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

Henry Smith
Advised by Professor Harrison Zhou

Yale University

January 14, 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 $\boldsymbol{w} \in \mathbb{R}^p$ 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 $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$—often has many global minima where zero training error is achieved. As keenly noted in [4], 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 [2] and [4] 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 $\alpha \boldsymbol{w}_0$ and trained with a fixed N-many observations. Here, $\alpha > 0$ serves as a scaling parameter. Under certain conditions on the model f and loss function L, it is observed that as $\alpha \to \infty$, the gradient flow solution generalizes poorly on test data. [2] and [4] refer to this limit as the "lazy" or "kernel" regime. Conversely, as $\alpha \to 0$ we observe the "rich limit" in which the gradient flow solution generalizes well. In the common setting of a neural network with weights initialized $\boldsymbol{w}_i \overset{i.i.d.}{\sim} \mathcal{W}$, one would analogously look at $\mathrm{Var}(\mathcal{W})$, with the kernel regime induced as $\mathrm{Var}(\mathcal{W}) \to \infty$ and the rich limit as $\mathrm{Var}(\mathcal{W}) \to 0$.

Naturally, one might ask why one cannot take the initialization scale α 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, $\boldsymbol{w} = \vec{0}$ is a saddle point for the objective $L(f(\boldsymbol{w}))$, and so taking α too small makes the optimization problem computationally intractable. Consequently, [4] 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 α 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 [4] 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 Computational Experiments

The first portion of the project will focus on reproducing and extending the results from [2] and [4]. 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.[1] Assuming that our model f and loss L satisfy some technical conditions detailed in [2], then as $\alpha \to \infty$ in the initialization $\boldsymbol{w}(0) = \alpha \boldsymbol{w}_0$, we have that f approaches the affine model

$$\bar{f}(\boldsymbol{w}) := f(\boldsymbol{w}(0)) + \langle \nabla f(\boldsymbol{w}(0)), \boldsymbol{w} - \boldsymbol{w}(0) \rangle \tag{1}$$

in the Hilbert space norm for all times $t \geq 0$ in the gradient flow dynamics [2]. 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 $\nabla \bar{f}(\bar{\boldsymbol{w}}(t)) = \nabla f(\bar{\boldsymbol{w}}(0))$ for all times $t \geq 0$, assuming that $\bar{\boldsymbol{w}}(t)$ evolves according to the gradient flow dynamics with $\bar{\boldsymbol{w}}(0) = \alpha \boldsymbol{w}_0$.

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

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

as a proxy for the kernel regime [3]. Specifically, as a model approaches the kernel limit, we should observe that the NTK remains almost constant throughout training.

This bring me to my first goal for the project: to visualize how the NTK evolves thoughout 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 least-squares model considered in [4]:

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

I plan to use TensorFlow, a deep learning library for Python, to construct and optimize the model with weight initialization $\alpha \boldsymbol{w}_0$. 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 nth iteration of the training procedure, I will evaluate the NTK at the training points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. 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 $\boldsymbol{w}$ evolve throughout training. From our previous discussion of the kernel regime, we know that as $\alpha \to \infty$, the gradient flow of $L(f(\boldsymbol{w}))$ approaches that of $L(\bar{f}(\boldsymbol{w}))$. Accordingly, we can observe the ℓ_2 distance between the weights of the model, $\boldsymbol{w}(t)$, and those of the affine model (1), $\bar{\boldsymbol{w}}(t)$, for various values of α during training. We will do so using a deep learning model implemented in Python as described in the previous paragraph.

Moving away from the mathematics motivating the kernel regime, we will look at the generalizability of neural networks under the kernel and rich regimes. From the prior section, we know that for a fixed number

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

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 demonstrate that this observation holds even for more complex models, such as fully-connected ReLU networks with multiple hidden layers or CNNs. Once again, we will build and train each model in Python with TensorFlow, and we will subsequently evaluate the test error once the model has been trained to approximately zero training loss. More important than exhibiting the transition between the rich and kernel regimes, we will find the specific initialization scale necessary to train in the rich regime. In doing so, we will consider both $\boldsymbol{w}(0) = \alpha \boldsymbol{w}_0$ and $\boldsymbol{w}(0)_i \overset{i.i.d.}{\sim} \mathcal{W}$.

Should time permit, I would also like to examine the role of network depth on the transition between the kernel and rich regimes. As proven in [1] and discussed in [4], for increasing degree of homogeneity of the model f, the transition between the kernel and rich regimes becomes much sharper. That is, for appropriate network f and loss L, then as the depth of the network increases, the α necessary to train in the rich regime also increases. To empirically verify this result, we would train models of varying depth D, each for a range of initialization scales α; upon training each network, we would then measure the test error. By plotting the test error versus initialization scale α for each network depth D, we could then understand the effect of depth on the transition between the kernel and rich regimes.

2.2 Theoretical Study of Convergence

3 Deliverables

1. *At least* one deep learning model implemented in Python with TensorFlow
2. Python visualization of the neural tangent kernel $K_{\boldsymbol{w}(t)}$ while training in the rich and kernel regimes
3. Plot of the ℓ_2 distance between the weights of the original model, $\boldsymbol{w}(t)$, and those of the affine model (1), $\bar{\boldsymbol{w}}(t)$, throughout training with gradient descent for various α
4. Graph of test error versus initialization scale for a ReLU network with k hidden layers
5. Analysis of the initialization scale necessary to train in the rich regime for a ReLU network with k hidden layers

4 Tentative Timeline

Literature review, discussion with Professor Zhou	January 25 – February 6
Code first model in Python with Tensorflow	January 30 – February 20
Visualize the NTK during training for various α	February 20 – March 6
Implement and train affine model (1)	March 6 – March 18
Write code for, train ReLU neural network	March 28 – April 10
Visualize test loss versus initialization scale	April 10 – April 24
Prepare final report, poster	April 17 – May 5

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] L. Chizat, E. Oyallon, and F. Bach, *On lazy training in differentiable programming*, arXiv preprint arXiv:1812.07956, (2018).

[3] A. Jacot, F. Gabriel, and C. Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, arXiv preprint arXiv:1806.07572, (2018).

[4] 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.