w}}_\alpha(t)$, is no greater than $\mathcal{O}(1/\alpha^2)$ for any $t \in [0, T]$. As a result, in the limit $\alpha \rightarrow \infty$ we have that $\mathbf{w}_\alpha(t) = \bar{\mathbf{w}}_\alpha(t)$ for any time $t \in \mathbb{R}_+$. Looking at the networks $ah(\mathbf{w}_\alpha(t)), ah(\bar{\mathbf{w}}_\alpha(t)) \in \mathcal{F}$ themselves, Theorem 2.2 also tells us that the distance between $ah(\mathbf{w}_\alpha(t))$ and $ah(\bar{\mathbf{w}}_\alpha(t))$ in the Hilbert space norm is no greater than $\mathcal{O}(1/\alpha)$ for each $t \in [0, T]$. Thus, as $\alpha \rightarrow \infty$ the scaled model ah evaluated along its gradient flow path at any time $t \in \mathbb{R}_+$, $\mathbf{w}_\alpha(t)$, is equal to the scaled linearized model $\alpha \bar{h}$ evaluated at its gradient flow path at this same time, $\bar{\mathbf{w}}_\alpha(t)$ [7].

As a consequence of the final two results discussed in Theorem 2.2, we have that in the $\alpha \rightarrow \infty$ limit, the limits reached by the gradient flow dynamics of $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ and $\frac{1}{\alpha^2}L(\alpha \bar{h}(\mathbf{w}))$ are equal, $\lim_{t \rightarrow \infty} \mathbf{w}_\alpha(t) = \lim_{t \rightarrow \infty} \bar{\mathbf{w}}_\alpha(t)$, as are the scaled model ah and the scaled linearized model $\alpha \bar{h}$ evaluated at this limit. Decidedly, Theorem 2.2 is a robust theoretical result: we have established that in the $\alpha \rightarrow \infty$ limit, the gradient flow of $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ is equivalent to that of $\frac{1}{\alpha^2}L(\alpha \bar{h}(\mathbf{w}))$, and the corresponding predictor functions are equal.

What Theorem 2.2 leaves out, however, is the dependence of the convergence on the finite time horizon T . As one can see by referencing the original proof of this result, there is an exponential dependence on the time horizon $T > 0$ in the results $\sup_{t \in [0, T]} \|\mathbf{w}_\alpha(t) - \bar{\mathbf{w}}_\alpha(t)\|_2 = \mathcal{O}(1/\alpha^2)$ and $\sup_{t \in [0, T]} \|\alpha h(\mathbf{w}_\alpha(t)) - \alpha \bar{h}(\bar{\mathbf{w}}_\alpha(t))\|_{\mathcal{F}} = \mathcal{O}(1/\alpha)$. That is, although we are guaranteed these convergence results as $\alpha \rightarrow \infty$, the convergence will be very slow for large time $t \in \mathbb{R}_+$. This reality makes this seemingly powerful theorem quite weak in practice, since we often want to observe the gradient flow paths of $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ and $\frac{1}{\alpha^2}L(\alpha \bar{h}(\mathbf{w}))$ for large t when they are close to convergence.

Serving as partial redress for the shortcomings of Theorem 2.2, Theorem 2.4 makes stronger assumptions on the model h and loss L to provide bounds that are uniform in time $t \in \mathbb{R}_+$. In particular, we require that

our loss function is both M -smooth and m -strongly convex. Furthermore, we need it to be true that the derivative of h evaluated at \mathbf{w}_0 is surjective and that Dh is globally Lipschitz. As noted in [7], it can only be the case that $Dh(\mathbf{w}_0)$ is surjective if the Hilbert space \mathcal{F} is finite-dimensional, such as for $\mathcal{F} = \mathcal{X}^*$. Equipped with these assumptions, Theorem 2.4 relaxes each of the three bounds in Theorem 2.2 to be uniform in time $t \in \mathbb{R}_+$. Just as important, Theorem 2.4 also states that for sufficiently large $\alpha \in \mathbb{R}_{++}$, the scaled model $\alpha h(\mathbf{w})$ evaluated along the gradient flow path $(\mathbf{w}_\alpha(t))_{t \geq 0}$ converges exponentially to the global minimizer f^* of the loss L . Note that because we assume the loss f^* is m -strongly convex, then f^* is unique. Surely, this convergence guarantee is a powerful result since the objective $L(h(\mathbf{w}))$ is not a priori convex in $\mathbf{w} \in \mathbb{R}^p$, and so we are solving a nonconvex optimization problem [7].

Recall our assumption that h is D -positive homogeneous, and so $\alpha h(\mathbf{w}) = h(\alpha^{1/D}\mathbf{w})$ as well as $\alpha \bar{h}(\mathbf{w}) = \bar{h}(\alpha^{1/D}\mathbf{w})$. Therefore, the gradient flow on the scaled objective $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$ with initialization $\mathbf{w}_\alpha(0) = \mathbf{w}_0$ is exactly the same as the gradient flow on the unscaled objective $\frac{1}{\alpha^2}L(h(\mathbf{w}))$ with initialization $\mathbf{w}_\alpha(0) = \alpha^{1/D}\mathbf{w}_0$ [13]. Throughout the remainder of the paper, we will consider the second scenario in which the initialization \mathbf{w}_0 is scaled by α as opposed to the model output $h(\mathbf{w}_\alpha(t))$. We prefer the latter interpretation because it involves scaling merely the initialization \mathbf{w}_0 while leaving the model itself untouched.

3.4 Defining the Kernel and Rich Regimes

Altogether, Theorems 2.2 and 2.4 from [7] formulate the *kernel limit* that will be of primary interest in Sections 4 and 5. To conclude this portion of our paper, we make clear our definition of the kernel limit and contrast it with the *rich limit*.

To be explicit, for h a D -homogeneous model that is unbiased at its initialization $h(\mathbf{w}_0) = 0$, the kernel limit occurs when the initialization scale $\alpha \in \mathbb{R}_{++}$ of the gradient flow $(\mathbf{w}_\alpha(t))_{t \geq 0}$, $\mathbf{w}_\alpha(0) = \alpha\mathbf{w}_0$ on objective $\frac{1}{\alpha^2}L(h(\mathbf{w}))$ tends to infinity. From the work of Chizat and colleagues in Theorem 2.2 we know that, under the assumptions outlined in Section 2 on h and L , training the nonlinear model h with gradient flow is equivalent to training the affine model \bar{h} as $\alpha \rightarrow \infty$. This is how the kernel limit gets its name: assuming certain conditions are satisfied, then in the limit $\alpha \rightarrow \infty$ training model h with loss L is equivalent to a kernel method with the neural tangent kernel K (see Section 3.2). As we mentioned in our discussion of Theorem 2.2, the kernel limit is equivalently characterized by the distance of the gradient flow from its initialization $\alpha\mathbf{w}_0$. In particular, as $\alpha \rightarrow \infty$, the network weights become constant throughout training. It is for this reason that Chizat and colleagues refer to the kernel limit as “lazy training” [7].

Note that in practice, we will never observe the true kernel limit $\alpha \rightarrow \infty$. Therefore, we use the term *kernel regime* to refer to the approximation of the kernel limit: when $\mathbf{w}_\alpha(t)$ is close to $\bar{\mathbf{w}}_\alpha(t)$ and $h(\mathbf{w}_\alpha(t))$ is close to $h(\bar{\mathbf{w}}_\alpha(t))$ in their respective norms for all times $t \in \mathbb{R}_+$ [13].

As evidenced by the contemporary field of deep learning, the models that we wish to optimize are often highly nonlinear in \mathbf{w} around their initialization $\alpha\mathbf{w}_0$, and so the gradient flow on $\frac{1}{\alpha^2}L(h(\mathbf{w}))$ does not operate in the kernel regime. That is, the difference between $\mathbf{w}_\alpha(t)$ and $\bar{\mathbf{w}}_\alpha(t)$ as well as the difference between $h(\mathbf{w}_\alpha(t))$ and $h(\bar{\mathbf{w}}_\alpha(t))$ are large for all times $t \in \mathbb{R}_+$. We refer to this phenomenon as the *rich regime* which corresponds to the *rich limit* $\alpha \rightarrow 0$. Analogous to the kernel limit, the rich limit can be equivalently described in terms of the distance between the gradient flow path $\mathbf{w}_\alpha(t)$ and the gradient flow initialization $\alpha\mathbf{w}_0$. In the rich limit, $\mathbf{w}_\alpha(t)$ moves far from its initialization $\mathbf{w}_\alpha(0) = \alpha\mathbf{w}_0$ during training. As a result, Woodworth and colleagues refer to the rich limit as “active training,” matching the title given to the kernel limit by Chizat and colleagues.

The distinction between the kernel and rich regimes is important because gradient flow in each regime leads to distinct *implicit biases* in the resulting model $h(\lim_{t \rightarrow \infty} \mathbf{w}_\alpha(t))$. In other words, the limit of the gradient flow on $\frac{1}{\alpha^2}L(h(\mathbf{w}))$ will result in a network $h(\lim_{t \rightarrow \infty} \mathbf{w}_\alpha(t)) = f(\lim_{t \rightarrow \infty} \mathbf{w}_\alpha(t), \mathbf{x})$, $\mathbf{x} \in \mathcal{X}$ that has different properties depending on whether gradient flow operated in the kernel or rich regime. As we will more thoroughly examine in Sections 4 and 5, the most influential hypothesis made by Woodworth and

colleagues in [13] is that training a network in the rich regime implicitly imposes sparsity on its parameters, which is not the case for the kernel regime. If true, this would allow for better generalization in certain sparse problems (i.e. the underlying model from which the data (\mathbf{x}, y) is generated is sparse) when the network is trained in the rich regime as opposed to the kernel regime.

4 Visualizing the Kernel and Rich Regimes

In Section 3, we rigorously defined the kernel and rich regimes for a D -homogeneous model $h : \mathbb{R}^p \rightarrow \mathcal{F}$ and loss function $L : \mathcal{F} \rightarrow \mathbb{R}_+$ satisfying some differentiability conditions. However, our discussion of the kernel and rich regimes has so far been purely theoretical. In this section, we seek to empirically demonstrate that the results from [7] do, in fact, hold. In order to do so, we focus our attention on the linear regression problem studied by Woodworth and colleagues in [13]. Although the model under consideration is the same, the code, experiments, and visualizations we present are of our own formulation.

4.1 Linear Regression

To begin, we introduce the linear regression model from [13] with which we will demonstrate the kernel and rich regimes.

Consider a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^n$ and $y_i \in \mathbb{R}$ for $1 \leq i \leq N$. Then, corresponding to this training dataset, we specify the linear regression model $h : \mathbb{R}^{2n} \rightarrow (\mathbb{R}^n)^*$ such that

$$h(\mathbf{w}) = f(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^n (\mathbf{w}_{+,i}^2 - \mathbf{w}_{-,i}^2) \mathbf{x}_i \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_+ \\ \mathbf{w}_- \end{bmatrix} \in \mathbb{R}^{2n} \quad \mathbf{x} \in \mathcal{X}. \quad (5)$$

It may appear odd that we have parameterized the linear regression model in this way. Namely, why do we have two sets of weights $\mathbf{w}_{+,i}$ and $\mathbf{w}_{-,i}$ corresponding to each input dimension, and why are the individual weights squared? As is discussed in [13], the two sets of weights ensure that the image of h is all of $(\mathbb{R}^n)^*$, the Hilbert space of linear functionals on \mathbb{R}^n . If this were not the case, then we would have linear regression with the additional constraint that all the coefficients need be nonnegative. The individual components of the weight vector \mathbf{w} are squared so that the model h is two-positive homogeneous: $\sum_{i=1}^n ((\alpha \mathbf{w}_{+,i})^2 - (\alpha \mathbf{w}_{-,i})^2) \mathbf{x}_i = \alpha^2 \sum_{i=1}^n (\mathbf{w}_{+,i}^2 - \mathbf{w}_{-,i}^2) \mathbf{x}_i$ for each $\alpha \in \mathbb{R}_{++}$.

Further, our linear regression model $f(\mathbf{w}, \mathbf{x})$ is linear in the input space \mathcal{X} . Therefore, as is guaranteed by the Riesz representation theorem, we can express $h(\mathbf{w})$ as

$$h(\mathbf{w}) = f(\mathbf{w}, \mathbf{x}) = \langle \boldsymbol{\beta}_{\mathbf{w}}, \mathbf{x} \rangle, \quad \boldsymbol{\beta}_{\mathbf{w}} = \mathbf{w}_+^2 - \mathbf{w}_-^2, \quad \mathbf{x} \in \mathcal{X}, \quad (6)$$

where \mathbf{z}^2 denotes element-wise squaring of the vector $\mathbf{z} \in \mathbb{R}^n$. As we previously pointed out in Section 2, one should recognize $\boldsymbol{\beta}_{\mathbf{w}}$ as the classical linear regression coefficient vector. Also, we mention that whenever \mathbf{w} satisfies $\mathbf{w}_+ = \mathbf{w}_-$, then $h(\mathbf{w}) = \langle \boldsymbol{\beta}_{\mathbf{w}}, \mathbf{x} \rangle = \langle \mathbf{0}, \mathbf{x} \rangle = 0$.

As for our loss function $L : (\mathbb{R}^n)^* \rightarrow \mathbb{R}_+$, we will work with the mean-squared error (1). This differs from the square loss considered in [13] only by a factor of $\frac{1}{N}$, and so the set of minimizers of the two losses is identical.

Having described both the linear regression model h as well as the mean-squared error L , we state an additional assumption that is important for investigating the problem:

Assumption. *Let $N \leq n$, meaning the dimension of the input space is at least as large as the number of training points.*

We make this assumption because, as previously discussed, we are interested in the different solutions to the gradient flow dynamics when trained in the kernel and rich regimes. Taking $N \leq n$ ensures that the system $\mathbf{X}\beta_{\mathbf{w}} = \mathbf{y}$ is underdetermined, and so there are usually (but not always), many solutions for $\beta_{\mathbf{w}}$. Equivalently stated, we choose our network $f(\mathbf{w}, \mathbf{x})$ to be overparameterized so that there are many potential weight vectors \mathbf{w} which minimize the objective $L(h(\mathbf{w}))$.

To address how the linear regression problem posed by Woodworth fits into the theory of Chizat, we assert that h and L satisfy the differentiability assumptions stated in Section 2, and so the lazy training results from Theorem 2.2 indeed hold. That is, we are guaranteed that $\alpha \rightarrow \infty$ is indeed the kernel limit as we have previously characterized it. However, by our assumption that the system $\mathbf{X}\beta_{\mathbf{w}} = \mathbf{y}$ is underdetermined, it cannot be the case that L is strongly convex. This is because there are many minimizers f^* of the loss function L . Consequently, we do not get the uniform time bounds nor the global convergence guarantee from Theorem 2.4.

4.1.1 Explicit Solutions for the Kernel and Rich Limits

Albeit quite straightforward and lacking in complexity, the linear regression problem is well-suited for studying the implicit biases resulting from training in the kernel and rich regimes. Woodworth and colleagues provide a thorough characterization of these implicit biases, which we summarize in the remainder of this section. In particular, let us consider the gradient flow on $L(h(\mathbf{w}))$, $(\mathbf{w}_\alpha(t))_{t \geq 0}$, with initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$. We are explicitly choosing $\mathbf{w}_0 = \mathbb{1}$ so that $h(\mathbf{w}_0) = 0$, as we pointed out above.

First, let us determine the kernel limit. Writing down the feature map $\varphi(\mathbf{x}) = \nabla_{\mathbf{w}} f(\mathbf{w}_\alpha(0), \mathbf{x})$ explicitly,

$$\begin{aligned} \varphi(\mathbf{x}) &= \nabla_{\mathbf{w}} \left(\sum_{i=1}^n (\mathbf{w}_{+,i}^2 - \mathbf{w}_{-,i}^2) \mathbf{x}_i \right) \Big|_{\mathbf{w}=\mathbf{w}_\alpha(0)} \\ &= 2 \begin{bmatrix} (\mathbf{w}_+) \odot \mathbf{x} \\ -(\mathbf{w}_-) \odot \mathbf{x} \end{bmatrix} \Big|_{\mathbf{w}=\mathbf{w}_\alpha(0)} \\ &= 2\alpha \begin{bmatrix} \mathbf{x} \\ -\mathbf{x} \end{bmatrix}, \quad \mathbf{x} \in \mathcal{X}. \end{aligned}$$

Here, $\mathbf{y} \odot \mathbf{z}$ denotes the element-wise product of vectors $\mathbf{y}, \mathbf{z} \in \mathbb{R}^n$. From [13], we know that the $\mathbf{w}^* = \lim_{t \rightarrow \infty} \bar{\mathbf{w}}_\alpha(t)$ reached by gradient flow on the objective $L(\bar{h}(\mathbf{w}))$ satisfies $\bar{h}(\mathbf{w}^*) = \arg \min_{f \in (\mathbb{R}^n)^*} \|f\|_K$, $f(\mathbf{X}) = \mathbf{y}$, where $\|\cdot\|_K$ is the RKHS norm determined by $\varphi(\mathbf{x})$. That is, the predictor function corresponding to the gradient flow solution \mathbf{w}^* is the closest function f to $f(\mathbf{w}_0)$ in the RKHS which is a global minimizer of the loss. Expanding out the definition of the RKHS norm, we get:

$$\begin{aligned} \bar{h}(\mathbf{w}^*) &= \arg \min_{f \in (\mathbb{R}^n)^*} \|f\|_K, \quad f(\mathbf{X}) = \mathbf{y} \\ &= \arg \min_{f \in (\mathbb{R}^n)^*} \left(\inf \left\{ \|\mathbf{w}\|_2 : \mathbf{w} \in \mathbb{R}^{2n}, f(\mathbf{x}) = \langle \mathbf{w}, \varphi(\mathbf{x}) \rangle, \forall \mathbf{x} \in \mathcal{X} \right\} \right), \quad f(\mathbf{X}) = \mathbf{y} \\ &= \arg \min_{f \in (\mathbb{R}^n)^*} \left(\inf \left\{ \|\mathbf{w}\|_2 : \mathbf{w} \in \mathbb{R}^{2n}, f(\mathbf{x}) = 2\alpha \langle \mathbf{w}_+ - \mathbf{w}_-, \mathbf{x} \rangle, \forall \mathbf{x} \in \mathcal{X} \right\} \right), \quad f(\mathbf{X}) = \mathbf{y} \\ &= \arg \min_{f \in (\mathbb{R}^n)^*} \left(\inf \left\{ \|\mathbf{w}\|_2 : \mathbf{w} \in \mathbb{R}^{2n}, f(\mathbf{x}) = \langle \mathbf{w}_+ - \mathbf{w}_-, \mathbf{x} \rangle, \forall \mathbf{x} \in \mathcal{X} \right\} \right), \quad f(\mathbf{X}) = \mathbf{y}. \end{aligned} \tag{7}$$

Since $f \in (\mathbb{R}^n)^* \Leftrightarrow f(\mathbf{x}) = \langle \beta, \mathbf{x} \rangle$, $\forall \mathbf{x} \in \mathcal{X}$ for some $\beta \in \mathbb{R}^n$, then (7) implies that $\bar{h}(\mathbf{w}^*) = f(\mathbf{w}^*, \mathbf{x}) = \langle \beta_{\mathbf{w}}^*, \mathbf{x} \rangle \forall \mathbf{x} \in \mathcal{X}$, where $\beta_{\mathbf{w}}^*$ satisfies

$$\begin{aligned} \beta_{\mathbf{w}}^* &= \arg \min_{\beta \in \mathbb{R}^n} \left(\inf \left\{ \|\mathbf{w}\|_2 : \mathbf{w} \in \mathbb{R}^{2n}, \beta = \mathbf{w}_+ - \mathbf{w}_- \right\} \right), \quad \mathbf{X}\beta = \mathbf{y} \\ &= \arg \min_{\beta \in \mathbb{R}^n} \|\beta\|_2, \quad \mathbf{X}\beta = \mathbf{y}. \end{aligned}$$

To summarize, the solution \mathbf{w}^* to the gradient flow dynamics $(\mathbf{w}_\alpha(t))_{t \geq 0}$, $\mathbf{w}(0) = \alpha \mathbb{1}$ on $L(h(\mathbf{w}))$ in the kernel limit $\alpha \rightarrow \infty$ satisfies $h(\mathbf{w}^*) = \langle \beta^{\ell^2}, \mathbf{x} \rangle$, $\forall \mathbf{x} \in \mathcal{X}$, where $\beta^{\ell^2} \in \mathbb{R}^n$ is the minimum ℓ^2 solution to $\mathbf{X}\beta = \mathbf{y}$ [13]. Note that we refer to $(\mathbf{w}_\alpha(t))_{t \geq 0}$ and $(\bar{\mathbf{w}}_\alpha(t))_{t \geq 0}$ as well as $h(\mathbf{w}^*)$ and $\bar{h}(\mathbf{w}^*)$ interchangeably since we are operating in the kernel limit $\alpha \rightarrow \infty$.

In order to treat the rich limit, we rely on a result from Gunasekar and colleagues. In particular, [8] considers the least-squares objective

$$\min_{\mathbf{Z} \succeq 0} F(\mathbf{Z}) = \|\mathcal{A}(\mathbf{Z}) - \mathbf{y}\|_2^2, \quad (8)$$

where $\mathcal{A} : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^N$ is a linear operator satisfying $\mathcal{A}(\mathbf{Z})_i = \langle \mathbf{A}_i, \mathbf{Z} \rangle$ for $\{\mathbf{A}_i\}_{i=1}^N$ measurement matrices as well as $\{y_i\}_{i=1}^N$ output values. Notice that when each \mathbf{A}_i is a standard basis vector in $\mathbb{R}^{n \times n}$, then (8) is a matrix completion problem. The authors define the gradient flow $(\mathbf{Z}_\alpha(t))_{t \geq 0}$, $\mathbf{Z}_\alpha(0) = \alpha \mathbf{Z}_0$ on the objective $\tilde{L}(\mathcal{A}(\mathbf{Z})) = \|\mathcal{A}(\mathbf{Z}) - \mathbf{y}\|_2^2$, where $\mathbf{Z}_0 \in \mathbb{R}^{n \times n}$, $\mathbf{Z}_0 \succeq 0$ is some starting point.

Although, at first glance, the vectorized linear regression problem at hand seems unrelated to the matrix least-squares problem (8), we remark that it is, in fact, a special case. To see why this is true, suppose that we take each \mathbf{A}_i to be a diagonal matrix $(\mathbf{A}_i)_{j,j} = (\mathbf{x}_i)_j$, $1 \leq j \leq n$ corresponding to the i th training observation in the linear regression data. Moreover, let us choose our initial matrix \mathbf{Z}_0 for gradient flow to be $\mathbb{I}_{n \times n}$. Then for all times $t \in \mathbb{R}_+$, $\mathbf{Z}_\alpha(t)$ will be a diagonal matrix with $\langle \mathbf{A}_i, \mathbf{Z}_\alpha(t) \rangle = \langle \beta_{\mathbf{w}_\alpha(t)}, \mathbf{x}_i \rangle$ for each $1 \leq i \leq N$, where $\beta_{\mathbf{w}_\alpha(t)} \in \mathbb{R}^n$, $(\beta_{\mathbf{w}_\alpha(t)})_j = (\mathbf{Z}_\alpha(t))_{j,j}$ is the vector containing the diagonal entries of $\mathbf{Z}_\alpha(t)$. From here, we see that this particular instance of the matrix completion problem (\mathbf{A}_i diagonal matrices, \mathbf{Z}_0 a diagonal matrix) as stated in [8] is simply an alternate parameterization of linear regression.

Now that we have justified looking at the matrix least squares problem, let us state the following result proven by Gunasekar and colleagues:

Theorem 1 (from [8]). *In the case where the matrices $\{\mathbf{A}_i\}_{i=1}^m$ commute, if $\mathbf{Z}_0^* = \lim_{\alpha \rightarrow 0} \mathbf{Z}_\alpha^*$ exists and is a global optima for (8) with $\mathcal{A}(\mathbf{Z}_0^*) = \mathbf{y}$, then $\mathbf{Z}_0^* \in \arg \min_{\mathbf{Z} \succeq 0} \|\mathbf{Z}\|_*$ s.t. $\mathcal{A}(\mathbf{Z}) = \mathbf{y}$. Here, \mathbf{Z}_α^* is the limit $\lim_{t \rightarrow \infty} \mathbf{Z}_\alpha(t)$ of the gradient flow $(\mathbf{Z}_\alpha(t))_{t \geq 0}$, $\mathbf{Z}_\alpha(0) = \alpha \mathbf{Z}_0$.*

In the case of our vectorized linear regression problem described above, the matrix \mathbf{Z} is diagonal, and so the nuclear norm is equal to the ℓ^1 norm of $\beta \in \mathbb{R}^n$, where $(\beta)_j = \mathbf{Z}_{j,j}$, $1 \leq j \leq n$. This means that

$$\mathbf{Z}_0^* \in \arg \min_{\mathbf{Z} \succeq 0} \|\mathbf{Z}\|_*, \quad \mathcal{A}(\mathbf{Z}) = \mathbf{y} \iff \beta_0^* \in \arg \min_{\beta \in \mathbb{R}_+^n} \|\beta\|_1, \quad \mathbf{X}\beta = \mathbf{y},$$

where $\beta_0^* = \lim_{\alpha \rightarrow 0} \beta_\alpha^*$ for $\beta_\alpha^* = \lim_{t \rightarrow \infty} \beta_{\mathbf{w}_\alpha(t)}$. That is, the solution \mathbf{w}^* to the gradient flow $(\mathbf{w}_\alpha(t))_{t \geq 0}$, $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ on $L(h(\mathbf{w}))$ in the rich limit $\alpha \rightarrow 0$ satisfies $h(\mathbf{w}^*) = \langle \beta^{\ell^1}, \mathbf{x} \rangle$, $\forall \mathbf{x} \in \mathcal{X}$, where β^{ℓ^1} is the minimum ℓ^1 solution to $\mathbf{X}\beta = \mathbf{y}$, $\beta \in \mathbb{R}_+^n$ [13]. If the minimum ℓ^1 solution to $\mathbf{X}\beta = \mathbf{y}$ is contained in \mathbb{R}_+^n , then $\beta^{\ell^1} = \arg \min_{\beta \in \mathbb{R}^n} \|\beta\|_1$, $\mathbf{X}\beta = \mathbf{y}$ (i.e. β^{ℓ^1} is the true ℓ^1 solution to the system $\mathbf{X}\beta = \mathbf{y}$).

And so we have written down explicit formulas both for the kernel and rich limits corresponding to the linear regression problem $h(\mathbf{w}) = f(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^n (\mathbf{w}_{+,i}^2 - \mathbf{w}_{-,i}^2) \mathbf{x}_i = \langle \beta_{\mathbf{w}}, \mathbf{x} \rangle$ with loss L the mean-squared error. The results we have presented from [13] are particularly salient as they provide a clear characterization of the implicit biases that result from training in the kernel and rich regimes. In particular, under appropriate conditions on the system $\mathbf{X}\beta = \mathbf{y}$, the solution to the gradient flow dynamics in the kernel limit is the minimum ℓ^2 solution β^{ℓ^2} , whereas the the solution in the rich limit is the minimum ℓ^1 solution β^{ℓ^1} . Woodworth and colleagues go even a step further and indicate how the gradient flow solution interpolates between the minimum ℓ^1 and ℓ^2 solutions as the initialization scale α interpolates between 0 and ∞ (see Theorem 1 in [13]). The consequences of this result are self-evident: in the linear regression setting, one can choose their initialization scale α carefully in order to achieve either ℓ^1 or ℓ^2 shrinkage.

4.2 The Linearized Model

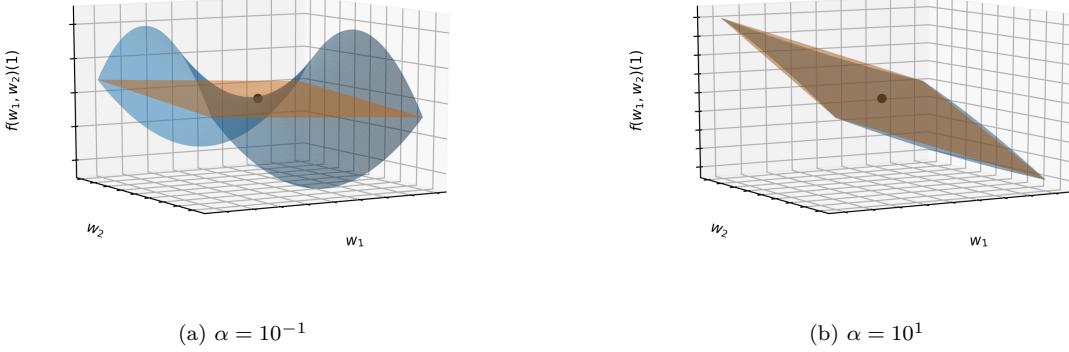


Figure 1: The linear regression model $f(\mathbf{w}, \mathbf{x})$ with one dimensional input $\mathbf{x} = 1$ and weights $\mathbf{w} = (w_1, w_2)$ initialized at $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$. The original linear regression model $f(w_1, w_2)(1)$ is plotted in blue, and the linearized model $\bar{f}(w_1, w_2)(1)$ is in orange.

Now that we have given exact characterizations of the kernel and rich limits for the linear regression problem, we wish to empirically demonstrate the transition between the two limits. Our first demonstration aims to represent the relationship between the original model $f(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^n (\mathbf{w}_{i,+}^2 - \mathbf{w}_{i,-}^2) \mathbf{x}_i$ and the linearized model $\bar{f}(\mathbf{w}, \mathbf{x}) = \langle \varphi(\mathbf{x}), \mathbf{w} - \mathbf{w}_\alpha(0) \rangle = 2\alpha \langle \mathbf{x}, \mathbf{w}_+ - \mathbf{w}_- \rangle$ with initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$. In particular, we know from Section 3.4 that as $\alpha \rightarrow \infty$, the gradient flow $(\mathbf{w}_\alpha(t))_{t \geq 0}$ is identical to $(\bar{\mathbf{w}}_\alpha(t))_{t \geq 0}$ with $\mathbf{w}_\alpha(0) = \bar{\mathbf{w}}_\alpha(0) = \alpha \mathbb{1}$. And so we should observe that for fixed $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{w}, \mathbf{x})$ is approximately equal to $\bar{f}(\mathbf{w}, \mathbf{x})$ as a function of $\mathbf{w} \in \mathbb{R}^p$ around the initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$. Conversely, as $\alpha \rightarrow 0$ we should observe that for fixed \mathbf{x} , $f(\mathbf{w}, \mathbf{x})$ differs substantially from $\bar{f}(\mathbf{w}, \mathbf{x})$ around $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$.

Since \mathbf{w} increases quickly in dimension as n grows, we consider the case of $n = 1$. That is, we have a linear regression model with a single slope coefficient $\beta_{\mathbf{w}} = \mathbf{w}_{+,1}^2 - \mathbf{w}_{-,1}^2$. For simplicity, let us define $w_1 := \mathbf{w}_{+,1}$, $w_2 := \mathbf{w}_{-,1}$, and so we can view f as a neural network with a one-dimensional input x and two weights w_1, w_2 . For our demonstrations, we choose the input $x = 1$ to be fixed. And so we are interested in the functions $f(w_1, w_2)(1), \bar{f}(w_1, w_2)(1) : \mathbb{R}^2 \rightarrow \mathbb{R}$ mapping from the parameter space to the network output.

In Figure 1, we plot each of $f(w_1, w_2)(1)$ and $\bar{f}(w_1, w_2)(1)$ on the square grid $[\alpha - 2, \alpha + 2] \times [\alpha - 2, \alpha + 2]$ for $\alpha = 10^{-1}, 10^1$. Notice that the initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ is at the center of this grid, and the black point represents $f(w_1, w_2)(1)$ evaluated at its initialization, $(\mathbf{w}_\alpha(0), f(\mathbf{w}_\alpha(0))(1)) = (\alpha \mathbb{1}, 0)$. For all $\alpha \in \mathbb{R}_{++}$, f is equal to \bar{f} at its initialization: $f(\mathbf{w}_\alpha(0))(1) = \bar{f}(\mathbf{w}_\alpha(0))(1)$. When $\alpha = 10^1$, we see that $|f(w_1, w_2)(1) - \bar{f}(w_1, w_2)(1)|$ is small for $\|(w_1, w_2) - \alpha \mathbb{1}\|_2 > 0$, meaning f is close to the affine model \bar{f} around $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$. On the other hand, when $\alpha = 10^{-1}$ we observe that $|f(w_1, w_2)(1) - \bar{f}(w_1, w_2)(1)|$ is large for $\|(w_1, w_2) - \alpha \mathbb{1}\|_2 > 0$. That is, f is highly nonlinear in (w_1, w_2) about $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$, and so is far from its linearization around $\mathbf{w}_\alpha(0)$, \bar{f} .

Letting the initialization scale α vary between 10^{-1} and 10^1 in Figure 2, we provide an even clearer representation of the relationship between the original model $f(w_1, w_2)(1)$ and the linearization of f around $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$, $\bar{f}(w_1, w_2)(1)$. Once again, we take the grid of input weights to be the region $[\alpha - 2, \alpha + 2] \times [\alpha - 2, \alpha + 2]$ containing $\mathbf{w}_\alpha(0)$.

Figure 2: The linear regression model $f(w_1, w_2)(1)$ (blue) and the corresponding linearized model $\bar{f}(w_1, w_2)(1)$ (orange) for initialization scale $10^{-1} < \alpha < 10^1$.

From our illustrations, it is evident that for $\alpha \ll 1$, the gradient flow on $L(h(\mathbf{w}))$ with initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ would result in quite a different model than that resulting from the gradient flow on $L(\bar{h}(\mathbf{w}))$ with $\bar{\mathbf{w}}_\alpha(0) = \mathbf{w}_\alpha(0)$. But as α grows away from 0, then h approaches \bar{h} around $\mathbf{w}_\alpha(0)$ and the two solutions achieved by gradient flow $\lim_{t \rightarrow \infty} \mathbf{w}(t), \lim_{t \rightarrow \infty} \bar{\mathbf{w}}(t)$ are nearly identical.

4.3 The Neural Tangent Kernel

The next aspect of the kernel and rich limits that we wish to illustrate is the neural tangent kernel (NTK). Recall that in Section 4.1 we derived the NTK corresponding to the linear regression model at $\mathbf{w}_\alpha(0)$ to be

$$\begin{aligned} K(\mathbf{x}_1, \mathbf{x}_2) &= \langle \nabla_{\mathbf{w}} f(\mathbf{w}, \mathbf{x}_1)|_{\mathbf{w}=\mathbf{w}_\alpha(0)}, \nabla_{\mathbf{w}} f(\mathbf{w}, \mathbf{x}_2)|_{\mathbf{w}=\mathbf{w}_\alpha(0)} \rangle \\ &= \left\langle 2\alpha \begin{bmatrix} \mathbf{x}_1 \\ -\mathbf{x}_1 \end{bmatrix}, 2\alpha \begin{bmatrix} \mathbf{x}_2 \\ -\mathbf{x}_2 \end{bmatrix} \right\rangle \\ &= 8\alpha^2 \langle \mathbf{x}_1, \mathbf{x}_2 \rangle \quad \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}, \end{aligned}$$

which is just proportional to the ℓ^2 kernel [13]. One will also recall our discussion in Section 3.2 that since h is unbiased at its initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ and L is the mean-squared error, then in the limit $\alpha \rightarrow \infty$ the gradient flow on $L(h(\mathbf{w}))$ is equivalent to a kernel method with kernel K . Put differently, in the kernel limit we are looking for a function f^* in the fixed feature space given by $\varphi(\mathbf{x})$ which is a global minimizer of $L(f)$, $f \in (\mathbb{R}^n)^*$ [11].

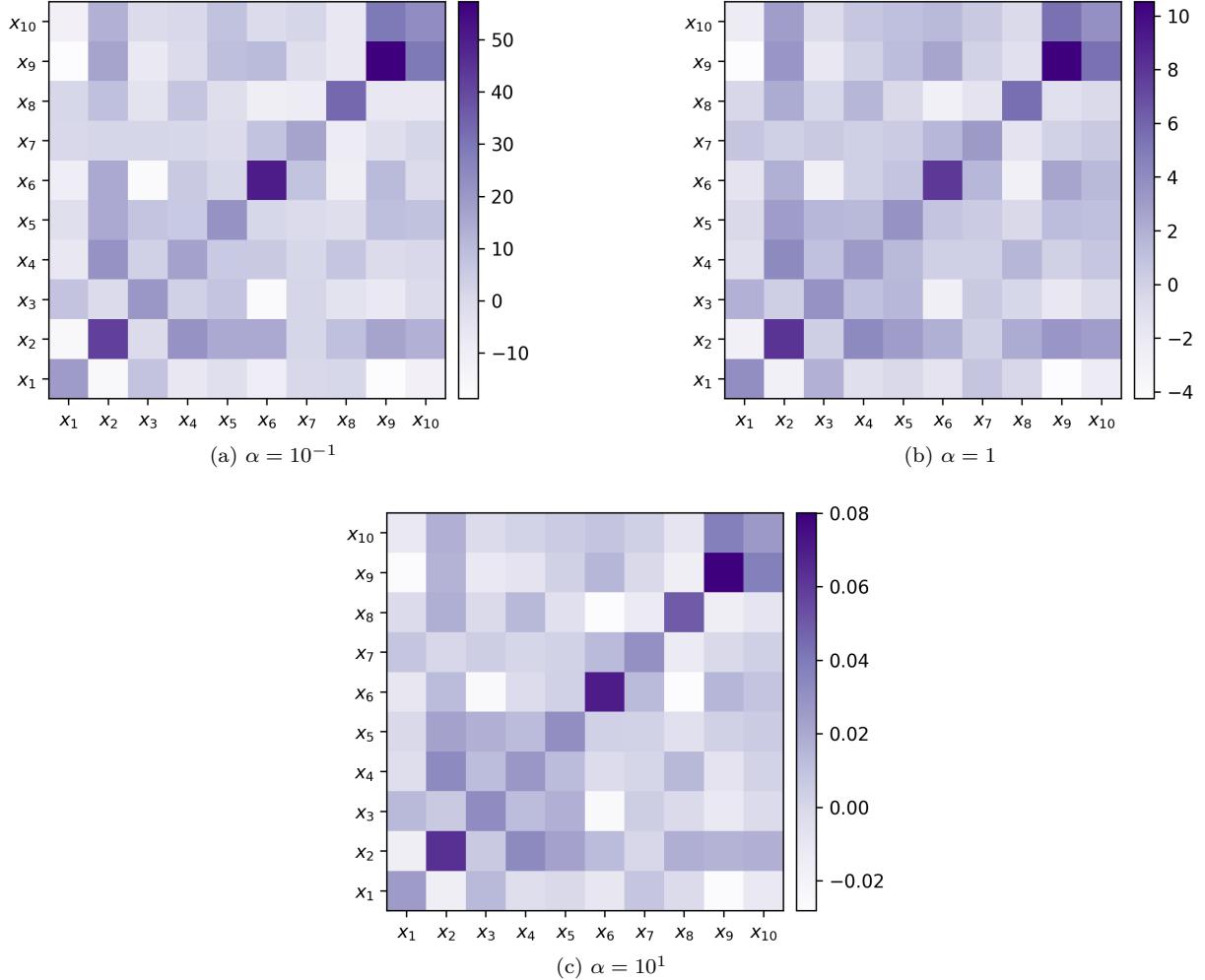


Figure 3: The change in the neural tangent kernel (NTK) on the grid of $N = 10$ training points $\{\mathbf{x}_i\}_{i=1}^{10}$ throughout training with gradient descent. For each of $\alpha = 10^{-1}, 1, 10^1$, we display the difference between the NTK at initialization $\mathbf{w}_\alpha(0)$ and at the end of training $\mathbf{w}_\alpha(t_{\text{end}})$.

Summarizing our theoretical discussion, we know that in the kernel limit, the neural tangent kernel $K(\mathbf{x}_1, \mathbf{x}_2)$, $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ should remain constant for all times $t \in \mathbb{R}_+$ in the gradient flow $(\mathbf{w}_\alpha(t))_{t \geq 0}$, $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ on $L(h(\mathbf{w}))$. Conversely, in the rich limit $\alpha \rightarrow 0$ we should observe that $K(\mathbf{x}_1, \mathbf{x}_2)$ evolves greatly throughout the gradient flow dynamics.

In order to demonstrate that the neural tangent kernel K does, in fact, behave in this way, we consider gradient descent on $L(h(\mathbf{w}))$. We once again start with $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ and let $(\mathbf{w}_\alpha(t))_{t \in \mathbb{N} \cup \{0\}}$ be our gradient descent path. As we previously hinted at in Section 2, gradient descent can be viewed as a forward Euler discretization of the gradient flow dynamics with positive stepsize $\eta > 0$. Accordingly, the gradient descent update is given by $\mathbf{w}_\alpha(t) = \mathbf{w}_\alpha(t-1) - \eta \nabla_{\mathbf{w}}(L(h(\mathbf{w})))|_{\mathbf{w}=\mathbf{w}_\alpha(t-1)}$ for each $t \in \mathbb{N}$.

As for our training data, we consider $N = 10$ points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where each $\mathbf{x}_i \in \mathbb{R}^n$, $n = 20$ and $y_i \in \mathbb{R}$. This indeed satisfies the overparameterization assumption from Section 4.1 since the dimension of the input space is much larger than the number of training points. As was done by Woodworth and colleagues in [13], we draw our input points according to $\mathbf{x}_i \stackrel{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \mathbb{I}_{n \times n})$. And to generate the corresponding set of response points, we compute $y_i = \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle$, where $\boldsymbol{\beta}$ is generated according to the joint uniform distribution

on $[-1, 1]^n$. Clearly, there is at least one vector, namely $\beta_w = \beta$, such that $f = \langle \beta, x \rangle$, $x \in \mathcal{X}$ is a global minimizer of the loss L .

Since our goal is to study the NTK in the kernel and rich regimes, we consider the gradient descent paths corresponding to initialization scales $\alpha = 10^{-1}, 1, 10^1$. To ensure that no one gradient descent path achieves smaller loss than the others, we stop training for each path once $L(h(w_\alpha(t))) < 10^{-4}$. As one could likely predict, the challenge then becomes choosing a stepsize $\eta > 0$ such that $L(h(w_\alpha(t)))$ converges within 10^{-4} of the global minimum. In particular, we would like to choose η to be as small as possible while still achieving the desired convergence within a maximum of 10^4 training epochs. Note that within each epoch, we use batch size 32 to calculate the gradients and perform the reciprocal update steps. A small learning rate η is preferable because in the limit $\eta \rightarrow 0$, the gradient descent with stepsize η reproduces the gradient flow dynamics. In the table below, we summarize our choice of stepsize η for each gradient descent path:

α	η	Number of Epochs to Convergence
10^{-1}	10^{-2}	1001
1	10^{-3}	757
10^1	10^{-4}	77

Table 1

Evidently, no single stepsize $\eta > 0$ works for all initialization scales. One aspect of our experiment that is potentially problematic is that the stepsize corresponding to the gradient descent path $\alpha = 10^{-1}$ is quite large. Therefore, it may be the case that the gradient descent path $(w_{\alpha=10^{-1}}(t))_{t \in \mathbb{N} \cup \{0\}}$ is quite far from the corresponding gradient flow dynamics $(w_{\alpha=10^{-1}}(t))_{t \geq 0}$. With more computational power, it may be possible to achieve convergence with a smaller stepsize by increasing the maximum number of training epochs to be greater than 10^4 .

Before the first training epoch and after the final epoch we evaluate the neural tangent kernel $K_{w_\alpha(t)}$ on the 10×10 grid of training points $\{\mathbf{x}_i\}_{i=1}^N$. We do the same every ten training epochs in order to understand how the neural tangent kernel evolves throughout training. Note that $K_{w_\alpha(t)}$ denotes the neural tangent kernel determined by the weight vector at epoch t of training, $w_\alpha(t)$. That is,

$$K_{w_\alpha(t)}(\mathbf{x}_1, \mathbf{x}_2) = \langle \nabla_{\mathbf{w}} f(\mathbf{w}, \mathbf{x}_1)|_{\mathbf{w}=w_\alpha(t)}, \nabla_{\mathbf{w}} f(\mathbf{w}, \mathbf{x}_2)|_{\mathbf{w}=w_\alpha(t)} \rangle, \quad \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}.$$

In Figure 3, we report the overall change in the NTK

$$K_{w_\alpha(t_{\text{end}})}(\mathbf{x}_i, \mathbf{x}_j) - K_{w_\alpha(0)}(\mathbf{x}_i, \mathbf{x}_j)$$

on $\{\mathbf{x}_i\}_{i=1}^N$ for the gradient descent paths with initialization scales $\alpha = 10^{-1}, 1, 10^1$. Here, t_{end} denotes the final epoch of gradient descent as indicated in Table 1. Noticeably, for the large initialization scale $\alpha = 10^1$, the neural tangent kernel evaluated on the training grid changes very little from the beginning to the end of gradient descent. This empirically verifies our previous remarks about how we would expect the NTK to behave in the kernel regime, since in the kernel limit we have $K_{w_\alpha(t)} = K_{w_\alpha(0)}$ for all times $t \in \mathbb{R}_+$. Conversely, for the small initialization scale $\alpha = 10^{-1}$, we observe that the NTK varies greatly from the beginning to the end of training. That is, $K_{w_\alpha(t_{\text{end}})}$ is very different from $K_{w_\alpha(0)}$. Once again, this is how we would expect the NTK to evolve in the rich regime since the feature space determined by $\varphi_t(\mathbf{x}) = \nabla_{\mathbf{w}} f(\mathbf{w}, \mathbf{x})|_{\mathbf{w}=w_\alpha(t)}$ is not fixed in time $t \in \mathbb{R}_+$ in the rich limit as it is in the kernel limit. From our experiment, we also see the interpolation between the kernel and rich regimes. At $\alpha = 1$, we remark that that the NTK exhibits behavior between the kernel and rich regimes. As we had previously commented, how the solutions reached by the gradient flow dynamics $(w_\alpha(t))_{t \geq 0}$ $w(0) = \alpha \mathbf{1}$ vary between the rich $\alpha \rightarrow 0$ and kernel $\alpha \rightarrow \infty$ limits is considered rigorously by Woodworth and colleagues [13].

Although Figure 3 certainly does portray how the neural tangent kernel changes from the beginning to the end of training, it does not fully capture how it evolves throughout training. To better understand the

Figure 4: The neural tangent kernel (NTK) evaluated on the grid of training points $\{\mathbf{x}_i\}_{i=1}^N$ for every 10 epochs of gradient descent with initialization $\mathbf{w}_{\alpha=10^{-1}}(0) = 10^{-1} \mathbb{1}$.

Figure 5: The neural tangent kernel (NTK) evaluated on the grid of training points $\{\mathbf{x}_i\}_{i=1}^N$ for every 10 epochs of gradient descent with initialization $\mathbf{w}_{\alpha=1}(0) = \mathbb{1}$.

continuous time change of the NTK, we plot $K_{\mathbf{w}_\alpha(t)}$ on $\{\mathbf{x}_i\}_{i=1}^N$ evaluated at every 10 epochs of gradient descent for the paths corresponding to $\alpha = 10^{-1}, 1$. We fix the scale of each plot Figure 4 and Figure 5 at $t = 0$ so that it does not automatically adjust at each evaluation of the NTK. Just as we would expect, the NTK corresponding to $\alpha = 10^{-1}$ experiences large relative changes, even at the end of training when $f(\mathbf{w}_\alpha(t), \mathbf{x})$ is close to a global minimum of the loss L . This is not the case for the gradient descent path corresponding to initialization scale $\alpha = 1$; in fact, it may appear as if the NTK does not change whatsoever. From Figure 3 we know that this is not the case. Rather, for $\alpha = 1$ the changes in the NTK $K_{\mathbf{w}_\alpha(t)}$ are small relative to its scale at $\mathbf{w}_\alpha(0)$.

4.4 The Model Weights

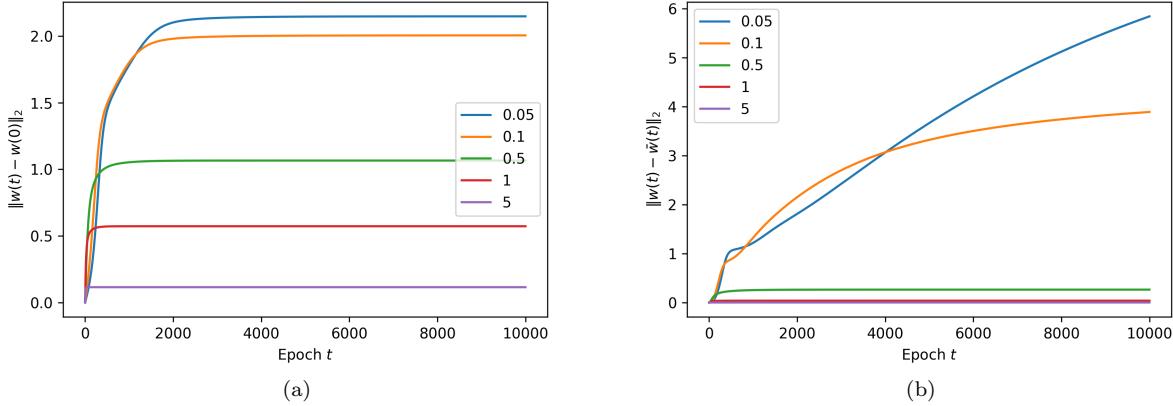


Figure 6: (a) The ℓ^2 distance between the weights of the model at epoch t of gradient descent, $\mathbf{w}_\alpha(t)$, and at initialization $\mathbf{w}_\alpha(0) = \alpha \mathbf{1}$. (b) The ℓ^2 distance between the weights of the model h , $\mathbf{w}_\alpha(t)$, and those of the linearized model \bar{h} , $\bar{\mathbf{w}}_\alpha(t)$, at epoch t of gradient descent. The initialization scale of each gradient descent path is indicated by the line color as reported in the plot legends.

For our final demonstration of the kernel and rich limits in the linear regression problem, we consider how the weights of the model themselves $\mathbf{w}_\alpha(t)$ evolve throughout the gradient flow dynamics on $L(h(\mathbf{w}))$. From the results of Theorem 2.2, we know that in the kernel limit $\alpha \rightarrow \infty$ it holds that $\|\mathbf{w}_\alpha(t) - \mathbf{w}_\alpha(0)\|_2 \rightarrow 0$ as well as $\|\mathbf{w}_\alpha(t) - \bar{\mathbf{w}}_\alpha(t)\|_2 \rightarrow 0$ for each $t \in \mathbb{R}_+$. The first result tells us that for all times $t \in \mathbb{R}_+$, the gradient flow path is asymptotically fixed at its initialization in the kernel limit. The second statement tells us that in the kernel limit, gradient flow on the original objective $L(h(\mathbf{w}))$ is equivalent to the gradient flow on $L(\bar{h}(\mathbf{w}))$. Using a similar setup to that in Section 4.3, we exhibit empirically that each of these two limits do, in fact, hold.

Just as in Section 4.3, we consider a training dataset of $N = 10$ points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with each $\mathbf{x}_i \in \mathbb{R}^n$, $n = 20$ and $y_i \in \mathbb{R}$. The specifics of how we generate these (\mathbf{x}_i, y_i) are exactly the same as in Section 4.3, and so we omit the details here. Similar to our experiment for the neural tangent kernel, we once again consider the gradient descent with initialization $\mathbf{w}_\alpha(0) = \alpha \mathbf{1}$, denoted $(\mathbf{w}_\alpha(t))_{t \in \mathbb{N} \cup \{0\}}$, as a discretization of the gradient flow dynamics. Here, we specifically look at the gradient descent paths corresponding to initialization scales $\alpha \in \{5 \times 10^{-2}, 10^{-1}, 5 \times 10^{-1}, 1, 5\}$. Unlike in the previous experiment, though, we fix a stepsize $\eta = 10^{-3}$ for each α and do not stop training early once $L(h(\mathbf{w}_\alpha(t))) < 10^{-4}$. Rather, we run each gradient descent path for a total of 10^4 epochs. The reason for this is that we would like to compare $\mathbf{w}_\alpha(t)$ for these various α , which we cannot do if the stepsize varies along with α . And once again, we use batch size 32 to compute the gradient and perform the corresponding updates within each epoch.

In order to demonstrate the first limit, $\|\mathbf{w}_\alpha(t) - \mathbf{w}_\alpha(0)\|_2 \rightarrow 0$ as $\alpha \rightarrow \infty$, we store the network weights $\mathbf{w}_\alpha(t)$ both at the beginning of training $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ as well as after every 10 epochs of gradient descent. In Figure 6a, we display the ℓ^2 distance of $\mathbf{w}_\alpha(t)$ from the initialization $\mathbf{w}_\alpha(0)$ as a function of $t \in \mathbb{R}_+$. Just as we would expect under the aforementioned theory, the gradient descent path remains close to $\mathbf{w}(0)$ whenever α is large. However, we do observe that the convergence of $\mathbf{w}_\alpha(t)$ to \mathbf{w}_0 is the most rapid for $t \in \mathbb{R}_+$ small. For the very large initialization scale $\alpha = 5$, we observe that $\mathbf{w}_\alpha(t)$ barely deviates from its initialization. On the opposite end of the paradigm, for $\alpha = 10^{-1}, 5 \times 10^{-2}$ we observe large changes in $\mathbf{w}_\alpha(t)$ throughout training. As we have previously elucidated, this active training is synonymous with the rich limit $\alpha \rightarrow 0$.

Furthermore, we are interested in the relationship between the gradient flow on $L(h(\mathbf{w}))$ and that on $L(\bar{h}(\mathbf{w}))$ in the kernel and rich limits. Since we have already computed $(\mathbf{w}_\alpha(t))_{t \in \mathbb{N} \cup \{0\}}$, then it remains to calculate $(\bar{\mathbf{w}}_\alpha(t))_{t \in \mathbb{N} \cup \{0\}}$, the gradient descent path on the linearized objective function $L(\bar{h}(\mathbf{w}))$ with initialization $\bar{\mathbf{w}}(0) = \alpha \mathbb{1}$. Just as we did for the original objective, we run gradient descent for 10^4 total training epochs with batch size 32 and stepsize $\eta = 10^{-3}$. Likewise, we store the weights $\bar{\mathbf{w}}_\alpha(t)$ at the beginning of training and following every 10 gradient descent epochs. In Figure 6b, we report the ℓ^2 distance between the gradient descent paths $\mathbf{w}_\alpha(t)$ and $\bar{\mathbf{w}}_\alpha(t)$ as a function of t . For large initialization scales $\alpha = 1, 5$, it is evident that $\mathbf{w}_\alpha(t)$ and $\bar{\mathbf{w}}_\alpha(t)$ are close, especially for small times $t \in \mathbb{R}_+$. This supports the assertion that the gradient flow on $L(h(\mathbf{w}))$ is equivalent to that on the linearized objective $L(\bar{h}(\mathbf{w}))$ in the kernel limit $\alpha \rightarrow \infty$. On the contrary, for $\alpha = 5 \times 10^{-2}, 10^{-1}$ small, we see that $\|\mathbf{w}_\alpha(t) - \bar{\mathbf{w}}_\alpha(t)\|_2$ increases substantially throughout training. This suggests that gradient flow on the original objective $L(h(\mathbf{w}))$ is very different from that on the linearized objective $L(\bar{h}(\mathbf{w}))$ in the rich limit $\alpha \rightarrow 0$.

5 Rich Training and Sparse Generalization

Thus far, we have provided theoretical characterizations of the kernel and rich limits for neural network training and have demonstrated that these limits do, in fact, hold for the linear regression model considered in [13]. What has been notably absent from our discussion, though, is why the distinction between kernel and rich training is meaningful. That is, why should one be conscious about whether they are training their network near the kernel limit or near the rich limit?

To address this question, we must study the implicit biases of networks trained in the kernel and rich limits. One will recall that in Section 4.1 we derived the explicit kernel and rich limits for the linear regression model from [13]. In particular, the gradient flow solution in kernel limit is the minimum ℓ^2 norm solution whereas the solution in the rich limit is the minimum ℓ^1 norm solution. As Woodworth and colleagues point out, this result suggests benefits to training in the rich regime when one suspects that there is sparsity in the underlying distribution of data. This association of the rich limit with implicit ℓ^1 regularization does not appear to be limited to the linear regression model, though. In Section 5.2, we provide experimental results which suggest that for a sparse logistic regression problem, the rich limit similarly imposes ℓ^1 regularization of the solution.

5.1 The Sparse Linear Regression Problem

To begin our examination of the implicit biases corresponding to the kernel and rich limits, we will look at the linear regression problem studied in Section 5. In particular, we know that for the model

$$h(\mathbf{w}) = f(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^n (\mathbf{w}_{+,i}^2 - \mathbf{w}_{-,i}^2) \mathbf{x}_i = \langle \boldsymbol{\beta}_{\mathbf{w}}, \mathbf{x} \rangle \quad \boldsymbol{\beta}_{\mathbf{w}} = \mathbf{w}_+^2 - \mathbf{w}_-^2$$

with loss function L the mean-squared error, then gradient flow on the objective $L(h(\mathbf{w}))$, denoted $(\mathbf{w}_\alpha(t))_{t \geq 0}$, with initialization $\mathbf{w}_\alpha(0) = \alpha \mathbb{1}$ satisfies

$$\lim_{t \rightarrow \infty} \beta_{\mathbf{w}_\alpha(t)} = \arg \min_{\beta \in \mathbb{R}^n} \|\beta\|_2 \quad \text{as } \alpha \rightarrow \infty, \quad \lim_{t \rightarrow \infty} \beta_{\mathbf{w}_\alpha(t)} = \arg \min_{\beta \in \mathbb{R}_+^n} \|\beta\|_1 \quad \text{as } \alpha \rightarrow 0.$$

This result suggests that training in the rich regime will outperform training in the kernel regime when there is sparsity in the underlying distribution ρ from which the data (\mathbf{x}_i, y_i) is drawn. When we say “outperform”, note that we are not referring to how well the model fits the training data. The gradient flow solutions reached in both the kernel and rich limits satisfy $\mathbf{X}\beta^* = \mathbf{y}$ for $\beta^* = \lim_{t \rightarrow \infty} \beta_{\mathbf{w}_\alpha(t)}$, meaning they fit the training data exactly. Instead, we are referring to the generalization of the model on unseen data. If $(\mathbf{x}, y) \sim \rho$, then we can quantify how well a model $f(\mathbf{w}) \in \mathcal{F}$ generalizes as

$$\mathbb{E}_{(\mathbf{x}, y) \sim \rho}[(y - f(\mathbf{w}, \mathbf{x}))^2], \quad (9)$$

which is the population risk corresponding to the square loss. That is, by computing the gradient flow limit of $L(h(\mathbf{w}))$ with L the mean-squared error we are finding a minimizer of the empirical risk, which may or may not correspond to a small value of the population risk. It is also important to note that when we say that the distribution ρ is “sparse”, we mean that $\mathbf{w} \mapsto \mathbb{E}_{(\mathbf{x}, y) \sim \rho}[(y - f(\mathbf{w}, \mathbf{x}))^2]$ is minimized by $\mathbf{w} \in \mathbb{R}^p$ sparse. Equivalently, we are stating that the true network parameters, those which minimize the population risk, are sparse.

In order to demonstrate this desirable property of training near the rich limit for the linear regression model, we consider the following problem as it is formulated by Woodworth and colleagues. Suppose we have a training dataset of N points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$ for $n = 250$. We maintain our previous assumption that $N \leq n$ and more specifically that there exists many solutions to the system $\mathbf{X}\beta_\mathbf{w} = \mathbf{y}$. The input data \mathbf{x}_i is distributed according to $\mathcal{N}(\mathbf{0}, \mathbb{I}_{n \times n})$ and the corresponding output data is distributed according to $\mathcal{N}(\langle \beta_\mathbf{w}, \mathbf{x}_i \rangle, \sigma^2)$ where $(\beta_\mathbf{w})_i = 1/\sqrt{5}$ for $1 \leq i \leq 5$, $(\beta_\mathbf{w})_i = 0$ otherwise. Here, $\sigma^2 = 10^{-2}$ is a constant which determines the amount of noise in the output data.

First, one should observe that the data we consider is very high-dimensional. Consequently, with no prior knowledge of the true data distribution ρ , one would expect that they would need many training samples to achieve good generalization of $h(\lim_{t \rightarrow \infty} \mathbf{w}_\alpha(t))$. In spite of the high dimensionality of the input space, though, the underlying data distribution ρ is specified such that y_i is determined by merely the first five coordinates of \mathbf{x}_i . Put more rigorously, by choosing $\mathbf{w} \in \mathbb{R}^{500}$ such that $(\mathbf{w}_+^2 - \mathbf{w}_-^2)_{1:5} = 1/\sqrt{5}$ and $\mathbf{w}_i = 0$ otherwise, then the corresponding network function $f(\mathbf{w})$ minimizes the population risk (9). And so we can theoretically minimize the population risk using a network with only 10 nonzero weights out of $2n = 500$ total and with weight vector satisfying $\|\mathbf{w}\|_1 = \sqrt{5}$.

Now that we have detailed the sparse regression problem from [13], we proceed to address how we will simulate the gradient flow on $L(h(\mathbf{w}))$. Just as in Section 4, we approximate the gradient flow dynamics on $L(h(\mathbf{w}))$, $(\mathbf{w}_\alpha(t))_{t \geq 0}$, using the corresponding gradient descent path, $(\mathbf{w}_\alpha(t))_{t \in \mathbb{N} \cup \{0\}}$, with small stepsize $\eta = 3 \times 10^{-4}$. To ensure that we are fairly comparing the solution vectors $\mathbf{w}_\alpha(t_{\text{end}})$, we stop training once $L(h(\mathbf{w}_\alpha(t))) < 10^{-4}$. Since it might be the case, as we encountered in Section 4.3, that gradient descent converges very slowly to a global minimum of the objective function, we stop training after a maximum of 10^4 epochs. Just as before, we use batch size 32 to compute the gradients and update the parameters.

Unlike in our previous experiments, though, we consider the gradient descent path both as a function of the number of training samples, N , as well as the initialization scale, α . To be more specific, we evaluate the gradient descent path using a training dataset of $N = 50, 100$, and 200 points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. And for each fixed N , we choose 20 logarithmically-spaced values for the initialization scale α on the interval $[10^{-3}, 1]$. Altogether, each combination of N and α specifies a gradient descent path which we simulate as previously described.

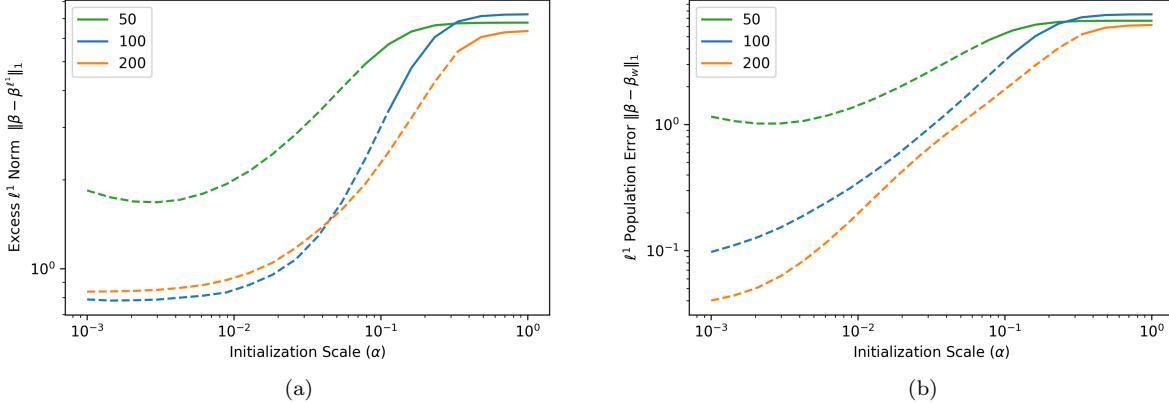


Figure 7: (a) The ℓ^1 difference between the gradient descent solution $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ and the minimum ℓ^1 solution $\arg \min_{\beta \in \mathbb{R}^n} \|\beta\|_1$, $\mathbf{X}\beta = \mathbf{y}$, where $\mathbf{X} \in \mathbb{R}^{N \times n}$ is the matrix of training inputs $\{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{y} \in \mathbb{R}^N$ is the vector of training outputs $\{y_i\}_{i=1}^N$. (b) The ℓ^1 difference between the gradient descent solution and the ground truth vector $\beta_{\mathbf{w}}$ from which the output data is generated. For each of (a) and (b), a dashed line indicates that the gradient descent path did not achieve the specified convergence.

In Figure 7a, we report the excess ℓ^1 norm of the solution vector $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ reached by the gradient descent path $(\mathbf{w}_\alpha(t))_{t \in \mathbb{N} \cup \{0\}}$, $\mathbf{w}_\alpha(0) = \alpha \mathbf{1}$. That is, we determine the minimum ℓ^1 solution to the system $\mathbf{X}\beta = \mathbf{y}$, denoted β^{ℓ^1} , and then calculate the ℓ^1 difference between this vector and the gradient flow solution $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$. From our plot, we see that for each of $N = 100, 200$, there is a clear pattern in which the gradient flow solution is far from the minimum ℓ^1 solution for $\alpha = 1$ (excess ℓ^1 norm ≈ 5) but gets significantly closer to this solution as α shrinks towards zero (excess ℓ^1 norm $\approx 8 \times 10^{-1}$ for $\alpha = 10^{-3}$). From our theoretical results regarding the gradient flow solution for the linear regression model in the rich limit $\alpha \rightarrow 0$, this is what we would expect. More explicitly, we know that in the limit $\alpha \rightarrow 0$, the gradient flow solution on the objective $L(h(\mathbf{w}))$ with initialization $\mathbf{w}_\alpha(0) = \alpha \mathbf{1}$ should correspond exactly to the minimum ℓ^1 solution of the system $\mathbf{X}\beta = \mathbf{y}$, $\beta \in \mathbb{R}_+^n$. On the other hand, since the gradient flow solution in the kernel limit $\alpha \rightarrow \infty$ corresponds to the minimum ℓ^2 solution, then $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ should not be close to β^{ℓ^1} for α large.

While the excess ℓ^1 norm suggests how far the gradient flow solution is from the minimum ℓ^1 norm solution, we would also like to consider how far it is from the vector $\beta_{\mathbf{w}}$ which minimizes the population risk. In Figure 7b we present the ℓ^1 population error, which is measured as the ℓ^1 distance between $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ and $\beta_{\mathbf{w}}$, where $\beta_{\mathbf{w}}$ is the vector defined above which determines the distribution of y_i (i.e. $(\beta_{\mathbf{w}})_i = 1/\sqrt{5}$ for $1 \leq i \leq 5$, $(\beta_{\mathbf{w}})_i = 0$ otherwise). One will notice that unlike β^{ℓ^1} , $\beta_{\mathbf{w}}$ is not a solution to the system $\mathbf{X}\beta = \mathbf{y}$ almost surely. This is a result of the small perturbations made to the output points y_i from their mean $\langle \beta_{\mathbf{w}}, \mathbf{x}_i \rangle$. Nonetheless, since $\beta_{\mathbf{w}}$ is a minimizer of the population risk (9), then it is desirable that $\|\beta_{\mathbf{w}_\alpha(t_{\text{end}})} - \beta_{\mathbf{w}}\|_1$ is small. Indeed, we observe that for $N = 100, 200$, the ℓ^1 population error is small for α small: when $\alpha = 10^{-3}$, the ℓ^1 population error is $\approx 10^{-1}$ with $N = 100$ training samples and $\approx 4 \times 10^{-2}$ with $N = 200$ training samples. Intuitively, we would suspect that $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ should be close to $\beta_{\mathbf{w}}$ for α small since $\beta_{\mathbf{w}}$ itself is sparse. This finding informs how one should choose their initialization scale α to minimize the population risk when they believe that the underlying data distribution ρ is sparse.

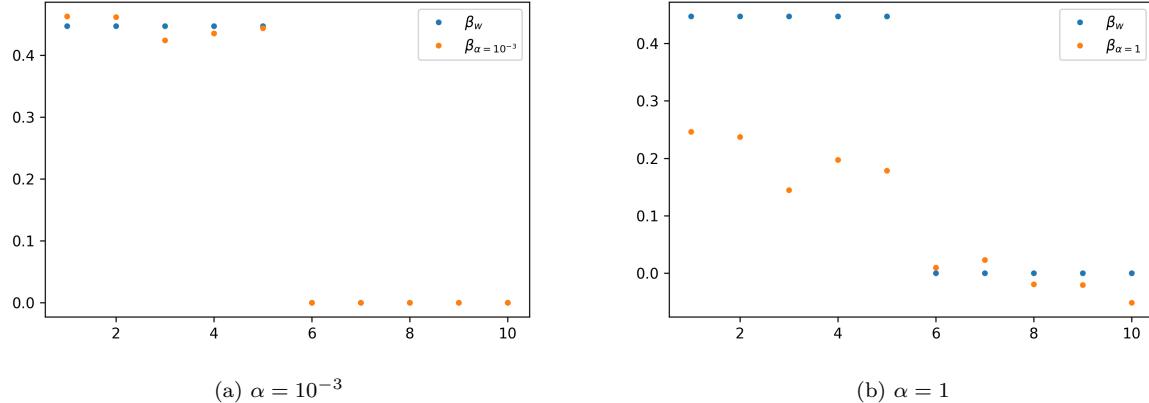


Figure 8: The first 10 coordinates of the gradient descent solution vector $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ (orange) and those of the ground truth vector $\beta_{\mathbf{w}}$ (blue). Gradient descent is performed using $N = 100$ training points.

To better visualize how the gradient flow solution changes as a function of the initialization scale, we plot the first 10 coordinates of $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ against those of $\beta_{\mathbf{w}}$ for each of $\alpha = 10^{-3}$ and $\alpha = 1$. What first sticks out to us is that the gradient descent solution corresponding to $\alpha = 1$ is not sparse. Although the solution vector certainly exhibits shrinkage of the second five coordinates, they are not close to zero as they are in $\beta_{\mathbf{w}}$. For the $\alpha = 10^{-3}$ solution vector, on the other hand, we do observe that the second five coordinates are all nearly zero. Also, the gradient descent solution picks out the first five coordinates as those which are most important for predicting y_i , as is the case in the ground truth vector $\beta_{\mathbf{w}}$. Not only does it identify these coordinates, but $(\beta_{\mathbf{w}_\alpha(t_{\text{end}})})_i$ is very close to the $(\beta_{\mathbf{w}})_i = 1/\sqrt{5}$ for $1 \leq i \leq 5$.

It is also worth mentioning that these experimental results agree with those originally reported by Woodworth and colleagues in [13]. We carefully chose our input dimension n , data distribution ρ , and learning rate η to match with those used by the authors in Figures 1(c) and 3(c). The problem considered in Figures 1(a) and 1(b) uses $N = 100$ training samples, just as we did, but has higher input dimension, $d = 10^3$, than that which we considered. Irrespective of this difference in dimension, we observe the same relationship between the initialization scale $\alpha \in \mathbb{R}_{++}$ of the gradient descent path $(\mathbf{w}_\alpha(t))_{t \geq 0}$ and the population error. Expressly, the authors illustrate that beginning at $\alpha \approx 10^{-1}$ there is a sharp decrease in the population error $\|\beta_{\mathbf{w}_\alpha(t_{\text{end}})} - \beta_{\mathbf{w}}\|_2^2$. And as the initialization scale $\alpha \rightarrow 0$, the population error appears to decrease monotonically towards 0, albeit at a slower rate. One will notice that the definition of ‘‘population error’’ adopted by Woodworth and colleagues, the ℓ^2 difference between $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$ and $\beta_{\mathbf{w}}$, differs from that we report in Figure 7b.

Similarly, in Figure 1(b) of [13], the authors plot the excess ℓ^1 norm of the gradient descent solution vector $\beta_{\mathbf{w}_\alpha(t_{\text{end}})}$. Just as we see from our own experiments in Figure 7a, starting once again at around $\alpha \approx 10^{-1}$, there is a sharp decrease in the excess ℓ^1 norm. It then appears to continue decreasing monotonically as $\alpha \rightarrow 0$. As we have mentioned before, because we know that $\beta_\alpha^* \rightarrow \beta^{\ell^1}$ as $\alpha \rightarrow \infty$, where $\beta_\alpha^* = \lim_{t \rightarrow \infty} \beta_{\mathbf{w}_\alpha(t)}$, then we would expect the excess ℓ^1 norm of the gradient descent solution to approach 0 as $\alpha \rightarrow 0$.

In summary, reproducing the work of Woodworth and colleagues, we have suggested a problem in which training near the rich limit is preferable to training near the kernel limit. In particular, we have shown that for the linear regression problem 5, the rich limit corresponds to the minimum ℓ^1 solution to the system $\mathbf{X}\beta = \mathbf{y}$, whereas the kernel limit corresponds to the minimum ℓ^2 solution. Accordingly, we would expect that for problems in which the underlying data distribution ρ is sparse, training our network $f(\mathbf{w}, \mathbf{x})$ near the rich limit would achieve smaller population risk than had the model been trained near the rich limit.

5.1.1 Computational Limitations of Rich Training

So far in the discussion of our results we have ignored an obvious deficiency in rich training. Patently, in Figure 7 we observe that for α small, the loss of the gradient flow descent path $L(h(\mathbf{w}_\alpha(t)))$ does not converge within 10^{-4} of the global minimum loss value, 0. In fact, we only see this convergence for $\alpha \approx 1$, which is quite a large initialization scale. Also, we observe that for the gradient descent path with $N = 50$ training samples, the excess ℓ^1 norm and ℓ^1 population error begin to increase for sufficiently small α . This belies our previous assertion that both the excess ℓ^1 norm and ℓ^1 population error monotonically decrease as $\alpha \rightarrow 0$.

These seemingly anomalous results can be attributed to two facts about the problem. First, as we have previously pointed out, gradient descent is a discretization of the gradient flow dynamics, and so we would expect the gradient descent path to differ from the corresponding gradient flow. This is why I suspect the excess ℓ^1 norm and ℓ^1 population error begin to increase for the gradient descent path with $N = 50$: when the initialization scale is very small, and so the gradient descent is initialized with $\mathbf{w}_\alpha(0) \approx \mathbf{0}$, a very small stepsize is necessary to attain convergence.

Beyond just the coarseness of the stepsize, though, there is another aspect of the optimization problem that makes training close to the rich limit intractable. Namely, as one may have guessed from Figure 2, the objective $L(h(\mathbf{w}))$ often has a saddle point at $\mathbf{w} = \mathbf{0}$ [13]. Therefore by taking $\alpha \rightarrow 0$ in the rich limit, we are initializing the weight vector closer and closer to a saddle point of $L(h(\mathbf{w}))$. Undoubtedly, this small initialization complicates the optimization problem since it takes longer and longer for the gradient flow path to escape the vicinity of the saddle point. We conjecture that this is exactly the problem we are facing in our simulation: for α small, we are not able to escape the saddle point within the specified 10^4 epochs. In fact, for α sufficiently small, we would not expect the gradient flow path to ever depart from $\mathbf{0}$ within any reasonable amount of time.

This reality of training in the rich regime gives rise to a tradeoff between optimization and generalization as characterized by Woodworth and colleagues. That is, in practice, one must choose α sufficiently small to train near the rich limit and achieve the corresponding ℓ^1 implicit regularization in the network weights. Yet, one must avoid training with α too small or else the gradient descent path will never escape the saddle point $\mathbf{0}$ and attain the desired convergence.

5.2 A New Problem: Sparse Logistic Regression

In Section 5.1, we illustrated both theoretically and experimentally that for the linear regression problem (5), the kernel limit $\lim_{\alpha \rightarrow \infty} \boldsymbol{\beta}_{\mathbf{w}_\alpha^*}$ corresponds to the minimum ℓ^2 solution of the linear system $\mathbf{X}\boldsymbol{\beta} = \mathbf{y}$, whereas the rich limit $\lim_{\alpha \rightarrow 0} \boldsymbol{\beta}_{\mathbf{w}_\alpha^*}$ corresponds to the minimum ℓ^1 solution [13]. Consequently, for problems in which the population risk (9) can be minimized by a sparse weight vector, we would prefer training near the rich limit to training near its kernel counterpart.

Although this result is of significant merit, contemporary network architectures are substantially more complicated than the single-layer network we considered. Moreover, almost all of these networks are not differentiable in their weights $\mathbf{w} \in \mathbb{R}^p$. Consider, for example, the ReLU nonlinearity $\max\{0, x\}$ applied element-wise to the output of the hidden layers of our network f . Without the differentiability of our model h , none of the results formulated by Chizat and colleagues can be applied to tell us how the networks behave in the $\alpha \rightarrow \infty$ and $\alpha \rightarrow 0$ limits [4].

Still, as articulated by Woodworth and colleagues, we suspect that the kernel and rich limits do exist for these nondifferentiable problems. Furthermore, we conjecture that the rich limit $\alpha \rightarrow 0$ corresponds to some form of implicit ℓ^1 regularization, as it did for the linear regression problem. For the remainder of the paper, one should understand that we refer to the ‘‘kernel’’ and ‘‘rich’’ limits colloquially. Specifically, we no longer maintain the original differentiability assumptions made on h in Section 2, and so the gradient flow paths

of $L(h(\mathbf{w}))$ and $L(\bar{h}(\mathbf{w}))$ with $\mathbf{w}_\alpha(0) = \bar{\mathbf{w}}_\alpha(0) = \alpha\mathbf{w}_0$ do not necessarily exist. In practice, though, one can still employ gradient descent or a subgradient method to find a minimizer of the objective function.

5.2.1 Logistic Regression

In order to investigate whether or not rich training enforces implicit ℓ^1 regularization in the network weights outside of vanilla linear regression, we look at the sparse logistic problem posed by Wei and colleagues [11].

For this problem, we consider data $(\mathbf{x}, y) \sim \rho$ for $\mathbf{x} \in \mathcal{X} = \{+1, 0, -1\}^n$, $y \in \{+1, -1\}$. The distribution ρ is defined such that

$$\begin{aligned} y = +1, \quad \mathbf{x}^T e_1 &= +1, \quad \mathbf{x}^T e_2 = 0 \quad \text{w/ prob. } 1/4 \\ y = +1, \quad \mathbf{x}^T e_1 &= -1, \quad \mathbf{x}^T e_2 = 0 \quad \text{w/ prob. } 1/4 \\ y = -1, \quad \mathbf{x}^T e_1 &= 0, \quad \mathbf{x}^T e_2 = +1 \quad \text{w/ prob. } 1/4 \\ y = -1, \quad \mathbf{x}^T e_1 &= 0, \quad \mathbf{x}^T e_2 = -1 \quad \text{w/ prob. } 1/4 \end{aligned}$$

and $\mathbf{x}^T e_i \sim \{+1, -1\}$ is drawn uniformly at random for $3 \leq i \leq n$ [11]. From this definition of ρ , it should be clear that the output y is entirely determined by the first two coordinates of the input vector \mathbf{x} . Although we have not yet suggested a model $h : \mathbf{w} \mapsto f(\mathbf{w}, \mathbf{x})$, one should anticipate that the population risk is minimized by a sparse vector $\mathbf{w} \in \mathbb{R}^p$.

To associate a model with this data distribution ρ , the authors investigate the case of $h(\mathbf{w}) = f(\mathbf{w}, \mathbf{x})$ a simple network consisting of a single hidden layer with m neurons:

$$f(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^m v_j [\mathbf{u}_j^T \mathbf{x}]_+. \quad (10)$$

Here, $[x]_+$ is the ReLU activation $\max\{0, x\}$ and \mathbf{w} is the collection of all $m \times (n+1)$ parameters $\{(v_j, \mathbf{u}_j)\}_{j=1}^m$. Unlike the weights of the linear regression model, which are initialized deterministically, the weights of the ReLU neural network (10) are initialized randomly: $v \sim \mathcal{N}(0, r_v^2)$ and $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, r_u^2 \mathbb{I}_{n \times n})$ [11]. Also, we remark that the ReLU neural network is 2-positive homogeneous in its weights \mathbf{w} . Accordingly, we use the kernel limit to refer to the case in which $r_v, r_u \rightarrow \infty$ and the rich limit to refer to the case in which $r_v, r_u \rightarrow 0$.

Right away, we notice that our model $h : \mathbf{w} \mapsto f(\mathbf{w}, \mathbf{x})$ is not differentiable for all $\mathbf{w} \in \mathbb{R}^p$, as we previously addressed. Also, the network is not linear in its input space $\mathcal{X} \subseteq \mathbb{R}^n$, which is the case for most contemporary deep learning models. In this way, we view the model (10) as more advanced than the linear regression model (5) we previously studied.

Since our output values $y \in \{+1, -1\}$ are binary labels, we let L be the logistic loss function

$$L(f(\mathbf{w}, \mathbf{x})) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\log(1 + \exp(-y f(\mathbf{w}, \mathbf{x})))] = \frac{1}{N} \sum_{i=1}^N \log(1 + \exp(-y_i f(\mathbf{w}, \mathbf{x}_i))), \quad (11)$$

where \mathcal{D} is the empirical distribution determined by the training data $\{\mathbf{x}_i, y_i\}_{i=1}^N$ [11]. Just as the MSE (1) is the empirical risk corresponding to the square loss, (11) is simply the empirical risk corresponding to the $\{+1, -1\}$ logistic loss $\ell(y, y') = \log(1 + \exp(-y \cdot y'))$.

Having given both the data distribution ρ and the model $h : \mathbf{w} \mapsto f(\mathbf{w}, \mathbf{x})$, we justify that the problem we have defined is indeed sparse. Specifically, if we choose \mathbf{w} such that $\mathbf{u}_1 = e_1$, $\mathbf{u}_2 = -e_1$, $\mathbf{u}_3 = e_2$, $\mathbf{u}_4 = -e_2$, and $v_1 = v_2 = +1$, $v_3 = v_4 = -1$ then \mathbf{w} minimizes $\mathbf{w} \mapsto \mathbb{E}_{(\mathbf{x}, y) \sim \rho} [\mathbb{1}_{f(\mathbf{w}, \mathbf{x}) y \leq 0}] = \mathbb{P}(f(\mathbf{w}, \mathbf{x}) y \leq 0)$ [11]. Equivalently, the population risk is minimized by a network $f(\mathbf{w}, \mathbf{x})$ which contains only four neurons and eight nonzero weights. For this reason, we would expect the implicit ℓ^1 regularization which we hypothesize.

is associated with the rich limit $r_v, r_u \rightarrow 0$ to result in a model with good generalization (i.e. low population risk). Here, we are considering the population risk corresponding to the binary $\{0, +1\}$ loss $\ell(y, y') = \mathbb{1}_{y \cdot y' \leq 0}$, rather than the logistic loss defined above. This is because the main result of Wei and colleagues, Theorem 2.1, concerns the population risk corresponding to the binary loss.

Prior to experimenting with the logistic regression problem from [11], we provide more context for the authors' work and comment on how it relates to our research. First, Wei and colleagues provide an explicit characterization of the neural tangent kernel at the network (10) initialization:

$$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{v \sim \mathcal{N}(0, r_v), u \sim \mathcal{N}(0, r_u^2)} [\langle \nabla_{(v, u)} f(\mathbf{w}, \mathbf{x}), \nabla_{(v, u)} f(\mathbf{w}, \mathbf{x}') \rangle]$$

where $\nabla_{(v, u)} f(\mathbf{w}, \mathbf{x}) = ([\mathbf{u}^T \mathbf{x}]_+, v \mathbb{1}_{\mathbf{u}^T \mathbf{x} > 0} \mathbf{x}) \in \mathbb{R}^{n+1}$ is the gradient of f with respect to an arbitrary pair of weights (v, \mathbf{u}) (since they are initialized *i.i.d.*). We point out that for $\mathbf{u}^T \mathbf{x} = 0$, $\nabla_{(v, u)} f(\mathbf{w}, \mathbf{x})$ does not exist, although $([\mathbf{u}^T \mathbf{x}]_+, v \mathbf{x})$ is contained in the subdifferential of f at (v, \mathbf{u}) . Although the neural tangent kernels corresponding to the linear regression model (5) and ReLU neural network (10) are both deterministic at their initializations, the latter involves evaluating an expectation over the distribution of weight initializations (v, \mathbf{u}) .

Wei and colleagues are interested in the neural tangent kernel K because they study the predictor function f^{kernel} in the RKHS determined by K : $f^{\text{kernel}}(\mathbf{x}, \gamma) = \sum_{i=1}^N \gamma_i K(\mathbf{x}_i, \mathbf{x})$. More explicitly, from the theory of Jacot and colleagues [9], the authors remark that by performing gradient descent in the infinite-width limit, one would obtain such a function f^{kernel} . Making no theoretical claims, the results of Chizat and colleagues [7] suggest the possibility that training with large r_v, r_u may similarly result in such a kernel predictor function.

Now that we have introduced f^{kernel} , we state the main result formulated and proven and proven by Wei and colleagues:

Theorem 2.1 (from [11]). *Let ρ be the distribution defined previously. With probability $1 - n^{-5}$ over the random draw of $N \lesssim n^2$ samples $(x_1, y_1), \dots, (x_N, y_N)$ from ρ , for all choices of γ , the kernel prediction function $f^{\text{kernel}}(\cdot, \gamma)$ will have at least $\Omega(1)$ error*

$$\mathbb{P}_{(\mathbf{x}, y) \sim \rho} [f^{\text{kernel}}(\mathbf{x}; \gamma) y \leq 0] = \Omega(1).$$

Moreover, let us define the ℓ^2 regularized logistic loss L_λ

$$L_\lambda(f(\mathbf{w}, \mathbf{x})) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\log(1 + \exp(-y f(\mathbf{w}, \mathbf{x}))) + \lambda \|\mathbf{w}\|_F^2],$$

where $\mathbf{w} \in \mathbb{R}^{m \times (n+1)}$, $\mathbf{w}_{i,:} = (v_i, \mathbf{u}_i)$ is the matrix containing the network parameters. Then for $\lambda \leq \text{poly}(N)^{-1}$, the regularized neural net solution $f(\mathbf{w}_\lambda, \mathbf{x})$ trained with at least four hidden units can have good approximation with $\mathcal{O}(n^2)$ samples because we have the following generalization error bound:

$$\mathbb{P}_{(\mathbf{x}, y) \sim \rho} [f(\mathbf{w}_\lambda, \mathbf{x}) y \leq 0] \lesssim \sqrt{\frac{n}{N}}.$$

This implies a $\Omega(n)$ sample-complexity gap between the regularized neural net and kernel prediction function.

Plainly, Theorem 2.1 states that the gradient descent on the ℓ^2 regularized objective $L_\lambda(f(\mathbf{w}, \mathbf{x}))$ achieves a model which generalizes well over the population distribution of data ρ with only n^2 samples, whereas any estimator contained in the RKHS determined by the neural tangent kernel K does not.

Now that we have detailed this result from [11], we compare the ℓ^2 regularized gradient descent solution \mathbf{w}_λ to the unregularized gradient descent solution \mathbf{w}_{r_v, r_u}^* (i.e. the gradient descent solution using the unregularized loss (11)) in the rich limit $r_v, r_u \rightarrow 0$. Namely, as we have previously discussed, we know that there exists implicit ℓ^0 sparsity in the logistic regression problem we have posed: using only eight nonzero weights, one can minimize the population risk $\mathbb{E}_{(\mathbf{x}, y) \sim \rho} [\mathbb{1}_{y f(\mathbf{w}, \mathbf{x}) \leq 0}]$. To impose this sparsity in their gradient descent

solution, Wei and colleagues utilize an explicit ℓ^2 regularizer in their loss function L_λ . However, from our knowledge of simple regularized problems, such as LASSO and ridge regression, we are doubtful whether an explicit ℓ^2 regularization term is the best choice to capture the sparsity in the network parameters. Rather, we hypothesize that ℓ^1 regularization would achieve better model generalization. It is for this reason that we are interested in the rich limit $r_v, r_u \rightarrow 0$: if gradient descent in the rich limit does indeed correspond to implicit ℓ^1 regularization, then we would expect that models trained near the rich limit should achieve better generalization than $f(\mathbf{w}_\lambda, \mathbf{x})$. We reiterate that our suspicion that the “rich limit” $r_v, r_u \rightarrow 0$ enforces ℓ^1 sparsity is no more than an educated guess and does not fall under the existing theory of [7].

5.2.2 Computational Experiments Near the Rich Limit

To explore the possibility that training the ReLU neural network (10) near the rich limit $r_v, r_u \rightarrow 0$ for the sparse logistic regression problem corresponds to implicit ℓ^1 regularization, we conduct a couple of experiments.

Explicitly, we will consider the gradient descent path $(\mathbf{w}_{r_v, r_u}(t))_{t \in \mathbb{N} \cup \{0\}}$ with $\mathbf{w}_{r_v, r_u}(0)$ initialized according to $v_i \sim \mathcal{N}(0, r_v^2)$, $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, r_u^2 \mathbb{I}_{n \times n})$ for each hidden unit $1 \leq i \leq m$. Differing from our experiments in Section 5.1, we mention that the starting point of the gradient descent path is random. Accordingly, rather than simulating gradient descent once for each r_v, r_u , we consider K gradient descent paths, each of which is initialized from a distinct random starting point $\mathbf{w}_{r_v, r_u}(0)$. And so to estimate $\mathbb{P}_{(x, y) \sim \rho}[f(\mathbf{w}_{r_v, r_u}^\star, \mathbf{x})y \leq 0]$ for each r_v, r_u , we average the test error $\sum_{i=1}^{N_{\text{test}}} \mathbb{1}_{y_i f(\mathbf{w}_{r_v, r_u}(t_{\text{end}}), \mathbf{x}_i) \leq 0}$ over the K gradient descent paths. Here, $\mathbf{w}_{r_v, r_u}^\star$ denotes the limit $t \rightarrow \infty$ of the gradient descent path $\mathbf{w}_{r_v, r_u}(t)$ and $t_{\text{end}} \in \mathbb{N} \cup \{0\}$ denotes the final epoch of gradient descent considered in our simulations. For each of our simulations we use a test dataset of $N_{\text{test}} = 10^3$ points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{\text{test}}}$, where each $(\mathbf{x}_i, y_i) \sim \rho$ for $1 \leq i \leq N_{\text{test}}$.

For our first experiment, we address the question of whether or not there exists some implicit ℓ^1 regularization in the rich limit $r_v, r_u \rightarrow 0$. In order to do so, we will consider an $n = 20$ dimensional input space and a ReLU neural network (10) with $m = 10$ hidden units. Observe that since there are $m \times (n + 1)$ total parameters in the network, then it is indeed overparameterized, as was the case for the linear regression model (5). We consider three different values for the training sample size $N \in \{200, 100, 50\}$ to see how the network performance changes as a function of N . Also, for each N , we consider three different values for the initialization scale of our weights, $r_v = r_u \in \{1, 10^{-1}, 10^{-2}\}$. Each of the experiments conducted by Wei and colleagues only considers the initialization $r_v = r_u = 1$, which we believe corresponds to gradient descent far from the rich limit.

As for the specifics of how we perform the optimization, we strive to mimic the procedures implemented by Wei and colleagues in [11]. More precisely, we conduct $K = 20$ simulations of gradient descent for each pair of training sample size N and initialization scale $r_v = r_u$. Further, we train for $t_{\text{end}} = 2 \times 10^4$ total epochs with learning rate $\eta = 10^{-1}$. As we pointed out in Section 4, choosing our learning rate η to be this large is undesirable insofar as it means the gradient descent path is far from the corresponding gradient flow dynamics, should they exist. Also, just as in Sections 4 and 5.1, we use batch size 32 when calculating the gradient for each update step.

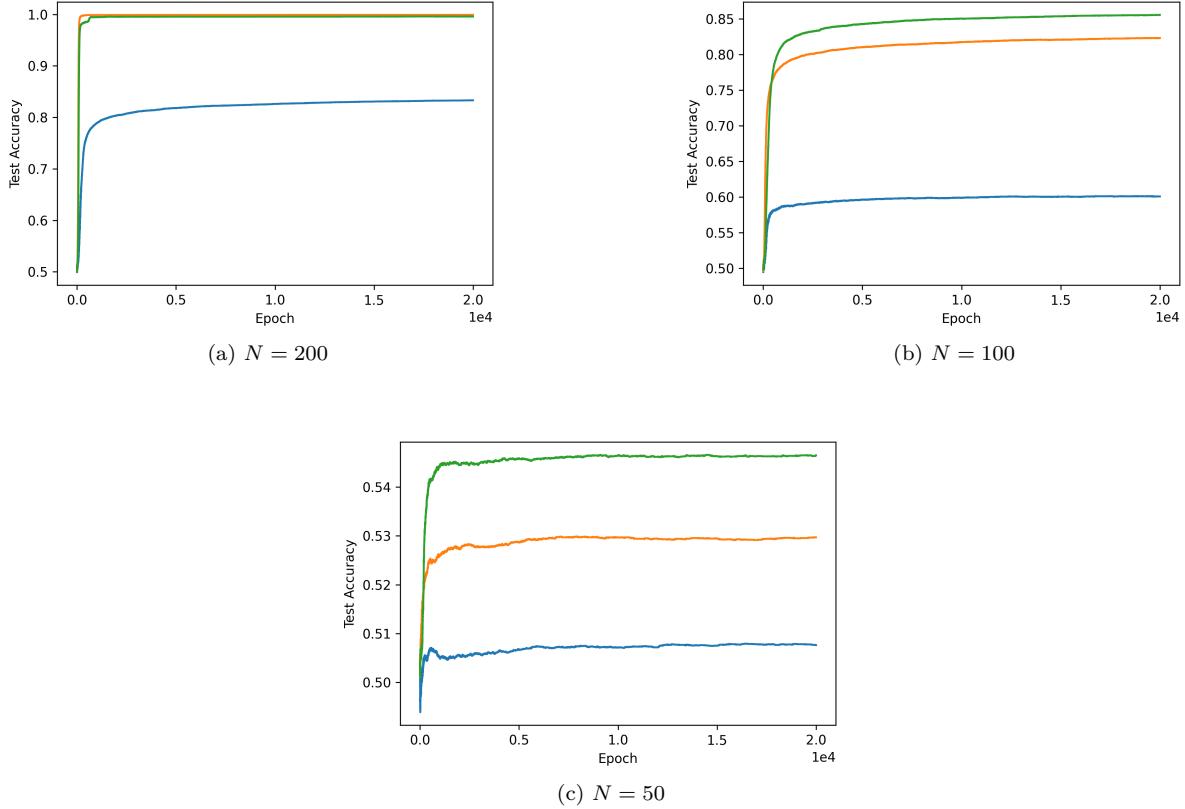


Figure 9: Average test accuracy versus epoch for $K = 20$ gradient descent paths. The test accuracy at epoch $t \in \mathbb{R}_+$ is computed according to $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}_{y_i f(\mathbf{w}_{rv,ru}(t), \mathbf{x}_i) \leq 0}$. The three paths in each plot correspond to the initialization scales $r_v = r_u = 1$ (blue), 10^{-1} (orange), and 10^{-2} (green).

The results of these experiments, summarized by Figure 9, provide strong evidence that training with small initialization scale leads to better generalization in the sparse logistic regression problem. Precisely, for each of $N \in \{200, 100, 50\}$ we see that the test accuracy averaged over the $K = 20$ gradient descent paths is markedly higher throughout the entirety of training for small initialization scale $r_v = r_u$. Before any further analysis, we rule out the possibility that these disparities in test error are a result of differences in performance on the training set: each gradient descent path, regardless of the initialization scale $r_v = r_u$, achieves perfect or near perfect training accuracy within only about 50 to 100 epochs. This is what we would expect, though, as the ReLU network (10) is overparameterized, implying that there are many global minima of the objective $L(h(\mathbf{w}))$ to which the gradient descent path can converge.

The aforementioned gap in test error is most extreme for $N = 200$: when $r_v = r_u \in \{10^{-1}, 10^{-2}\}$, the test accuracy jumps to 1 almost immediately, whereas when $r_v = r_u = 1$ the test accuracy flattens off at around 0.8. Similarly, for $N = 100$ the average test accuracy corresponding to $r_v = r_u = 10^{-2}$ is excellent, upwards of 0.85, whereas that corresponding to $r_v = r_u = 1$ is only slightly better than random guessing, around 0.6. As evidenced by the Figure 9c, even when the number of training observations N is very small in comparison to the input dimension n , we still see gains from initializing the network weights \mathbf{w} with small scale $r_v = r_u$. Although these results do not explicitly confirm the presence of ℓ^1 regularization in the “rich limit” $r_v = r_u \rightarrow 0$, they do suggest some form of implicit bias that leads to better generalization properties for small r_v, r_u .

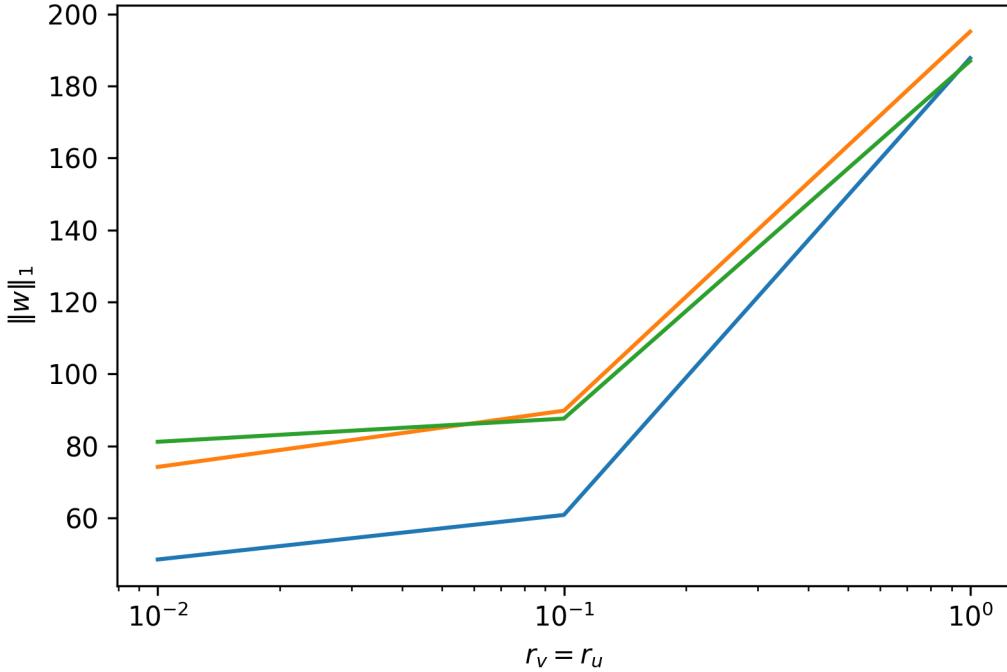


Figure 10: The average ℓ^1 norm of the gradient descent path at the end of training, $\mathbf{w}_{r_v, r_u}(t_{\text{end}})$ for $K = 20$ simulations. We include the gradient descent simulations corresponding to $N = 200$ (blue), $N = 100$ (orange), and $N = 50$ training samples.

In Figure 10, we address the question of whether or not there exists implicit ℓ^1 regularization, in particular, in the gradient descent solution vector $\mathbf{w}_{r_v, r_u}(t_{\text{end}})$. Excitingly, for each of $N \in \{200, 100, 50\}$ we discern drastic decreases in the average ℓ^1 norm of the gradient descent solution \mathbf{w}_{r_v, r_u} as $r_v = r_u$ decreases from 1 to 10^{-1} . That is, for r_v, r_u small, the neural network solution reached by gradient descent is much more sparse in its weights than when the network is trained with large initialization r_v, r_u . We also notice that while there is a sharp decrease in the average ℓ^1 norm of the gradient descent solution vector when shifting the initialization scale from $r_v = r_u = 1$ to $r_v = r_u = 10^{-1}$, the decrease is relatively small for the shift from 10^{-1} to 10^{-2} . We suspect that this phenomenon is a consequence of the large stepsize that we previously flagged: since the stepsize $\eta = 10^{-1}$ is so large in comparison to the initialization scales $r_v = r_u \in \{10^{-1}, 10^{-2}\}$, the difference between the gradient descent paths corresponding to these two initialization scales is likely almost negligible. Perhaps by performing gradient descent with a smaller stepsize, we would observe a greater distinction between the average ℓ^1 norm of the gradient flow solution vectors corresponding to $r_v = r_u = 10^{-1}$ and $r_v = r_u = 10^{-2}$.

In all, we have provided compelling evidence that training in the rich limit $r_v, r_u \rightarrow 0$ enforces implicit ℓ^1 regularization of the gradient descent solution $\lim_{t \rightarrow \infty} \mathbf{w}_{r_v, r_u}(t)$. This result in and of itself is noteworthy, as neither the existing theory of Chizat nor that of Woodworth would suggest this to be the case.

The next step in our investigation of the sparse logistic regression problem is to compare the generalization of the implicit ℓ^1 regularization we have demonstrated in the limit $r_v, r_u \rightarrow 0$ with the explicit ℓ^2 regularization employed by Wei and colleagues [11]. One will recall from Theorem 2.1 that for a neural network with at least four hidden units, training with the ℓ^2 regularized logistic loss achieves generalization error

$$\mathbb{P}_{(x,y) \sim \rho}[f(\mathbf{w}_\lambda, \mathbf{x})y \leq 0] \lesssim \sqrt{\frac{n}{N}}.$$

But as we previously mentioned, we believe that ℓ^1 regularization would be preferred to ℓ^2 regularization to produce a weight vector \mathbf{w} that is ℓ^0 sparse. Consequently, it is our conjecture that the implicit ℓ^1 regularization in the limit $r_v, r_u \rightarrow 0$ achieves a generalization bound at least as good as, if not better than, that in Theorem 2.1.

In order to get an approximation of the bound on $\mathbb{P}_{(\mathbf{x}, y) \sim \rho}[f(\mathbf{w}_{r_v, r_u}, \mathbf{x})y \leq 0]$, we must see how the number of training samples necessary to attain population risk less than some threshold p^* changes as a function of the input dimension n . For our experiment, we fix $r_v = r_u = 10^{-1}$ and take $p^* = 0.4$. Since in our previous simulations we demonstrated that the gradient descent solution vector $\mathbf{w}_{r_v=r_u=10^{-1}}(t_{\text{end}})$ is subject to significant ℓ^1 regularization, then this is a good choice for $r_v = r_u$. Just as in the previous set of experiments, we use the test misclassification error $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}_{y_i f(\mathbf{w}_{r_v, r_u}(t_{\text{end}}), \mathbf{x}_i) \leq 0}$ averaged over K gradient descent trials as a proxy for the population risk. And so for each value of the input dimension n , we would like to find the smallest number of training samples N_n^* needed to obtain average test accuracy ≥ 0.6 averaged over K trials.

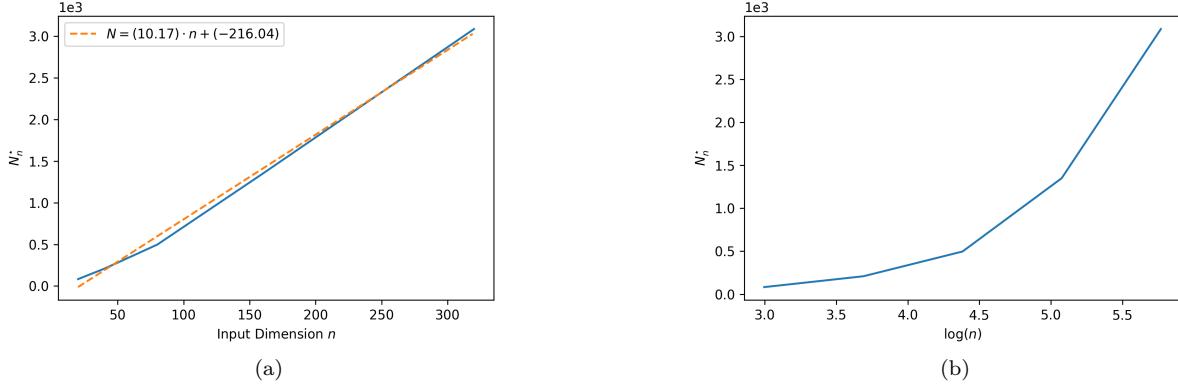


Figure 11: The minimum number of training samples, N_n^* , necessary to attain average test accuracy ≥ 0.6 with input dimension n . $K = 10$ gradient descent simulations are used to determine the average test error. (a) We fit a simple linear regression model to estimate the constant C such that $\mathbb{P}_{(\mathbf{x}, y) \sim \rho}[f(\mathbf{w}_{r_v, r_u}, \mathbf{x})y \leq 0] \leq \sqrt{\frac{C \cdot d}{n}}$ (b) The input dimension n is replaced with $\log(n)$.

Undoubtedly, this is a computationally burdensome experiment: for each input dimension n , we must search over values of N , performing K gradient descent simulations each time, to find the smallest value which achieves average test accuracy $\geq 1 - p^*$. Accordingly, it is necessary that we make modifications to our previous experimental conditions. Specifically, we reduce the total number of training epochs to 10^4 and the number of gradient trials to $K = 10$. The results from our first experiment presented in Figure 9 suggest that we do not witness significant increases in the test accuracy past 10^4 epochs, and so this reduction in epochs is reasonable. And as for the input dimension, we consider values $n \in \{20, 40, 80, 160, 320\}$. As we will discuss later in our analysis, this is quite a restricted range of values for n .

Despite the limitations of our experiment, the results we obtain are quite compelling. In Figure 1, we plot the input dimension n against the minimum number of training samples N_n^* necessary to achieve test accuracy $\geq 1 - p^* = 0.6$. We observe that there exists an almost perfectly linear relationship between n and N_n^* . That is, it appears that a Δn increase in the input dimension n corresponds to a $C \cdot \Delta n$ increase in N_n^* for some constant $C \in \mathbb{R}_{++}$. In Figure 11a, we obtain an explicit estimate for the value of the slope coefficient C by fitting a linear regression model. The regression model estimates $C \approx 10.17$ and has coefficient of determination $r^2 = 0.996$.

The reason why this linear fit is so interesting is that it suggests a bound on the generalization error of the

gradient descent solution $\mathbf{w}_{r_v=r_u=10^{-1}}^*$ of the form

$$\mathbb{P}_{(x,y) \sim \rho}[f(\mathbf{w}_{r_v=r_u=10^{-1}}^*, \mathbf{x})y \leq 0] \lesssim \sqrt{\frac{n}{N}}.$$

That is, we have provided evidence to indicate that the generalization bound we achieve by shrinking the scale of the initialization $r_v = r_u$ is on the same order as that we achieve by adding an explicit ℓ^2 regularization term to the logistic loss L [11]. Candidly, this is quite a extraordinary experimental result: just by choosing the initialization scale $r_v = r_u$ to be small (i.e. using no explicit regularization whatsoever), we appear to get the same generalization error bound as had we used an explicit ℓ^2 regularizer.

To conclude our discussion, we consider the possibility that the implicit ℓ^1 regularization present in the rich limit $r_v = r_u \rightarrow 0$ outperforms explicit ℓ^2 regularization. So far, we have only discussed the possibility that the implicit ℓ^1 regularization performs just as well as the explicit ℓ^2 regularization. However, from our prior knowledge of sparse problems, we believe that an ℓ^1 regularizer might attain a better bound on the generalization error,

$$\mathbb{P}_{(x,y) \sim \rho}[f(\mathbf{w}_{r_v=r_u=10^{-1}}^*, \mathbf{x})y \leq 0] \lesssim \sqrt{\frac{\log(n)}{N}},$$

than its ℓ^2 counterpart. While our experimental results do not suggest such a bound, they also do not rule it out either. Expressly, as evidenced by Figure 11b, we do not observe a linear relationship between $\log(n)$ and N_n^* . This being said, we also comment that the range of values we have for $\log(n)$ is so small that, if such a bound were to exist, it would be difficult to detect in our experimental data anyways. Accordingly, if we wanted to more rigorously consider whether or not a bound of this order exists, we would need to perform the same experiment with much larger n . Also, we point out that if the limit $r_v, r_u \rightarrow 0$ does truly correspond to ℓ^1 regularization of the gradient descent limit \mathbf{w}_{r_v,r_u}^* , then by taking $r_v = r_u = 10^{-1}$ we are only approximating the ℓ^1 regularization present in the rich limit. Consequently, by choosing $r_v = r_u$ to be even smaller with appropriate decreases in the learning rate η , we can better estimate the generalization error bound in the underlying rich limit.

6 Conclusion

As a whole, our work has provided a comprehensive look at two opposing limits present in neural network training. Recall that we began our paper by formalizing the kernel and rich limits, explaining what they mean in terms of the gradient flow on the scaled objective $\frac{1}{\alpha^2}L(\alpha h(\mathbf{w}))$, $(\mathbf{w}_\alpha(t))_{t \geq 0}$, and under what conditions they are each observed. Following this discussion, we conducted a number of experiments to visualize each of the network function, the neural tangent kernel, and the network weights as gradient flow interpolates between the kernel and rich regimes. The visualizations we produced provide fresh intuition for the kernel and rich limits that is not captured by the theory alone. And for the most exciting portion of our research, we investigated how networks trained near the kernel and rich limits generalize in problems where the underlying distribution of data ρ is sparse. Our experimental results corresponding to the sparse logistic regression problem posed by Wei and colleagues are especially invigorating. They suggest that by simply reducing the scale with which the network weights are initialized, one achieves a generalization bound that is on the same order as that achieved by using an explicit ℓ^2 regularizer in the loss function L .

Plainly, our experimental results make a strong case for further theoretical study into the rich limit. Having provided evidence that the rich limit $r_v, r_u \rightarrow 0$ indeed corresponds to implicit ℓ^1 regularization in the sparse logistic regression problem, the next step would be to formalize this observation with a theoretical statement. Even more generally, there is reason to believe that in the limit as the initialization scale of the weights tends to zero, the same implicit ℓ^1 regularization of the gradient descent solution \mathbf{w}^* would occur for other, perhaps more applicable, models.

References

- [1] S. ARORA, N. COHEN, W. HU, AND Y. LUO, *Implicit regularization in deep matrix factorization*, Advances in Neural Information Processing Systems, 32 (2019), pp. 7413–7424.
- [2] A. R. BARRON, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information theory, 39 (1993), pp. 930–945.
- [3] ———, *Approximation and estimation bounds for artificial neural networks*, Machine learning, 14 (1994), pp. 115–133.
- [4] L. CHIZAT AND F. BACH, *A note on lazy training in supervised differentiable programming*, arXiv preprint arXiv:1812.07956, 8 (2018).
- [5] ———, *On the global convergence of gradient descent for over-parameterized models using optimal transport*, arXiv preprint arXiv:1805.09545, (2018).
- [6] ———, *Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss*, in Conference on Learning Theory, PMLR, 2020, pp. 1305–1338.
- [7] L. CHIZAT, E. OYALLON, AND F. BACH, *On lazy training in differentiable programming*, arXiv preprint arXiv:1812.07956, (2018).
- [8] S. GUNASEKAR, B. WOODWORTH, S. BHOJANAPALLI, B. NEYSHABUR, AND N. SREBRO, *Implicit regularization in matrix factorization*, in 2018 Information Theory and Applications Workshop (ITA), IEEE, 2018, pp. 1–10.
- [9] A. JACOT, F. GABRIEL, AND C. HONGLER, *Neural tangent kernel: Convergence and generalization in neural networks*, arXiv preprint arXiv:1806.07572, (2018).
- [10] C. MA, L. WU, ET AL., *A priori estimates of the population risk for two-layer neural networks*, arXiv preprint arXiv:1810.06397, (2018).
- [11] C. WEI, J. LEE, Q. LIU, AND T. MA, *Regularization matters: Generalization and optimization of neural nets vs their induced kernel*, (2019).
- [12] A. WIBISONO, *Gradient flow and gradient descent*. <http://awibisono.github.io/2016/06/13/gradient-flow-gradient-descent.html>, July 2016.
- [13] B. WOODWORTH, S. GUNASEKAR, J. D. LEE, E. MOROSHKO, P. SAVARESE, I. GOLAN, D. SOUDRY, AND N. SREBRO, *Kernel and rich regimes in overparametrized models*, in Conference on Learning Theory, PMLR, 2020, pp. 3635–3673.