

# S&DS Senior Project Proposal: Kernel and Rich Regimes in Deep Learning

Henry Smith  
Advised by Professor Harrison Zhou  
Yale University

January 29, 2022

## 1 Problem Overview

Much of the popularity and success of contemporary deep learning models can be attributed to their ability to generalize well when presented with unseen data. Indeed, researchers have constructed convolutional neural networks (CNNs) that achieve  $< 1\%$  test error on the MNIST dataset, outpacing other popular machine learning classifiers (k-nearest neighbors, support vector machines, etc.). Despite the widespread application of neural networks, practitioners remain largely unsure about the mathematical justification for why they generalize so well.

Broadly speaking, neural networks are highly overparameterized; that is, the dimension of the parameter space  $\mathbf{w} \in \mathbb{R}^p$  is much larger than the number of observations  $N$  with which the network is trained. As a result, the loss function  $L$ —which measures the misfit of the model for the training data  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ —often has many global minima where zero training error is achieved. As keenly noted in [5], though, not all of these minima result in a model  $f$  which generalizes equally well.

The goal of the present project is to understand the conditions under which training an overparameterized neural network will lead to a solution that generalizes well. Both [1] and [5] investigate this problem and establish a connection between a network’s initialization and the solution reached by the gradient flow dynamics. In particular, consider a network  $f$  initialized with weight vector  $\alpha \mathbf{w}_0$  and trained with a fixed  $N$ -many observations. Here,  $\alpha > 0$  serves as a scaling parameter and  $\mathbf{w}_0$  is the “shape” of the initialization. Under certain conditions on the model  $f$  and loss function  $L$ , it is observed that as  $\alpha \rightarrow \infty$ , the gradient flow solution generalizes poorly on test data. [1] and [5] refer to this limit as the “lazy” or “kernel” regime. Conversely, as  $\alpha \rightarrow 0$  we observe the “rich limit” in which the gradient flow solution generalizes well. In the common setting of a neural network with weights initialized  $\mathbf{w}_i \stackrel{i.i.d.}{\sim} \mathcal{W}$ , one would analogously look at  $\text{Var}(\mathcal{W})$ , with the kernel regime induced as  $\text{Var}(\mathcal{W}) \rightarrow \infty$  and the rich limit as  $\text{Var}(\mathcal{W}) \rightarrow 0$ .

Naturally, one might ask why one cannot take the initialization scale  $\alpha$  to be arbitrarily small so as to achieve a model with the desired test error. It is important to note, though, that for many problems,  $\mathbf{w} = \vec{0}$  is a saddle point for the objective  $L(f(\mathbf{w}))$ , and so taking  $\alpha$  too small makes the optimization problem computationally intractable. Consequently, [5] discusses the ideal scenario in which one should choose an initialization scale that acts “at the edge of the rich limit.” That is, one should choose  $\alpha$  small enough so that the solution realized by gradient flow generalizes well (i.e. is in the “rich” training regime), but is also not so small that optimization is infeasible.

Woodworth and colleagues in [5] posit that the contemporary successes in deep learning are a result of models

that operate in the rich regime. The implications of this statement are self-evident, and they warrant further computational and theoretical study of the kernel and rich limits.

## 2 Problem Statement & Methodology

### 2.1 Observing the Kernel Regime

The first portion of the project will focus on visualizing and reproducing the results from [1] and [5]. Underlying the previously mentioned kernel and rich regimes, there is rich literature discussing what happens mathematically as these limits are reached.

Without digging too deep into the weeds, suppose that model  $f$  maps from the parameter space  $\mathbb{R}^p$  to Hilbert space  $\mathcal{F}$  that represents the space of possible functions our neural network can represent.<sup>1</sup> Further, suppose that  $f$  is a  $D$ -homogeneous function, meaning  $f(\alpha \mathbf{w}) = \alpha^D f(\mathbf{w})$ . Under certain technical conditions on model  $f$  and loss  $L$  detailed in [1], then as  $\alpha \rightarrow \infty$  in the initialization  $\mathbf{w}(0) = \alpha \mathbf{w}_0$ , we have that  $f$  approaches the affine model

$$\bar{f}(\mathbf{w}) := \alpha^D f(\mathbf{w}(0)) + \langle \nabla f(\mathbf{w}(0)), \mathbf{w} - \mathbf{w}(0) \rangle \quad (1)$$

in the Hilbert space norm for all times  $t \geq 0$  in the gradient flow dynamics [1]. That is, in the kernel regime, training our model  $f$  is exactly equivalent to training an affine model  $\bar{f}$ . Notice that for this affine model  $\bar{f}$ , the gradient  $\nabla \bar{f}(\bar{\mathbf{w}}(t)) = \nabla f(\bar{\mathbf{w}}(0))$  for all times  $t \geq 0$ , assuming that  $\bar{\mathbf{w}}(t)$  evolves according to the gradient flow dynamics with  $\bar{\mathbf{w}}(0) = \alpha \mathbf{w}_0$ .

Therefore, we can look at the *neural tangent kernel* (NTK)

$$K_{\mathbf{w}(t)}(\mathbf{x}, \mathbf{x}') = \langle \nabla_{\mathbf{w}} f(\mathbf{w}(t))(\mathbf{x}), \nabla_{\mathbf{w}} f(\mathbf{w}(t))(\mathbf{x}') \rangle$$

as a proxy for the kernel regime [2]. Specifically, as a model approaches the kernel limit, we should observe that the NTK remains almost constant throughout training.

This brings me to my first goal for the project: to visualize how the NTK evolves throughout training for the rich versus kernel regimes. To do so, I will generate a training dataset in Python which corresponds to an appropriate model. Of particular interest is the linear regression model considered in [5]:

$$f(\mathbf{w})(\mathbf{x}) = \sum_{i=1}^d (\mathbf{w}_{+,i}^2 - \mathbf{w}_{-,i}^2) \mathbf{x}_i = \langle \beta_{\mathbf{w}}, \mathbf{x} \rangle, \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_+ \\ \mathbf{w}_- \end{bmatrix} \in \mathbb{R}^{2d}, \quad \beta_{\mathbf{w}} = \mathbf{w}_+^2 - \mathbf{w}_-^2. \quad (2)$$

I plan to use TensorFlow, a deep learning library for Python, to construct and optimize the model with weight initialization  $\alpha \mathbf{w}_0$ . Training will be performed using gradient descent, which can be thought of as a discretization of the gradient flow dynamics with some stepsize  $\eta > 0$ . During each  $n$ th iteration of the training procedure, I will evaluate the NTK at the training points  $\mathbf{x}_1, \dots, \mathbf{x}_N$ . This will allow us to visualize the changing of the NTK as we interpolate between the kernel and rich regimes.

In addition to the neural tangent kernel, we will also study how the network weights  $\mathbf{w}$  evolve throughout training. From our previous discussion of the kernel regime, we know that as  $\alpha \rightarrow \infty$ , the gradient flow of  $L(f(\mathbf{w}))$  approaches that of  $L(\bar{f}(\mathbf{w}))$ . Accordingly, we can observe the  $\ell_2$  distance between the weights of the model,  $\mathbf{w}(t)$ , and those of the affine model (1),  $\bar{\mathbf{w}}(t)$ , for various values of  $\alpha$  during training. We will do so using the linear regression model (2) implemented in Python as described in the previous paragraph.

<sup>1</sup>A common example is  $\mathcal{F} = L^2(\rho_x, \mathbb{R}^k)$ , where  $\rho_x$  is the distribution of training observations.

## 2.2 The Kernel Regime & Model Generalization

Moving away from the mathematics motivating the kernel regime, we will look at the generalization of neural networks trained under the kernel and rich regimes. From the prior section, we know that for a fixed number of training observations  $N$ , overparameterized networks trained in the rich regime generally achieve smaller test errors than those trained in the kernel regime. We will empirically demonstrate that this observation holds for the linear regression model (2), which was previously illustrated by Woodworth and colleagues in [5]. Differing from Woodworth’s analysis, though, we will also consider the time necessary to train a given model. That is, we will plot the number of iterations  $n_{\alpha,\kappa}$  of gradient descent necessary to attain training error below some threshold  $\kappa$  as a function of the initialization scale  $\alpha$ . We expect that as we get closer and closer to the rich limit  $\alpha \rightarrow 0$ , the resulting models will become more and more costly to train.

Moving past the example of linear regression, we hope to consider other interesting models, such as the [overparameterized] logistic regression model studied in [4]. Wei and colleagues establish the mathematical result that training the logistic model in the kernel regime leads to poor generalization, which can be mitigated with the addition of a regularization term. It would be especially appealing to investigate whether by training the unregularized logistic regression model in the rich regime instead, we could achieve comparable test error to that of the regularized model. Of further interest are sparse problems, where the underlying model  $f$  from which the data  $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$  is generated exhibits sparsity in its parameters [3]. We would expect that, since the parameters  $\mathbf{w}$  change less during training when in the kernel regime than in the rich regime (see [1]), we would observe much smaller test error in the rich regime.

## 3 Deliverables

1. Multiple deep learning models implemented in Python with TensorFlow
2. Python visualization of the neural tangent kernel  $K_{\mathbf{w}(t)}$  while training in the rich and kernel regimes
3. Plot of the  $\ell_2$  distance between the weights of the original model,  $\mathbf{w}(t)$ , and those of the affine model (1),  $\bar{\mathbf{w}}(t)$ , throughout training with gradient descent for various  $\alpha$
4. Graphs of test error versus initialization scale for various overparameterized models (linear regression (2), logistic regression, sparse)
5. Analysis of the relationship between the initialization scale  $\alpha$  and number of training iterations  $n_{\alpha,\kappa}$  for the linear regression model

## 4 Tentative Timeline

Literature review, discussion with Professor Zhou	January 25 – February 6
Code linear regression (2) and affine (1) models	January 30 – February 20
Visualize the NTK, weights during training	February 20 – March 6
Examine test loss versus initialization scale for (2)	March 6 – March 20
Code logistic regression model, examine test loss	March 20 – April 10
Code sparse model, examine test loss	April 10 – April 24
Prepare final report, poster	April 17 – May 5

## References

- [1] L. CHIZAT, E. OYALLON, AND F. BACH, *On lazy training in differentiable programming*, arXiv preprint arXiv:1812.07956, (2018).
- [2] A. JACOT, F. GABRIEL, AND C. HONGLER, *Neural tangent kernel: Convergence and generalization in neural networks*, arXiv preprint arXiv:1806.07572, (2018).
- [3] C. MA, L. WU, ET AL., *A priori estimates of the population risk for two-layer neural networks*, arXiv preprint arXiv:1810.06397, (2018).
- [4] C. WEI, J. LEE, Q. LIU, AND T. MA, *Regularization matters: Generalization and optimization of neural nets vs their induced kernel*, (2019).
- [5] 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.