UNIVERSITY OF CALIFORNIA

Los Angeles

Asymptotics of Learning in Neural Networks

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Melikasadat Emami

2022

ABSTRACT OF THE DISSERTATION

Asymptotics of Learning in Neural Networks

by

Melikasadat Emami

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2022

Professor Alyson K. Fletcher, Chair

Modern machine learning models, particularly those used in deep networks, are characterized by massive numbers of parameters trained on large datasets. While these large-scale learning methods have had tremendous practical successes, developing theoretical means that can rigorously explain when and why these models work has been an outstanding issue in the field. This dissertation provides a theoretical basis for the understanding of learning dynamics and generalization in high-dimensional regimes. It brings together two important tools that offer the potential for a rigorous analytic understanding of modern problems: statistics of high-dimensional random systems and neural tangent kernels. These frameworks enable the precise characterization of complex phenomena in various machine learning problems. In particular, these tools can overcome the non-convex nature of the loss function and non-linearities in the estimation process. The results shed light on the asymptotics of learning for two popular neural network models in high dimensions: Generalized Linear Models (GLMs) and Recurrent Neural Networks (RNNs).

We characterize the generalization error for Generalized Linear Models (GLMs) using a framework called Multi-Layer Vector Approximate Message Passing (ML-VAMP). This framework is a recently developed and powerful methodology for the analytical understanding of estimation problems. It allows us to analyze the effect of essential design choices, such as the degree of over-parameterization, loss function, and regularization, as well as initialization,

feature correlation, and a train/test distributional mismatch.

Next, we investigate the restrictiveness of a class of Recurrent Neural Networks (RNNs) with unitary weight matrices. Training RNNs suffers from the so-called vanishing/exploding gradient problem. The unitary RNN is a simple approach to mitigate this problem by imposing a unitary constraint on these networks. We theoretically show that for RNNs with ReLU activations, there is no loss in the expressiveness of the model from imposing the unitary constraint.

Finally, we explore the learning dynamics of RNNs trained under gradient descent using the recently-developed kernel regime analysis. Our results show that linear RNNs learned from random initialization are functionally equivalent to a certain weighted 1D-convolutional network. Importantly, the weightings in the equivalent model cause an implicit bias to elements with smaller time lags in the convolution and hence shorter memory. Interestingly, the degree of this bias depends on the variance of the transition matrix at initialization.

The dissertation of Melikasadat Emami is approved.

Lieven Vandenberghe

Sundeep Rangan

Jonathan Kao

Lin Yang

Alyson K. Fletcher, Committee Chair

University of California, Los Angeles

2022

To my beloved parents,

Shohreh and Hamed

Contents

List of Figures

List of Tables

Acknowledgements

This dissertation would not have been possible without the support of my advisors, colleagues, family, and friends.

First of all, I would like to thank my advisor, Professor Alyson K. Fletcher, for providing me the opportunity to pursue my graduate studies at UCLA and for her endless guidance, help, and support throughout my MS and PhD years. I had the opportunity to expand the breadth and depth of my knowledge as her student, and her advice and unique perspective helped me learn and grow as an independent researcher. Her trust made me confident in my abilities, and her patience and kindness have always been inspiring. I am forever grateful to her.

Words cannot express my gratitude and appreciation to Professor Sundeep Rangan for guiding me during these years. His extensive knowledge and intuition still keep me in awe and his technical insights played a significant role in my training. He taught me how to look at the big picture in every problem and ask fundamental questions. I was very fortunate to be able to work under his supervision and learn from him during my PhD years.

I would also like to thank my committee members, Professor Lieven Vandenberghe, who provided me with a solid base and profound understanding of the fundamentals of optimization; Professor Jonathan Kao, who taught me a great deal of my knowledge in deep learning; and Professor Lin Yang for his insightful comments and invaluable suggestions.

I want to express my gratitude to my first friends and then colleagues Mojtaba Sahraee and Parthe Pandit. We share many great memories in all these years and I am very lucky to have their true friendship and great support. They have always been very patient with me, and I learned a great deal by being in their presence.

Finally, my journey would not have been possible without the support of my family. My very special thanks go to my husband and best friend, Mahdi, for being an unceasing source of love, kindness, and support and for always believing in me, even when I did not believe in myself. I cannot be more grateful to my parents, Shohreh and Hamed, for giving me their unconditional love, reminding me to always pursue my dreams, and supporting me from thousands of miles away. I am incredibly thankful to my little sister, Nika, for being a constant source of inspiration, and filling my heart with joy and my life with happiness all these years.

Vita

Education

University of California, Los Angeles — Sep 2016 - June 2018

Master of Science in Electrical and Computer Engineering

University of Tehran — Sep 2012 - June 2016

Bachelor of Science in Electrical and Computer Engineering

Internships

Research Intern, Microsoft — March - June 2021

Awards

Caswell Grave Scholarship — 2017

Faculty of Engineering Award, University of Tehran — 2015, 2016

Chapter 1

Introduction

Over the past decade, models based on neural networks have become commonplace in virtually all machine learning applications. As collecting data has become cheaper, the rapid evolution of large-scale data analysis techniques using machine learning methodology has revolutionized the field in several application domains, including speech recognition, machine vision, and natural language processing. The growth of practical concepts and the success of deep learning have created a substantial gap between our theoretical understanding of modern machine learning models and empirical advances. This gap results in limited principles to lead the design of robust models, such as selecting the model architecture and parameter tuning. Providing theoretical frameworks for analyzing neural networks is critical to establish such principles.

A key feature of contemporary machine learning problems is their massive scale, both in terms of the number of parameters as well as the size of datasets for training them. Over-parameterized neural networks, in particular, have attracted growing attention, mainly because they generalize well in practice. However, due to the enormous complexity of these models, their theoretical analyses are primarily intractable. This task is even made harder by the non-convexity of the underlying learning problems.

From a theoretical standpoint, there are many questions related to success of neural

networks and high-dimensional models. This dissertation focuses on understanding the asymptotics of learning for two popular neural network models, namely, Generalized Linear Models (GLMs) and Recurrent Neural Networks (RNNs) and gives partial answers to the following aspects of these models:

- **Generalization:** The generalization of machine learning models is their ability to perform on previously unseen data. Recent studies have shown that contrary to the traditional statistical bias/variance trade-off, i.e., the U-shaped curve, high-dimensional neural networks are able to generalize well with consistently decreasing test error [17]. We provide an analysis framework to precisely characterize the asymptotic generalization error for the Generalized Linear model (GLM) class. Our framework enables studying the effect of over-parameterization and non-linearity as well as choices of loss functions and regularization and other design choices.

- **Expressive power:** Given a function class, expressive power is about understanding what functions can be realized or approximated by the functions in that class. We focus on a restricted type of Recurrent Neural Networks with unitary transition weight matrices that are appealing for optimization purposes to mitigate the vanishing and exploding gradients problem and evaluate their expressive power compared to vanilla RNNs. We rigorously show that for RNNs with ReLU activations, there is no loss in the expressiveness of the model when imposing the unitary constraint.

- **Implicit bias:** In principle, general-purpose machine learning models such as neural networks do not require an explicit design of features but instead, rely on implicit *meaningful* representations of the data. A lack of understanding of the potential implicit biases that these representations bring to the prediction of the model is of significant importance. We explore the learning dynamics of RNNs trained under gradient descent and provide precise reasoning for their implicit bias behavior toward short-term memory. We further show how the degree of this bias connects to the RNN initialization.

1.1 The Statistical Learning Framework

In a general supervised learning setting, given a training set $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ that is independently and identically distributed (i.i.d.) with some unknown distribution $\mathcal{D}$ over $\mathcal{X}$ and $\mathcal{Y}$, we look for a function $h : \mathcal{X} \to \mathcal{Y}$ (also called a hypothesis) such that, given a real-valued and non-negative loss function $\mathcal{L}$, it minimizes the expected risk

$$\underset{h \in \mathcal{H}}{\operatorname{argmin}} R(h) := \mathbb{E}_{(\boldsymbol{x},\mathbf{y})\sim\mathcal{D}} \left[\mathcal{L}(h(\mathbf{x}), \mathbf{y})\right] \tag{1.1}$$

for some hypothesis class $\mathcal{H}$. Common loss functions used in this setting include the squared loss: $\mathcal{L}(\widehat{\mathbf{y}}, \mathbf{y}) = \|\widehat{\mathbf{y}} - \mathbf{y}\|^2$ for regression problems and the 0-1 loss: $\boldsymbol{I}(\widehat{\mathbf{y}} \neq \mathbf{y})$, for classification problems where $\boldsymbol{I}$ is the indicator function.

Since the data distribution $\mathcal{D}$ is unknown to the learning algorithm, one is interested in computing an approximation for the expected risk by averaging the loss function over the training set, formally known as the empirical risk,

$$R_{\text{emp}}(h) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(h(\mathbf{x}_i), \mathbf{y}_i). \tag{1.2}$$

The empirical risk minimization problem, therefore, chooses a hypothesis $\widehat{h}$ that minimizes the following risk:

$$\widehat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} R_{\text{emp}}(h). \tag{1.3}$$

In a parametric setting, the hypothesis class is assumed to be parameterized by a vector $\boldsymbol{\theta} \in \mathbb{R}^p$ and the minimization (1.3) is taken over $\boldsymbol{\theta}$. Considering some prior knowledge on the parameters, a regularization term $Reg(.)$ can be added to the optimization problem (1.3) as well, i.e.

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(h(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i) + \lambda Reg(\boldsymbol{\theta}), \tag{1.4}$$

with the regularization coefficient λ. The regularization function encourages certain structures

for the solution. Common functions include ridge regularization: $\|\boldsymbol{\theta}\|_2^2 = \sum_{i=1}^p |\theta_i|^2$, LASSO: $\|\boldsymbol{\theta}\|_1 = \sum_{i=1}^p |\theta_i|$, and hard sparsity: $\|\boldsymbol{\theta}\|_0 = \sum_{i=1}^p \mathbf{1}(\theta_i \neq 0)$ among others.

Maximum Likelihood Estimator In the parametric setting, minimizing the empirical risk (1.2) can also traditionally be viewed as finding the maximum likelihood estimator (MLE) for targets $\mathbf{y}$, given inputs $\mathbf{x}$. In this case, the loss function $\mathcal{L}$ is the negative log-likelihood function. This is equivalent to maximizing the likelihood of observations $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ given a parametric family of conditional probability distributions $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ indexed by $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\theta}}_{\text{MLE}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i, \mathbf{y}_i; \boldsymbol{\theta}). \tag{1.5}$$

The main attraction of MLE is that classically, it has shown to be asymptotically the best estimator in terms of the rate of convergence if $p = O(\log(N))$. MLE is a statistically consistent and asymptotically normal estimator, i.e. $\sqrt{N}(\widehat{\boldsymbol{\theta}}_{\text{MLE}}) - \boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(0, \boldsymbol{I}(\boldsymbol{\theta}^*)^{-1})$, where $\boldsymbol{I}(\boldsymbol{\theta}^*)^{-1}$ is the Fisher Information matrix.

Similar to (1.4), a regularization function can be added to (1.5) to encourage certain solutions. The Bayesian interpretation of the regularization function is that it is equivalent to the negative log of the prior distribution for the parameter $\boldsymbol{\theta}$. In this case, the corresponding optimization problem is a maximum a posteriori (MAP) estimator, which chooses the mode of the posterior distribution. This estimator has the leverage of adding information given by the prior that does not exist in the training data. This will result in a decrease in the variance error and an increase in the bias error of the MAP estimator compared to the MLE.

Note that minimizing the empirical risk over the training set does not guarantee a low expected error over the data distribution. To avoid predictors that memorize the training set, we are interested in the performance of our predictor over previously unseen data, commonly known as the *generalization error*. The exact definition of the generalization error may vary in different works [17, 59, 90, 113]. We will precisely define the quantity of our interest later.

1.2 High-Dimensional Regimes

Classically, the estimators such as (1.4) are studied in a setting where the number of samples N goes to infinity, but the number of parameters, p, remains fixed. This setting is commonly known as the *large sample limit*. In this limit, the asymptotic properties of estimators can be analyzed using tools such as the law of large numbers or the Central Limit Theorem (CLT). Although these results are only theoretically valid for the case where $N \to \infty$, they might still be approximately correct for finite sample sizes. In modern problems, however, the number of samples is usually comparable to the number of parameters or even less. Analyzing the properties of estimators in such finite settings is much more complicated and requires more advanced tools, such as concentration of measures.

The interest in over-parameterized settings has attracted a lot of attention to a regime where the number of parameters $p \to \infty$. Neural nets in this regime enjoy certain theoretical properties such as easier optimization to global minimum [41], Gaussian process behavior at initialization [70], and equivalence in training to kernel methods using gradient descent [63, 72, 129]. The tools to analyze neural nets in this regime include mean-field theory and kernel regime-based analysis among others.

Recently, a newly developed regime considers the case where both the number of parameters and the number of samples go to infinity, but their respective ratio is bounded, i.e., $N, p \to \infty,\ p/N \to \beta \in (0, \infty)$. This regime is called the *proportional asymptotics regime* and is considered more realistic for modern machine learning tasks. The convergent behavior that emerges by taking the limit in this setting – such as convergence of the spectrum of large random matrices to known distributions– makes the analysis of modern problems much more manageable. Fortunately, many results in this regime are approximately correct, even in finite sample cases. The tools to analyze our models in this setting include many results from random matrix theory and the Approximate Message Passing framework.

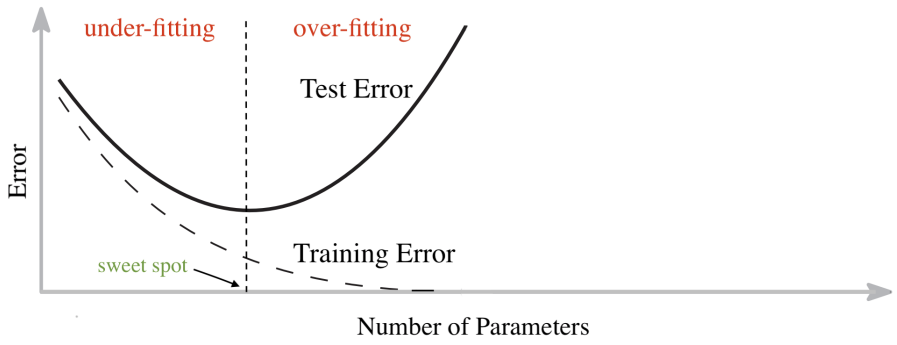

Figure 1.1: The classic U-shaped curve representing the bias-variance trade-off

1.3 Bias-Variance Trade-off and the Double Descent Curve

As mentioned earlier, machine learning aims to find a predictor $h \in \mathcal{H}$ that performs well on previously unseen data. Conventional wisdom suggests that the balance between under-fitting and over-fitting training data can be achieved either explicitly by controlling the function class's capacity or implicitly by using some form of regularization, such as early stopping. A small capacity function class results in having a large empirical risk, while a very large $\mathcal{H}$ results in memorizing the patterns in the training set and hence poor performance on new data, i.e., over-fitting. Finding this balance, therefore, translates into finding a sweet spot in the classical U-shaped bias-variance curve, Figure 1.1.

Recent empirical evidence on modern machine learning problems such as large neural networks has shown that despite the high capacity of their function class and perfect fit of the training data, these models perform very well on new data [2, 17, 55, 91]. Such observations summarized into a new curve called the "*double descent*" curve, Figure 1.2, introduced in [17]. If the function class capacity is below a certain threshold, the classic U-shaped curve is observed. However, as this capacity passes beyond the *interpolation threshold* (zero loss training), the test error decreases even below the sweet spot in the classical regime. This new regime is called the *modern regime*. Note that in Figure 1.2, the number of parameters is considered as the measure of capacity of the model. Of course, there exist more elaborate measures for capacity as well.

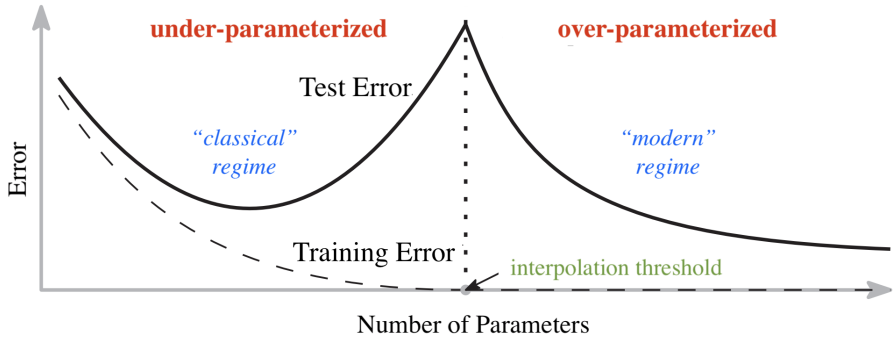

Figure 1.2: The double descent curve introduced in [17].

Explaining such behavior in high-capacity functions, such as neural nets with a very large number of parameters, has become an open problem for researchers in recent years. The idea lies behind the implicit biases that exist for the specific problems at hand. Such biases could, for instance, come from the smoothness of the target function measured by certain function space norms [17]. Nevertheless, theoretical explaining of the second descent is challenging, and recent works have studied this for various linear models for certain classification and regression problem instances [36, 59, 82, 88]. In Chapter 3, we rigorously and analytically explain the double descent curve for the class of Generalized Linear Models (GLMs).

1.4 Learning Dynamics of Wide Neural Networks under Gradient Descent

From an optimization standpoint, the loss landscape of neural networks is highly non-convex. This non-convexity implies the existence of a large number of saddle points as dominant critical points, as well as many local minima at low loss values [31, 35]. Characterizing the gradient-based learning dynamics in such settings, therefore, becomes difficult. Similar to the techniques we mentioned earlier, considering these models in their extreme limits makes analyzing their learning dynamics much easier and such limit for neural networks could be the number of hidden units.

At initialization, due to the randomness of the parameters, the network function is a random function. Under the infinite width limit, the output of the network at initialization becomes a Gaussian process (GP) [70]. Other than zero training error and good generalization, the interest in exploring the dynamics of neural networks at this limit relies on some previous results that connect infinite width neural networks to high-dimensional asymptotics of kernel ridge regression. The leading result in this area is presented in [63] stating that at infinite width limit, a fully connected neural network is governed by by a linear model obtained from the first-order Taylor expansion of the network at its initialization. In this network, the full batch gradient descent in parameter space corresponds to kernel gradient descent in function space with the so-called Neural Tangent Kernel (NTK).

This "kernel regime" analysis also allows convergence proofs to zero error solutions in over-parameterized settings. These solutions correspond to the minimum norm solution for the appropriate RKHS and hence can have the inductive biases present in the corresponding RKHS [4, 5, 7, 29, 74, 83, 132]. Observe that this kernel depends on the architecture and the initialization of the network. Of course, it is well-known that nonlinear networks, in general, can offer significant advantages over simple linear models. Our result does not contradict these observations. Instead, it highlights the importance of distributional assumptions in the high-dimensional regimes and that more complex models are needed to understand real problems.

1.5 Organization of the Dissertation

The rest of this dissertation is organized as follows: In Chapter 2, we provide some preliminary results and background on the framework analysis used throughout the subsequent chapters, including the Approximate Message Passing framework and kernel regime-based analysis. Next, Chapter 3 provides a framework to characterize the asymptotic generalization error for the class of generalized linear models (GLMs) in a certain random high dimensional regime.

Our key tool in this chapter is the Approximate Message Passing (AMP) framework. The exact predictions given by the AMP algorithms estimators enable us to calculate the generalization error and find its relation to the problem's key parameters, including the sampling ratio, the regularizer, the output function, and the distributions of the true weights. We are also able to capture the distributional mismatch between the training and test data. The results in this chapter appeared in [47]. Chapter 4 studies a constrained version of a recurrent neural network (RNN) with a unitary weight transition matrix. We show that unitary RNNs are at least as powerful as contractive RNNs in modeling input-output mappings if enough hidden units are used. The results in this chapter appeared in [46]. Finally, Chapter 5 provides a rigorous explanation for the empirical observation of the short-term bias of recurrent neural networks. Using kernel regime analysis, we show why this bias exists and how the degree of it is related to the initialization of the RNNs weight transition matrix. The results in this chapter appeared in [48].

Chapter 2

Background and Preliminaries

In this Chapter, we first give a brief overview of the general notation used throughout this dissertation. Next, we review some technical definitions required for the subsequent chapters. We then introduce the main frameworks for analyzing our problems of interest. These frameworks include the Approximate Message Passing and the kernel regime, including the Neural Tangent Kernel for neural networks. Finally, we provide some useful lemmas.

2.1 Notation

We have specified the dimensions of any vectors or matrices specifically in each chapter; however, we mention the general rule here for convenience. Vectors are denoted by boldface lowercase letters $\mathbf{x}$ and x_i is the i^{th} coordinate of $\mathbf{x}$. A subvector of $\mathbf{x}$ is denoted as $\boldsymbol{x}_j$ for $j \subseteq \{1, \ldots, dim(\mathbf{x})\}$. Matrices are denoted by boldface uppercase letters $\mathbf{X}$ and X_{ij} is the element in the i^{th} row and the j^{th} column of $\mathbf{X}$. Non-bold face characters are mostly used to indicate random variables and scalars. These rules hold in general unless stated otherwise.

2.2 Technical Definitions and Useful Results

Pseudo-Lipschitz Continuity For a given $p \geq 1$, a function $f : \mathbb{R}^d \to \mathbb{R}^m$ is called pseudo-Lipschitz of order p, denoted by PL(p), if

$$\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| \leq C\|\mathbf{x}_1 - \mathbf{x}_2\| \left(1 + \|\mathbf{x}_1\|^{p-1} + \|\mathbf{x}_2\|^{p-1}\right) \tag{2.1}$$

for some constant $C > 0$. This is a generalization of the standard definition of Lipshitiz continuity. A PL(1) function is Lipschitz with constant $3C$.

Uniform Lipschitz Continuity Let $f(\mathbf{x}, \boldsymbol{\theta})$ be a function on $\mathbf{x} \in \mathbb{R}^d$ and $\boldsymbol{\theta} \in \mathbb{R}^s$. We say that $f(\mathbf{x}, \boldsymbol{\theta})$ is *uniformly Lipschitz continuous* in $\mathbf{x}$ at $\boldsymbol{\theta} = \overline{\boldsymbol{\theta}}$ if there exists constants $L_1, L_2 \geq 0$ and an open neighborhood U of $\overline{\boldsymbol{\theta}}$ such that

$$\|f(\mathbf{x}_1, \boldsymbol{\theta}) - f(\mathbf{x}_2, \boldsymbol{\theta})\| \leq L_1\|\mathbf{x}_1 - \mathbf{x}_2\| \tag{2.2}$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ and $\boldsymbol{\theta} \in U$; and

$$\|f(\mathbf{x}, \boldsymbol{\theta}_1) - f(\mathbf{x}, \boldsymbol{\theta}_2)\| \leq L_2 \left(1 + \|\mathbf{x}\|\right) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \tag{2.3}$$

for all $\mathbf{x} \in \mathbb{R}^d$ and $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in U$.

Empirical Convergence of a Sequence Consider a sequence of vectors $\mathbf{x}(N) = \{\mathbf{x}_n(N)\}_{n=1}^N$ with $\mathbf{x}_n(N) \in \mathbb{R}^d$. So, each $\mathbf{x}(N)$ is a block vector with a total of Nd components. For a finite $p \geq 1$, we say that the vector sequence $\mathbf{x}(N)$ converges empirically with p-th order moments if there exists a random variable $X \in \mathbb{R}^d$ such that

(i) $\mathbb{E}\|X\|_p^p < \infty$; and

(ii) for any $f : \mathbb{R}^d \to \mathbb{R}$ that is pseudo-Lipschitz continuous of order p,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n(N)) = \mathbb{E}\left[f(X)\right]. \tag{2.4}$$

In this case, with some abuse of notation, we will write

$$\lim_{n\to\infty} \mathbf{x}_n \overset{PL(p)}{=} X, \tag{2.5}$$

where we have omitted the dependence on N in $\mathbf{x}_n(N)$. We note that the sequence $\{\mathbf{x}(N)\}$ can be random or deterministic. If it is random, we will require that for every pseudo-Lipschitz function $f(\cdot)$, the limit (2.4) holds almost surely. In particular, if $\mathbf{x}_n \sim X$ are i.i.d. and $\mathbb{E}\|X\|_p^p < \infty$, then $\mathbf{x}$ empirically converges to X with p^{th} order moments.

Weak convergence (or convergence in distribution) of random variables is equivalent to

$$\lim_{n\to\infty} \mathbb{E}f(X_n) = \mathbb{E}f(X), \quad \text{for all bounded functions } f. \tag{2.6}$$

It is shown in [16] that PL(p) convergence is equivalent to weak convergence plus convergence in p moment.

Wasserstein-2 Distance Let $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ be two distributions on some Euclidean space $\mathcal{X}$. The Wasserstein-2 distance between ν and μ is defined as

$$W_2(\boldsymbol{\nu}, \boldsymbol{\mu}) = \left(\inf_{\gamma\in\Gamma} \mathbb{E}\left\|X - X'\right\|_2^2 \right)^{\frac{1}{2}}, \tag{2.7}$$

where Γ is the set of all distributions with marginals consistent with $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$.

A sequence $\mathbf{x}_n$ converges PL(2) to X if and only if the empirical measure $\widehat{\mathbb{P}}_N = \frac{1}{N}\sum_{n=1}^{N} \delta(\mathbf{x} - \mathbf{x}_n)$ (where $\delta(\cdot)$ is the Dirac measure,) converges in Wasserstein-2 distance to

distribution of X [125], i.e.

$$\mathbf{x}_n \overset{PL(2)}{=} X \quad \iff \quad \lim_{n\to\infty} W_2(\widehat{\mathbb{P}}_N, \mathbb{P}_X) = 0. \tag{2.8}$$

For two zero mean Gaussian measure $\boldsymbol{\nu} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_1)$, $\boldsymbol{\mu} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_2)$ the Wasserstein-2 distance is given by [57]

$$W_2^2(\boldsymbol{\nu}, \boldsymbol{\mu}) = \mathrm{tr}(\boldsymbol{\Sigma}_1 - 2(\boldsymbol{\Sigma}_1^{1/2}\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{1/2})^{1/2} + \boldsymbol{\Sigma}_2). \tag{2.9}$$

Therefore, for zero mean Gaussian measures, convergence in covariance, implies convergence in Wasserstein-2 distance, and hence if the empirical covariance of a zero mean Gaussian sequence $\mathbf{x}_n$ converges to some covaraince matrix $\boldsymbol{\Sigma}$, then using (2.8) $\mathbf{x}_n \overset{PL(2)}{=} X$ where $X \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$.

2.2.1 Marchenko-Pastur Distribution

Consider the matrix $\mathbf{H} \in \mathbb{R}^{N\times p}$ such that $H_{ij} \sim \mathcal{N}(0, \frac{1}{N})$. As $N, p \to \infty$ with $\frac{p}{N} \to \beta$, the positive eigenvalues of $\mathbf{H}^\mathsf{T}\mathbf{H}$ have an empirical distribution which converges to the following density [122]:

$$\mu_\beta(x) = \frac{\sqrt{(b_\beta - x)_+(x - a_\beta)_+}}{2\pi\beta x} \tag{2.10}$$

where $a_\beta = (1 - \sqrt{\beta})^2$, $b_\beta := (1 + \sqrt{\beta})^2$. Similarly, the positive eigenvalues of $\mathbf{H}\mathbf{H}^\mathsf{T}$ have an empirical distribution converging to the density $\beta\mu_\beta$. We note the following integral which is useful in our analysis:

$$\begin{aligned} G_0 :&= \lim_{z\to 0^-} \mathbb{E}\frac{1}{S_{\mathrm{mp}}^2 - z}\mathbb{1}_{\{S_{\mathrm{mp}}>0\}} \\ &= \lim_{z\to 0^-} \int_{a_\beta}^{b_\beta} \frac{1}{x/\beta - z}\mu_\beta(x)dx = \frac{\beta}{|\beta - 1|}. \end{aligned} \tag{2.11}$$

More generally, the Stieltjes transform of the density is given by:

$$G_{\text{mp}}(z) = \mathbb{E}\frac{1}{S_{\text{mp}}^2 - z}\mathbb{1}_{\{S_{\text{mp}}>0\}} = \int_{a_\beta}^{b_\beta} \frac{1}{x/\beta - z}\mu_\beta(x)dx \tag{2.12}$$

2.3 Analysis Framework: Approximate Message Passing

The Approximate Message Passing framework is a powerful methodology that provides computationally efficient algorithms for high-dimensional estimation problems. It was originally introduced for solving linear inverse problems [37–39] and has later been extended for inference and learning in a wide variety of tasks, including inference in multi-layer networks, bilinear problems, and models with structured priors [16,50,51,53,60,79,97]. The key property of these algorithms is that they give optimality guarantees in cases where other known approaches do not. Especially, their iterations weakly converge to a set of deterministic recursive equations called the *State Evolution* that provide exact predictions on the performance of the estimates of the algorithm in the large system limit.

The general AMP framework focuses on estimating an unknown signal given a set of measurements. Consider estimating a vector $\mathbf{w} \in \mathbb{R}^p$ from measurements $\mathbf{y} \in \mathbb{R}^N$ generated through a function f with known parameters $\mathbf{X}$ and unknown noise $\boldsymbol{\xi}$, i.e.,

$$\mathbf{y} = f(\mathbf{w}, \mathbf{X}, \boldsymbol{\xi}). \tag{2.13}$$

In signal processing applications, $\mathbf{w}$ is the unknown signal, such as an image, and $\mathbf{X}$ is the parameters of the measurement process $f(.)$ (e.g., blurring). In supervised learning problems, $\mathbf{X}$ contains the training features, and $\mathbf{y}$ is the training label. $\dot{f()}$ then becomes the model that connects the features to the labels with unknown parameters $\mathbf{w}$.

We are usually interested in evaluating the performance of the estimator $\widehat{\mathbf{w}}$ for $\mathbf{w}$. For regression problems, this performance metric may be the Mean Squared Error (MSE), $\mathcal{E}_p(\mathbf{X}) := \frac{1}{p}\mathbb{E}[\|\mathbf{w} - \widehat{\mathbf{w}}\|^2]$, whereas for classification problems this may be the test error rate.

This metric is a function of the dimension as well as the parameters $\mathbf{X}$ and noise.

2.3.1 Large System Limit

We follow the Large System Limit (LSL) analysis in the Approximate Message Passing framework. Consider a sequence of problems indexed by the number of measurements N. For each N, we suppose that the number of features $p = p(N)$ grows linearly with N, i.e.,

$$\lim_{N\to\infty} \frac{p(N)}{N} \to \beta \tag{2.14}$$

for some constant $\beta \in (0, \infty)$. To evaluate the MSE in LSL, we assume the true $\mathbf{w}$ is drawn from some distribution $p(\mathbf{w})$. We further assume the known parameters $\mathbf{X}$ are the realization of some random matrix with certain statistics. We then compute the limit $\lim_{p\to\infty} \mathcal{E}_p(\mathbf{X})$. By concentration phenomena in high dimensions, this limit is only a function of the statistics of $\mathbf{X}$, the true $\mathbf{w}$, and noise as well as the ratio β and the estimator used for $\widehat{\mathbf{w}}$.

This analysis can show the performance of a wide variety of estimators (both Bayes-optimal and certain classes of sub-optimal estimators) as a function of the statistics mentioned above in the high-dimensional limit.

2.3.2 AMP for Linear Inverse Problems

Consider the linear inverse problem for estimation $\mathbf{w} \in \mathbb{R}^p$, given the measurements $\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\xi}$ with a known linear transformation $\mathbf{X}$ and white noise $\boldsymbol{\xi}$. Analyzing this problem without any prior information on $\mathbf{x}$ is straightforward. The least squares estimator (LS) $\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$ is typically used in this case. This problem becomes interesting when there exists some prior information on $\mathbf{w}$. In this case a regularized version of the problem is used to encourage the structure in $\mathbf{w}$, i.e.,

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \mathcal{R}(\mathbf{w}). \tag{2.15}$$

For example, when $\mathcal{R}(\mathbf{w}) = \lambda \|\mathbf{w}\|_1$ the problem is known as LASSO [121]. The first appearance of AMP was to solve for LASSO [39].

Given some probabilistic model for $\mathbf{w}$, several questions arise regarding the performance of the LS estimator in (2.15), and besides Gaussian settings, rigorous analysis is quite challenging. In the case of compressed sensing, for example, scaling laws were introduced to connect the Mean Squared Error (MSE) on the estimates to the number of measurements, properties of $\mathbf{X}$ such as restricted isometry constant, and the sparsity of the true $\mathbf{w}$ [26, 27]. These bounds were inexact, and the proof techniques were specific to the true $\mathbf{w}$ structure and to the form of the regularizer $\mathcal{R}(.)$.

AMP methods, on the other hand, enable precise and rigorous analysis of the properties of the estimates $\widehat{\mathbf{w}}$ in the large system limit (LSL). The AMP recursion for the problem (2.15) can be derived using an approximation of loopy belief propagation in graphical models [85,87,103,112]. Another important feature of AMP algorithms is that they are often an order of magnitude faster than other algorithms used for solving such problems, e.g. FISTA [87]. The recursion for the LS problem (2.15) is given by:

$$\mathbf{v}^k = \mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}^k + \tfrac{N}{p}\big\langle f'(\mathbf{r}^{k-1})\big\rangle \mathbf{v}^{k-1}, \tag{2.16}$$

$$\mathbf{r}^k = \widehat{\mathbf{w}}^k + \mathbf{X}^\mathsf{T}\mathbf{v}^k, \tag{2.17}$$

$$\widehat{\mathbf{w}}^{k+1} = f(\mathbf{r}^k),$$

where k is the iteration number, $f(.)$ is a Lipschitz denoising function, $f'(.)$ is the derivative, and $\langle . \rangle$ denotes the component-wise average.

This algorithm was first introduced by [37] and later analyzed in [16]. Different choices of denoiser functions $f(.)$ corresponds to different type of estimators. It is easy to show that by considering

$$f(\mathbf{r}) = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{R}(\mathbf{w}) + \frac{\gamma}{2}\|\mathbf{w} - \mathbf{r}\|_2^2, \tag{2.18}$$

this algorithm finds the solution to (2.15). Note that with this denoiser and by removing the

second term in (2.16) – commonly known as the *Onsager correction* term – the algorithm is equivalent to the proximal gradient descent for problem (2.15). We can also consider the maximum a-posteriori estimator for $\mathbf{w}$ by taking $\mathcal{R}(\mathbf{w}) = -\log(p(\mathbf{w}))$ or minimum mean squared error (MMSE) estimator by taking the mean of the posterior distribution $p(\mathbf{w}|\mathbf{r}) = \frac{1}{Z}\left[\mathcal{R}(\mathbf{w}) + \frac{\gamma}{2}\left\|\mathbf{w}-\mathbf{r}\right\|_2^2\right]$ and an appropriate choice of γ.

Suppose that $\mathbf{X}$ is i.i.d. sub-Gaussian with $\mathbb{E}[X_{ij}] = 0$ and $\mathbb{E}[X_{ij}^2] = \frac{1}{p}$. Further, suppose that $\mathbf{w}_{\text{true}}$ is i.i.d. with some generic prior $p(w^0)$, the noise $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, and the denoiser $f(.)$ is a generic Lipschitz, separable function $[f(w)]_i = \bar{f}(w_i)\ \forall i$. It is shown in [14, 16] that as $N \to \infty$, the $\mathbf{r}^k$ in (2.16) behave as

$$\mathbf{r}^k = \mathbf{w}_{\text{true}} + \mathcal{N}(\mathbf{0}, \tau^k\mathbf{I}), \tag{2.19}$$

for some variance τ^k. It is perceivable now why $f(.)$ is called as a denoiser; essentially, it recovers $\mathbf{w}_{\text{true}}$ from the noise corrupted signal $\mathbf{r}$. The variance τ^k is given by a scalar state evolution (SE) recursion:

$$\tau^{k+1} = \sigma^2 + \delta^{-1}\mathcal{E}(\tau^k), \quad \mathcal{E}(\tau^k) := \mathbb{E}\Big[\big[\bar{f}\big(w^0 + \mathcal{N}(0,\tau^k)\big) - w^0\big]^2 \ \Big|\ w^0 \sim p(w^0)\Big]. \tag{2.20}$$

The exact value of the large-system MSE, $\lim_{N\to\infty} \frac{1}{N}\ \mathbb{E}[\|\widehat{\mathbf{w}}^k - \mathbf{w}_{\text{true}}\|^2] = \mathcal{E}(\tau^k)$, for each iteration can be calculated using the SE. Note that although SE predictions are exact for an infinite-dimensional system, it has been shown that they still hold approximately true for large enough finite systems.

In general, the fixed points of SE provide a way of computing the asymptotic performance of the algorithm. In certain cases, we can show that the algorithm will achieve the Bayes optimal solution [11, 53, 69, 104, 107]. This implies that AMP can give a computationally simple method even for non-convex $\mathcal{R}(.)$. Results obtained from AMP algorithms are exact but only hold in the asymptotic regime and can describe the behavior of our system of interest using distributions on low dimensional random variables.

Algorithm 1 Vector Approximate Message Passing (VAMP)- MMSE form

Require: Estimators $\mathbf{g}^+$ and $\mathbf{g}^-$ and number of iterations $N_{\rm it}$

Set $\mathbf{r}_0^- = \mathbf{0} \in \mathbb{R}^p$ and initialize $\gamma_0^- > 0$.
for $k = 0, 1, \ldots, N_{\rm it} - 1$ **do**
  $\widehat{\mathbf{w}}_k^+ = \mathbf{g}^+(\mathbf{r}_k^-, \gamma_k^-)$
  $\lambda_k^+ = \gamma_k^- / \left\langle \frac{\partial \mathbf{g}^+}{\partial \mathbf{r}_k^-}(\mathbf{r}_k^-, \gamma_k^-) \right\rangle$,
  $\gamma_k^+ = \lambda_k^+ - \gamma_k^-$
  $\mathbf{r}_k^+ = (\lambda_k^+ \widehat{\mathbf{w}}_k^+ - \gamma_k^- \mathbf{r}_k^-)/\gamma_k^+$
  $\widehat{\mathbf{w}}_k^- = \mathbf{g}^-(\mathbf{r}_k^+, \gamma_k^+)$
  $\lambda_k^- = \gamma_k^+ / \left\langle \frac{\partial \mathbf{g}^-}{\partial \mathbf{r}_k^+}(\mathbf{r}_k^+, \gamma_k^+) \right\rangle$,
  $\gamma_k^- = \lambda_k^- - \gamma_k^+$
  $\mathbf{r}_k^- = (\lambda_k^- \widehat{\mathbf{w}}_k^- - \gamma_k^+ \mathbf{r}_k^+)/\gamma_k^-$
end for

2.3.3 Vector Approximate Message Passing (VAMP)

One of the main limitations of the original AMP algorithm is that it requires the matrix $\mathbf{X}$ to have i.i.d sub-Gaussian entries. For a generic $\mathbf{X}$, the algorithm is fragile and the iterations can diverge. Vector Approximate Message Passing (VAMP), introduced in [105], is an algorithm similar to AMP that applies to a much larger class of design matrices and is closely related to expectation propagation (EP) [86, 120] and expectation consistent approximate inference (EC) [49, 96]. If the design matrix $\mathbf{X}$ has the singular value decomposition of $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}}$, the VAMP algorithm converges as long as the singular values are bounded and the matrix $\mathbf{V}$ has a rotationally invariant distribution, i.e., it is uniformly distributed over the space of orthogonal matrices.

The VAMP algorithm operates in a set of iterations given in Algorithm 1. For the observation model $\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\xi}$, $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, it requires two denoisers: $\mathbf{g}^+$ which is similar to the denoiser f in the original AMP algorithm, and $\mathbf{g}^-$ which is a linear MMSE denoiser under a Gaussian prior given by

$$\mathbf{g}^-(\mathbf{r}_k^+, \gamma_k^+) := (\sigma^2\mathbf{X}^{\mathsf{T}}\mathbf{X} + \gamma_k^+\mathbf{I})^{-1}(\sigma^2\mathbf{X}^{\mathsf{T}}\mathbf{y} + \gamma_k^+\mathbf{r}_k^+). \tag{2.21}$$

Similar to AMP, the performance of VAMP estimates at each iteration can also be character-

ized by random variables obtained from the state evolution equations. Specifically, in the proportional asymptotics regime, the following convergence holds:

$$(\mathbf{w}^0, \widehat{\mathbf{w}}_k^+, \mathbf{r}_k^-) \overset{PL(2)}{=} (W^0, \widehat{W}_k^+, R_k^-), \tag{2.22}$$

where $R_k^- = W^0 + \mathcal{N}(0, \tau_k^-)$ and $\widehat{W}_k^+ = g^+(R_k^-, \bar{\gamma}_k^-)$ for a separable $\mathbf{g}^+$. The terms $\bar{\gamma}_k^-, \tau_k^-$ are given by the SE equations. Details of VAMP state evolution equations are given in [106].

2.3.4 Multi-Layer Vector Approximate Message Passing (ML-VAMP)

The Multi-Layer Vector Approximate Message Passing (ML-VAMP) algorithm is a generalized version of the VAMP used for inference over multi layer networks. This algorithm is the main tool used in Chapter 3 to analyze the GLM learning problem and it was first introduced in the works [51, 99]. We give a brief summary of it in this section.

Consider the following L layer stochastic neural network:

$$\begin{aligned} \mathbf{p}_\ell &= \mathbf{W}_\ell \, \mathbf{z}_{\ell-1} + \mathbf{b}_\ell + \boldsymbol{\nu}_\ell \quad \ell = 1, \dots, L \\ \mathbf{z}_\ell &= \boldsymbol{\phi}_\ell(\mathbf{p}_{\ell-1}, \boldsymbol{\xi}_\ell) \quad \ell = 1, \dots, L \end{aligned} \tag{2.23}$$

where $\boldsymbol{\nu}_\ell, \boldsymbol{\xi}_\ell$ are noise vectors and $\boldsymbol{\phi}_\ell$ is a separable non-linear function. The inference problem over this network is an attempt to find the input $\mathbf{z}_0$ and the signals $\mathbf{z}_\ell$ and $\mathbf{p}_\ell$ for all internal layers given the output of the network $\mathbf{z}_L$. Note that for inference, we assume that the network parameters i.e., the weights $\mathbf{W}_\ell$ and biases $\mathbf{b}_\ell$ are known. Furthermore, we assume that the $\mathbf{z}_0$ input distribution is also known. This is different from the learning problem where the attempt is to find the network parameters. As mentioned earlier, the inference problem arises in many inverse problems such as compressed sensing. Other state-of-the-art problems include modeling deep generative priors for complex structured data such as images, videos and text.

The ML-VAMP algorithm works in a series of forward and backward iterations that flow information through the network. It generates the estimates of the hidden signals using the *denoiser* functions that similar to VAMP could be configured as MAP or MMSE estimators. It is important to note that for the network (2.23), finding the denoisers are computationally inexpensive. The key properties of ML-VAMP is as follows: (1) The fixed points of this algorithm correspond to the stationary points of the variational formulation of these estimators and (2) In the proportional asymptotics regime, similar to VAMP, the performance of the estimators can exactly be characterized by the state evolution equations.

We will look further into details of using ML-VAMP for analysing the GLM learning problem in Chapter 3. For more details about the algorithm and the SE please refer to [99].

2.4 Analysis Framework: Kernel Regime and Neural Tangent Kernel

In this section we provide a short overview of reproducing kernel Hilbert spaces, Gaussian regression, and neural tangent kernels.

2.4.1 Kernel Regression

In kernel regression, the estimator $\widehat{y}(\mathbf{x})$ is a function that belongs to a reproducing kernel Hilbert space (RKHS). A kernel $K : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$ that is an inner product in a possibly infinite dimensional space $\mathcal{H}$ called the *feature space*, i.e. $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ where $\phi : \mathbb{R}^p \to \mathcal{H}$ is called the feature map. With this feature map, the functions in the RKHS are of the form $f(\mathbf{x}) = \langle \phi(\mathbf{x}), \boldsymbol{\theta} \rangle_{L^2}$ which is a nonlinear function in $\mathbf{x}$ but linear in the parameters $\boldsymbol{\theta}$. Commonly used kernels in practice are usually of the form

$$K(\mathbf{x}_i, \mathbf{x}_j) = g\left(\frac{\|\mathbf{x}_i\|_2^2}{p}, \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{p}, \frac{\|\mathbf{x}_j\|_2^2}{p}\right) \tag{2.24}$$

which include inner product kernels as well as shift-invariant kernels. Kernels such as RBF kernels, polynomial kernels, as well as the neural tangent kernel are of this form.

In kernel methods, the estimator is often learned via a regularized ERM

$$\widehat{f}_{\text{ker}} = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(y_i, f(\mathbf{x}_i)\right) + \lambda \|f\|_{\mathcal{H}}^2, \tag{2.25}$$

where $\mathcal{L}$ is a loss function and $\|f\|_{\mathcal{H}} := \sqrt{\langle f, f \rangle_{\mathcal{H}}}$ is the RKHS norm. By writing $f(\mathbf{x}) = \langle \phi(\mathbf{x}), \boldsymbol{\theta} \rangle$ as a parametric function with parameters $\boldsymbol{\theta} \in \mathcal{H}$, this optimization over the function space can be written as an optimization over the parameter space as

$$\widehat{f}_{\text{ker}}(\mathbf{x}) = \left\langle \phi(\mathbf{x}), \widehat{\boldsymbol{\theta}} \right\rangle \quad \widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(y_i, \langle \phi(\mathbf{x}_i), \boldsymbol{\theta} \rangle\right) + \lambda \|\boldsymbol{\theta}\|_{L^2}^2. \tag{2.26}$$

Note that this optimization is often very high-dimensional as the dimension of feature space could be very high or even infinite. By the representer theorem [111], the solution to the optimization problem in (2.25) has the form

$$\widehat{f}_{\text{ker}}(\mathbf{x}) = \sum_{i=1}^{N} K(\mathbf{x}, \mathbf{x}_i)\alpha_i. \tag{2.27}$$

By the reproducing property of the kernel, it is easy to show that $\|\widehat{f}_{\text{ker}}\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^{\mathsf{T}} \mathbf{K} \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^{\mathsf{T}}$ and $\mathbf{K}$ is the data kernel matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. The optimization problem in (2.25) can then be written in terms of α_is as

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(y_i, \mathbf{K}_i \boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^{\mathsf{T}} \mathbf{K} \boldsymbol{\alpha}, \tag{2.28}$$

where $\mathbf{K}_i$ is the ith row of $\mathbf{K}$. Observe that this optimization problem only depends on the kernel evaluated over the data points, and hence the optimization problem in (2.25) can be solved without ever working in the feature space $\mathcal{H}$. If we let $\mathbf{X}_{\text{tr}}$ to represent the data matrix with $\mathbf{x}_i$ as its ith row, and $\mathbf{y}_{\text{tr}}$ the vector of observations, then for the special case of square

loss the optimization problem in (2.28) has the closed form solution $\widehat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}_{\text{tr}}$ which corresponds to the estimator

$$\widehat{f}_{\mathsf{krr}}(\mathbf{x}) = K(\mathbf{x}, \mathbf{X}_{\text{tr}})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}_{\text{tr}}, \tag{2.29}$$

where $K(\mathbf{x}, \mathbf{X}_{\text{tr}}) = [K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n)]$.

2.4.2 Gaussian Process Regression

A Gaussian process f is a stochastic process in which for every fixed set of points $\{\mathbf{x}_i\}_{i=1}^N$, the joint distribution of $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))$ has multivariate Gaussian distribution. As in multivariate Gaussian distribution, the distribution of a Gaussian process is completely determined by its first and second order statistics, known as the mean function and covariance kernel respectively. If we denote the mean function by $\mu(\cdot)$ and the covariance kernel by $K(\cdot, \cdot)$, then for any finite set of points

$$\Big(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)\Big) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{2.30}$$

where $\boldsymbol{\mu}$ the vector of mean values $\boldsymbol{\mu}_i = \mu(\mathbf{x}_i)$ and $\mathbf{K}$ is the covariance matrix with $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Next, assume that a priori we set the mean function to be zero everywhere. Then, the problem of Gaussian process regression can be stated as follows: we are given training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$

$$y_i = f(\mathbf{x}_i) + \xi_i, \quad \xi_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2), \tag{2.31}$$

where f is a zero mean Gaussian process with covariance kernel K. Given a test point $\mathbf{x}_{\text{ts}}$, we are interested in the posterior distribution of $y_{\text{ts}} := f(\mathbf{x}_{\text{ts}}) + \xi_{\text{ts}}$ given the training samples.

Defining $\mathbf{X}_{\text{tr}}$ and $\mathbf{y}_{\text{tr}}$ as in previous section we have

$$\begin{bmatrix} \mathbf{y}_{\text{tr}} \\ y_{\text{ts}} \end{bmatrix} \mid \mathbf{X}_{\text{tr}}, \mathbf{x}_{\text{ts}} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X_{\text{tr}}, X_{\text{tr}}) + \sigma^2 \mathbf{I} & K(\mathbf{X}_{\text{tr}}, \mathbf{x}_{\text{ts}}) \\ K(\mathbf{x}_{\text{ts}}, \mathbf{X}_{\text{tr}}) & K(\mathbf{x}_{\text{ts}}, \mathbf{x}_{\text{ts}}) + \sigma^2 \end{bmatrix} \right), \tag{2.32}$$

where $K(\mathbf{X}_{\text{tr}}, \mathbf{X}_{\text{tr}})$ is the kernel matrix evaluated at training points. Therefore, if we define $\mathbf{K} := K(\mathbf{X}_{\text{tr}}, \mathbf{X}_{\text{tr}})$ we have $y_{\text{ts}} | \mathbf{y}_{\text{tr}}, \mathbf{X}_{\text{tr}}, \mathbf{x}_{\text{ts}} \sim \mathcal{N}(\widehat{y}_{\text{ts}}, \sigma^2_{\text{ts}})$ where

$$\begin{aligned} \widehat{y}_{\text{ts}} =& K(\mathbf{x}_{\text{ts}}, \mathbf{X}_{\text{tr}})(\mathbf{K} + \sigma^2 I)^{-1} \mathbf{y}_{\text{tr}}, \\ \sigma^2_{\text{ts}} =& \sigma^2 + K(\mathbf{x}_{\text{ts}}, \mathbf{x}_{\text{ts}}) - K(\mathbf{x}_{\text{ts}}, \mathbf{X}_{\text{tr}})(\mathbf{K} + \sigma^2 \mathbf{I})^{-1} K(\mathbf{X}_{\text{tr}}, \mathbf{x}_{\text{ts}}). \end{aligned} \tag{2.33}$$

For a given $\mathbf{x}_{\text{ts}}$, the MMSE estimator is given by $\widehat{f}_{\text{MMSE}}(\mathbf{x}_{\text{ts}}) = \widehat{y}_{\text{ts}}$ where $\widehat{y}_{\text{ts}}$ minimizes the posterior risk

$$\mathcal{E}(\mathbf{x}_{\text{ts}}) := \mathbb{E}[(\widehat{y}_{\text{ts}} - y_{\text{ts}})^2 \, | \mathbf{x}_{\text{ts}}, \mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}] \tag{2.34}$$

and the expectation is with respect to the randomness in f as well as $\{\xi_i\}$. The estimator that minimizes this risk is the mean of the posterior, i.e. $\widehat{y}_{\text{ts}}$ in (2.33) is the Bayes optimal estimator with respect to mean squared error and its MSE is $\mathcal{E}(\mathbf{x}_{\text{ts}}) = \sigma^2_{\text{ts}}$. Note that while this estimator is linear in the training outputs, it is nonlinear in the input data.

The problem of Gaussian process regression arises for systems that are in the Gaussian kernel regime. More specifically, assume that we have training and test data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and $(\mathbf{x}_{\text{ts}}, y_{\text{ts}})$ that are generated by a parametric model $y = f(\mathbf{x}, \boldsymbol{\theta}) + \xi$ where $\xi \sim \mathcal{N}(0, \sigma^2)$. Furthermore, assume that conditioned on $\mathbf{X}_{\text{tr}}$ and $\mathbf{x}_{\text{ts}}$, the vector $[f(\mathbf{x}_{\text{ts}}, \boldsymbol{\theta}), f(\mathbf{x}_1, \boldsymbol{\theta}), \ldots, f(\mathbf{x}_N, \boldsymbol{\theta})]^{\mathsf{T}}$, which is $N+1$-dimensional vector of the function values on the training and test inputs is jointly Gaussian and zero mean. Also, for $\mathbf{x}$ and $\mathbf{x}'$, in the training and test inputs define the kernel function by

$$K(\mathbf{x}, \mathbf{x}') := \mathbb{E}_{\boldsymbol{\theta}} \left[f(\mathbf{x}, \boldsymbol{\theta}) f(\mathbf{x}', \boldsymbol{\theta}) \right]. \tag{2.35}$$

Then the problem of estimating $\widehat{y}_{\text{ts}}$ can be considered as a Gaussian regression problem.

An important instance of this kernel model is when $f(\mathbf{x}, \boldsymbol{\theta})$ a wide neural network with parameters $\boldsymbol{\theta}$ drawn from random Gaussian distributions and a linear last layer. In this case, one can show that conditioned on the input, all the pre-activation signals in the neural network, i.e. all the signals right before going through the non-linearities, as well as the gradients with respect to the parameters are Gaussian processes.

2.4.3 Neural Tangent Kernel

Consider a neural network function $f(\mathbf{x}, \boldsymbol{\theta}) = \widetilde{\mathbf{z}}^{(L)}(\mathbf{x}, \boldsymbol{\theta})$ defined recursively as

$$\begin{aligned} \mathbf{z}^{(0)}(\mathbf{x}, \boldsymbol{\theta}) &= \mathbf{x}, \\ \widetilde{\mathbf{z}}^{(\ell+1)}(\mathbf{x}, \boldsymbol{\theta}) &= \frac{1}{\sqrt{n_\ell}} \mathbf{W}^{(\ell)} \mathbf{z}^{(\ell)}(\mathbf{x}, \boldsymbol{\theta}) + \nu \mathbf{b}^{(\ell)}, \\ \mathbf{z}^{(\ell)}(\mathbf{x}, \boldsymbol{\theta}) &= \sigma(\widetilde{\mathbf{z}}^{(\ell)}(\mathbf{x}, \boldsymbol{\theta})), \end{aligned} \tag{2.36}$$

where σ is a elementwise nonlinearity, $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$, and $\boldsymbol{\theta}$ is the collection of all weights $\mathbf{W}^{(\ell)}$ and biases $\mathbf{b}^{(\ell)}$ which are all initialized with i.i.d. draws from the standard normal distribution. As noted in many works [34, 70, 81, 93], conditioned on the input signals, with a Lipschitz non-linearity $\sigma(\cdot)$, the entries of the pre-activations $\widetilde{\mathbf{z}}^{(\ell)}$ converge in distribution to an i.i.d. Gaussian processes in the limit of $n_1, \ldots, n_{L-1} \to \infty$ with covariance $\Sigma^{(\ell)}$ defined recursively as

$$\begin{aligned} \Sigma^{(\ell)}(\mathbf{x}, \mathbf{x}') &= \widetilde{\Sigma}^{(\ell)}(\mathbf{x}, \mathbf{x}') \otimes \mathbf{I}_{n_\ell}, \\ \widetilde{\Sigma}^{(\ell+1)}(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{(u,v) \sim \mathcal{N}(\mathbf{0}, \mathbf{M})} \sigma(u)\sigma(v) + \nu^2, \end{aligned} \tag{2.37}$$

with

$$\Sigma^{(1)}(\mathbf{x}, \mathbf{x}') = \frac{1}{n_0} \mathbf{x}^\top \mathbf{x}' + \nu^2, \qquad \mathbf{M} = \begin{bmatrix} \widetilde{\Sigma}^{(\ell)}(\mathbf{x}, \mathbf{x}) & \widetilde{\Sigma}^{(\ell)}(\mathbf{x}, \mathbf{x}') \\ \widetilde{\Sigma}^{(\ell)}(\mathbf{x}', \mathbf{x}) & \widetilde{\Sigma}^{(\ell)}(\mathbf{x}', \mathbf{x}') \end{bmatrix}.$$

Therefore, if the true model is a random deep network plus noise, the optimal estimator would be as in (2.33) with the covariance in (2.37) used as the kernel.

The main result of [63] considers the problem of fitting a neural network to a training data using gradient descent. It is shown that in the limit of wide networks (i.e. $n_\ell \to \infty$ for all ℓ), training a neural network with gradient descent is equivalent to fitting a kernel regression with respect to a specific kernel called the neural tangent kernel (NTK). When $f(\mathbf{x}, \boldsymbol{\theta})$ is a neural network with scalar output, the neural tangent kernel (NTK) is defined as

$$K(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \langle \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} f(\mathbf{x}'; \boldsymbol{\theta}) \rangle \,. \tag{2.38}$$

In the limit of wide fully connected neural networks, [63] show that this kernel converges in probability to a kernel that is fixed throughout the training $K(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) \stackrel{p}{=} K(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_0)$.

Similar to (2.37), the neural tangent kernel for the same architecture and initialization can be evaluated via a the following recursive equations:

$$\begin{aligned} K^{(\ell)}(\mathbf{x}, \mathbf{x}') &= \widetilde{K}^{(\ell)}(\mathbf{x}, \mathbf{x}') \otimes \mathbf{I}_{n_\ell}, \\ \widetilde{K}^{(\ell+1)}(\mathbf{x}, \mathbf{x}') &= \widetilde{\Sigma}^{(\ell+1)}(\mathbf{x}, \mathbf{x}') + \widetilde{K}^{(\ell)}(\mathbf{x}, \mathbf{x}') \; \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\mathbf{M})} \sigma'(u)\sigma'(v), \\ \widetilde{K}^{(1)} &= \widetilde{\Sigma}^{(1)}, \end{aligned} \tag{2.39}$$

with $\mathbf{M}$, and $\widetilde{\Sigma}$ defined in (2.37). The details of calculations can be found in [63]. Similar results for architectures other than fully connected networks have since been proven [3, 7, 129, 130].

For a fully connected network with ReLU non-linearities, the NTK has a closed recursive form given by [23]. Let $f(\mathbf{x}; \boldsymbol{\theta}) = \sqrt{\frac{2}{n_{L-1}}} \left\langle \mathbf{w}_L, \mathbf{z}^{(L-1)} \right\rangle$ with $\mathbf{z}^{(1)} = \sigma(\mathbf{W}_1 \mathbf{x})$ and

$$\mathbf{z}^{(\ell)} = \sigma\left(\sqrt{\frac{2}{n_{\ell-1}}} \mathbf{W}_\ell \mathbf{z}^{(\ell-1)} \right), \quad \ell = 2, \ldots, L-1, \tag{2.40}$$

where $\sigma(\cdot)$ is the ReLU function, $\mathbf{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, $\mathbf{w}_L \in \mathbb{R}^{L-1}$ and all the parameters $\mathbf{w}_L$ and $\mathbf{W}_\ell, \ell = 1, 2, \ldots, L-1$, are initialized with i.i.d. entries drawn from $\mathcal{N}(0,1)$. Then the corresponding NTK, $K(\mathbf{u}, \mathbf{v}) := K_L(\mathbf{u}, \mathbf{v})$ can be obtained recursively by

$$\Sigma_\ell(\mathbf{u}, \mathbf{v}) = \|\mathbf{u}\|\|\mathbf{v}\|\kappa_1\left(\frac{\Sigma_{\ell-1}(\mathbf{u}, \mathbf{v})}{\|\mathbf{u}\|\|\mathbf{v}\|}\right), \tag{2.41}$$

$$K_\ell(\mathbf{u}, \mathbf{v}) = \Sigma_\ell(\mathbf{u}, \mathbf{v}) + K_{\ell-1}(\mathbf{u}, \mathbf{v})\kappa_0\left(\frac{\Sigma_{\ell-1}(\mathbf{u}, \mathbf{v})}{\|\mathbf{u}\|\|\mathbf{v}\|}\right) \tag{2.42}$$

for $\ell = 1, \ldots, L$ and $K_0(\mathbf{u}, \mathbf{v}) = \Sigma_0(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\mathsf{T}\mathbf{v}$, and

$$\kappa_0(t) = 1/\pi(\pi - \arccos(t)),$$
$$\kappa_1(t) = 1/\pi\left(t\left(\pi - \arccos(t)\right) + \sqrt{1-t^2}\right).$$

It is important to note that in the NTK regime, the dynamics of learning by gradient descent are simple and completely captured by the kernel and the target output. Investigating the properties of the kernel can therefore help us understand the trainability and generalization of the model of interest. Moreover, these properties can elucidate understanding the implicit biases in different models. As we will see in Chapter 5, we use the NTK regime analysis to provide the rigorous explanation of the short-term bias in RNNs.

2.5 Recursions with Random Gaussians

In this section, we prove a special lemma that is useful in Chapter 5. To avoid confusion, please pay special attention to the vectors and matrices dimensions.

Consider a recursion of the form,

$$\mathbf{q}_{t+1} = \sum_{\ell=1}^{L} \frac{1}{\sqrt{n}} \mathbf{A}_\ell \overline{G}_\ell(\mathbf{q}_t, \mathbf{u}_t), \tag{2.43}$$

where $\mathbf{q}_t \in \mathbb{R}^{n \times d_q}$, $\mathbf{u}_t \in \mathbb{R}^{n \times d_u}$, and $\overline{G}_\ell(\mathbf{q}_t, \mathbf{u}_t)$ acts row-wise, meaning

$$\overline{G}_\ell(\mathbf{q}_t, \mathbf{u}_t)_{i,:} = G_\ell(\mathbf{q}_{t,i,:}, \mathbf{u}_{t,i,:}), \tag{2.44}$$

for some Lipschitz functions $G_\ell : \mathbb{R}^{d_q} \times \mathbb{R}^{d_u} \to \mathbb{R}^{d_q}$. That is, the outptut of row i of $\overline{G}_\ell(\cdot)$ depends only the i-th rows of its inputs. We will analyze this system for a fixed horizon, $t = 0, \ldots, T-1$. Assume that

$$(\mathbf{q}_0, \mathbf{u}_0, \ldots, \mathbf{u}_{T-1}) \overset{PL(2)}{\to} (Q_0, U_0, \ldots, U_{T-1}), \tag{2.45}$$

to random variables $(Q_0, U_0, \ldots, U_{T-1})$ where $Q_0 \in \mathbb{R}^{d_q}$ is independent of $(U_0, \ldots, U_{T-1})$, and $Q_0 \sim \mathcal{N}(0, P_0)$ for some covariance matrix $P_0 \in \mathbb{R}^{d_q \times d_q}$. Assume the matrices $\mathbf{A}_\ell \in \mathbb{R}^{n \times n}$ are independent with i.i.d. components, $(\mathbf{A}_\ell)_{i,j} \sim \mathcal{N}(0, \nu_\ell)$.

Lemma 1. *Under the above assumptions,*

$$(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{T-1}, \mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}) \overset{PL(2)}{\to} (Q_0, Q_1, \ldots, Q_{T-1}, U_0, \ldots U_{T-1}) \tag{2.46}$$

where each $Q_i \in \mathbb{R}^{d_q}$ and $(Q_0, Q_1, \ldots, Q_{T-1})$ are zero mean Gaussian processes independent

of $(U_0, \dots U_{T-1})$*, generated recursively through evolution equations given by*

$$D_\ell = \mathcal{N}(0, \nu_\ell), \tag{2.47a}$$

$$Z_{t\ell} = G_\ell(Q_t, U_t), \tag{2.47b}$$

$$\mu_{t\ell} = \mathbb{E}(Z_{t\ell}), \quad \widetilde{Z}_{t\ell} = Z_{t\ell} - \mu_{t\ell}, \tag{2.47c}$$

$$F_{t,:,\ell} = \min_{F_1,\dots,F_{t-1}} \mathbb{E}\left\| \widetilde{Z}_{t\ell} - \sum_{j=1}^{t} \widetilde{Z}_{t-j,\ell} F_j \right\|^2, \tag{2.47d}$$

$$P_{t\ell} = \mathbb{E}(\widetilde{Z}_{t\ell} - \sum_{j=1}^{t} \widetilde{Z}_{t-j,\ell} F_{tj\ell})^{\mathsf{T}} (\widetilde{Z}_{t\ell} - \sum_{j=1}^{t} \widetilde{Z}_{t-j,\ell} F_{tj\ell}), \tag{2.47e}$$

$$\widetilde{R}_{t\ell} = \sum_{j=1}^{t} \widetilde{R}_{t-j,\ell} F_{tj\ell} + \mathcal{N}(0, \nu_W P_{t\ell}), \tag{2.47f}$$

$$R_{t\ell} = \widetilde{R}_{t\ell} + D_\ell \mu_{t\ell}, \tag{2.47g}$$

$$Q_{t+1} = \sum_{\ell=1}^{L} R_{t\ell}. \tag{2.47h}$$

Proof. See Appendix A.1 for the proof. □

This lemma shows that $(q_0, \dots, q_t)$ converges $PL(2)$ to a Gaussian vector $(Q_0, \dots, Q_t)$ with zero mean. We use a special case of this lemma in Chapter 5.

Chapter 3

Learning Generalized Linear Models in High Dimensions

[1] At the heart of machine learning lies the question of generalizability of learned rules over previously unseen data. While over-parameterized models based on neural networks are now ubiquitous in machine learning applications, our understanding of their generalization capabilities is incomplete and this task is made harder by the non-convexity of the underlying learning problems. We provide a general framework to characterize the asymptotic generalization error for single-layer neural networks (i.e., generalized linear models) with arbitrary non-linearities, making it applicable to regression as well as classification problems. This framework enables analyzing the effect of (i) over-parameterization and non-linearity during modeling; (ii) choices of loss function, initialization, and regularizer during learning; and (iii) mismatch between training and test distributions. As examples, we analyze a few special cases, namely linear regression and logistic regression. We are also able to rigorously and analytically explain the *double descent* phenomenon in generalized linear models.

[1] This chapter is based on the work [47] and is coauthered with Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K. Fletcher.

3.1 Introduction

Quantifying the generalization of neural networks is one of the fundamental goals of machine learning and methods for this end are critical in assessing the performance of any machine learning approach. We are interested in characterizing the generalization error for a class of generalized linear models (GLMs) of the form

$$y = \phi_{\text{out}}(\langle \mathbf{x}, \mathbf{w}^0 \rangle, d), \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^p$ is a vector of input features, y is a scalar output, $\mathbf{w}^0 \in \mathbb{R}^p$ are weights to be learned, $\phi_{\text{out}}(\cdot)$ is a known link function, and d is random noise. The notation $\langle \mathbf{x}, \mathbf{w}^0 \rangle$ denotes an inner product. We use the superscript "0" to denote the "true" values in contrast to estimated or postulated quantities. The output may be continuous or discrete to model either regression or classification problems.

We measure the generalization error in a standard manner: we are given training data $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$ from which we learn some parameter estimate $\widehat{\mathbf{w}}$ via a regularized empirical risk minimization of the form

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} F_{\text{out}}(\mathbf{y}, \mathbf{X}\mathbf{w}) + F_{\text{in}}(\mathbf{w}), \tag{3.2}$$

where $\mathbf{X} = [\mathbf{x}_1 \, \mathbf{x}_2 \, \ldots \, \mathbf{x}_N]^\mathsf{T}$, is the data matrix, F_{out} is some output loss function, and F_{in} is some regularizer on the weights. We are then given a new test sample, $\mathbf{x}_{\text{ts}}$, for which the true and predicted values are given by

$$y_{\text{ts}} = \phi_{\text{out}}(\langle \mathbf{x}_{\text{ts}}, \mathbf{w}^0 \rangle, d_{\text{ts}}), \quad \widehat{y}_{\text{ts}} = \phi(\langle \mathbf{x}_{\text{ts}}, \widehat{\mathbf{w}} \rangle), \tag{3.3}$$

where d_{ts} is the noise in the test sample, and $\phi(\cdot)$ is a postulated inverse link function that may be different from the true function $\phi_{\text{out}}(\cdot)$. The generalization error is then defined as

the expectation of some expected loss between y_{ts} and $\widehat{y}_{\text{ts}}$ of the form

$$\mathbb{E}\, f_{\text{ts}}(y_{\text{ts}}, \widehat{y}_{\text{ts}}), \tag{3.4}$$

for some test loss function $f_{\text{ts}}(\cdot)$ such as squared error or prediction error.

Even for this relatively simple GLM model, the behavior of the generalization error is not fully understood. Recent works [36, 82, 88, 110] have characterized the generalization error of various linear models for classification and regression in certain large random problem instances. Specifically, the number of samples N and number of features p both grow without bound with their ratio satisfying $p/N \to \beta \in (0, \infty)$, and the samples in the training data $\mathbf{x}_i$ are drawn randomly. In this limit, the generalization error can be exactly computed. The analysis can explain the so-called *double descent* phenomena [17]: in highly under-regularized settings, the test error may initially *increase* with the number of data samples N before decreasing. Perhaps the first empirical evidence of the double descent curve can be traced back to [24].

3.1.1 Key Contributions

Our main result (Theorem 1) provides a procedure for exactly computing the asymptotic value of the generalization error (3.4) for GLM models in a certain random high-dimensional regime called the Large System Limit (LSL). The procedure enables the generalization error to be related to key problem parameters including the sampling ratio $\beta = p/N$, the regularizer, the output function, and the distributions of the true weights and noise. Importantly, our result holds under very general settings including: arbitrary test metrics f_{ts}; arbitrary training loss functions F_{out} as well as decomposable regularizers F_{in}; arbitrary link functions ϕ_{out}; correlated covariates $\mathbf{x}$; and distributional mismatch in training and test data.

3.1.2 Prior Work

Many recent works characterize generalization error of various machine learning models, including special cases of the GLM model considered here. For example, the precise characterization for asymptotics of prediction error for least squares regression has been provided in [18, 59, 89]. The former confirmed the double descent curve of [17] under a Fourier series model and a noisy Gaussian model for data in the over-parameterized regime. The latter obtained this scenario under both linear and non-linear feature models for ridge regression and min-norm least squares using random matrix theory. Also, [2] studied the same setting for deep linear and shallow non-linear networks.

The analysis of the the generalization for max-margin linear classifiers in the high dimensional regime has been done in [88]. The exact expression for asymptotic prediction error is derived and in a specific case for two-layer neural network with random first-layer weights, the double descent curve was obtained. A similar double descent curve for logistic regression as well as linear discriminant analysis has been reported by [36]. Random feature learning in the same setting has also been studied for ridge regression in [82]. The authors have, in particular, shown that highly over-parametrized estimators with zero training error are statistically optimal at high signal-to-noise ratio (SNR). The asymptotic performance of regularized logistic regression in high dimensions is studied in [110] using the Convex Gaussian Min-max Theorem in the under-parametrized regime. In what we present here, we can consider all these models as special cases. Bounds on the generalization error of over-parametrized linear models are also given in [13, 94].

Although what we present here and several other recent works consider only simple linear models and GLMs, much of the motivation is to understand generalization in deep neural networks where classical intuition may not hold [19, 94, 131]. In particular, a number of recent papers have shown the connection between neural networks in the over-parametrized regime and kernel methods [4, 8, 33, 34, 42, 63].

3.1.3 Approximate Message Passing

Our key tool to study the generalization error is approximate message passing (AMP), a class of inference algorithms originally developed in [15, 37, 39] for compressed sensing. We show that the learning problem for the GLM can be formulated as an inference problem on a certain multi-layer network. Multi-layer AMP methods [51, 60, 80, 98] can then be applied to perform the inference. The specific algorithm we use here is the multi-layer vector AMP (ML-VAMP) algorithm of [51, 98] which itself builds on several works [25, 49, 78, 96, 105]. The ML-VAMP algorithm is not necessarily the most computationally efficient procedure for the minimization (3.2). For our purposes, the key property is that ML-VAMP enables exact predictions of its performance in the large system limit. Specifically, the error of the algorithm estimates in each iteration can be predicted by a set of deterministic recursive equations called the *state evolution* or SE. The fixed points of these equations provide a way of computing the asymptotic performance of the algorithm. In certain cases, the algorithm can be proven to be Bayes optimal [1, 11, 53, 107].

This approach of using AMP methods to characterize the generalization error of GLMs was also explored in [11] for i.i.d. distributions on the data. The explicit formulae for the asymptotic mean squared error for the regularized linear regression with rotationally invariant data matrices is proved in [56]. The ML-VAMP method enables extensions to correlated features and to capture mismatch between training and test distributions.

3.2 System Model for Generalization Error

We consider the problem of estimating the weights $\mathbf{w}$ in the GLM model (3.1). As stated in the Introduction, we suppose we have training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ arranged as $\mathbf{X} := [\mathbf{x}_1\, \mathbf{x}_2 \ldots \mathbf{x}_N]^\mathsf{T} \in \mathbb{R}^{N\times p}$, $\mathbf{y} := [y_1\, y_2 \ldots y_N]^\mathsf{T} \in \mathbb{R}^N$. Then we can write

$$\mathbf{y} = \boldsymbol{\phi}_{\text{out}}(\mathbf{X}\mathbf{w}^0, \mathbf{d}), \tag{3.5}$$

where $\boldsymbol{\phi}_{\text{out}}(\mathbf{z}, \mathbf{d})$ is the vector-valued function such that $[\boldsymbol{\phi}_{\text{out}}(\mathbf{z}, \mathbf{d})]_n = \phi_{\text{out}}(z_n, d_n)$ and $\{d_n\}_{n=1}^N$ are general noise.

Given the training data $(\mathbf{X}, \mathbf{y})$, we consider estimates of $\mathbf{w}^0$ given by a regularized empirical risk minimization of the form (3.2). We assume that the loss function F_{out} and regularizer F_{in} are separable functions, i.e., one can write

$$F_{\text{out}}(\mathbf{y}, \mathbf{z}) = \sum_{n=1}^{N} f_{\text{out}}(y_n, z_n), \quad F_{\text{in}}(\mathbf{w}) = \sum_{j=1}^{p} f_{\text{in}}(w_j), \tag{3.6}$$

for some functions $f_{\text{out}} : \mathbb{R}^2 \to \mathbb{R}$ and $f_{\text{in}} : \mathbb{R} \to \mathbb{R}$. Many standard optimization problems in machine learning can be written in this form: logistic regression, support vector machines, linear regression, Poisson regression.

Large System Limit: We follow the LSL analysis of [15] commonly used for analyzing AMP-based methods. Specifically, we consider a sequence of problems indexed by the number of training samples N. For each N, we suppose that the number of features $p = p(N)$ grows linearly with N, i.e.,

$$\lim_{N \to \infty} \frac{p(N)}{N} \to \beta \tag{3.7}$$

for some constant $\beta \in (0, \infty)$. Note that $\beta > 1$ corresponds to the over-parameterized regime and $\beta < 1$ corresponds to the under-parameterized regime.

True parameter: We assume the *true* weight vector $\mathbf{w}^0$ has components whose empirical distribution converges as

$$\lim_{N \to \infty} \{w_n^0\} \overset{PL(2)}{=} W^0, \tag{3.8}$$

for some limiting random variable W^0. This means that the empirical distribution $\frac{1}{p} \sum_{i=1}^{p} \delta_{w_i}$ converges, in the Wasserstein-2 metric (see Chap. 6 [125]), to the distribution of the finite-variance random variable W^0. Importantly, the limit (3.8) will hold if the components $\{w_i^0\}_{i=1}^p$ are drawn i.i.d. from the distribution of W^0 with $\mathbb{E}(W^0)^2 < \infty$. However, the convergence

can also be satisfied by correlated sequences and deterministic sequences.

Training data input: For each N, we assume that the training input data samples, $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, N$, are i.i.d. and drawn from a p-dimensional Gaussian distribution with zero mean and covariance $\mathbf{\Sigma}_{\text{tr}} \in \mathbb{R}^{p \times p}$. The covariance can capture the effect of features being correlated. We assume the covariance matrix has an eigenvalue decomposition,

$$\mathbf{\Sigma}_{\text{tr}} = \tfrac{1}{p}\mathbf{V}_0^{\mathsf{T}} \text{diag}(\mathbf{s}_{\text{tr}}^2)\mathbf{V}_0, \tag{3.9}$$

where $\mathbf{s}_{\text{tr}}^2$ are the eigenvalues of $\mathbf{\Sigma}_{\text{tr}}$ and $\mathbf{V}_0 \in \mathbb{R}^{p \times p}$ is the orthogonal matrix of eigenvectors. The scaling $\frac{1}{p}$ ensures that the total variance of the samples, $\mathbb{E}\|\mathbf{x}_i\|^2$, does not grow with N. We will place a certain random model on $\mathbf{s}_{\text{tr}}$ and $\mathbf{V}_0$ momentarily.

Using the covariance (3.9), we can write the data matrix as

$$\mathbf{X} = \mathbf{U}\,\text{diag}(\mathbf{s}_{\text{tr}})\mathbf{V}_0, \tag{3.10}$$

where $\mathbf{U} \in \mathbb{R}^{N \times p}$ has entries drawn i.i.d. from $\mathcal{N}(0, \frac{1}{p})$. For the purpose of analysis, it is useful to express the matrix $\mathbf{U}$ in terms of its SVD:

$$\mathbf{U} = \mathbf{V}_2 \mathbf{S}_{\text{mp}} \mathbf{V}_1, \quad \mathbf{S}_{\text{mp}} := \begin{bmatrix} \text{diag}(\mathbf{s}_{\text{mp}}) & \mathbf{0} \\ \mathbf{0} & * \end{bmatrix} \tag{3.11}$$

where $\mathbf{V}_1 \in \mathbb{R}^{N \times N}$ and $\mathbf{V}_2 \in \mathbb{R}^{p \times p}$ are orthogonal and $\mathbf{S}_{\text{mp}} \in \mathbb{R}^{N \times p}$ with non-zero entries $\mathbf{s}_{\text{mp}} \in \mathbb{R}^{\min\{N,p\}}$ only along the principal diagonal. $\mathbf{s}_{\text{mp}}$ are the singular values of $\mathbf{U}$. A standard result of random matrix theory is that, since $\mathbf{U}$ is i.i.d. Gaussian with entries $\mathcal{N}(0, \frac{1}{p})$, the matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ are Haar-distributed on the group of orthogonal matrices and $\mathbf{s}_{\text{mp}}$ is such that

$$\lim_{N \to \infty} \{s_{\text{mp},i}\} \overset{PL(2)}{=} S_{\text{mp}}, \tag{3.12}$$

where $S_{\text{mp}} \geq 0$ is a non-negative random variable such that S_{mp}^2 satisfies the Marcencko-Pastur

distribution. Note that we describe the random variable S_{mp} defined in (3.12) where S_{mp}^2 has a rescaled Marchenko-Pastur distribution. Notice that the positive entries of $\mathbf{s}_{\text{mp}}$ are the positive eigenvalues of $\mathbf{U}^\mathsf{T}\mathbf{U}$ (or $\mathbf{U}\mathbf{U}^\mathsf{T}$).

Observe that $U_{ij} \sim N(0, \frac{1}{p})$, whereas, the standard scaling while studying the Marchenko-Pastur distribution in Section 2.2.1 is for matrices $\mathbf{H}$ such that $H_{ij} \sim \mathcal{N}(0, \frac{1}{N})$ (for e.g. see equation (1.10) from [122] and the discussion preceding it). Also notice that $\sqrt{\beta}\mathbf{U}$ has the same distribution as $\mathbf{H}$. Thus the results from [122] apply directly to the distributions of eigenvalues of $\beta\mathbf{U}^\mathsf{T}\mathbf{U}$ and $\beta\mathbf{U}\mathbf{U}^\mathsf{T}$. [2]

Training data output: Given the input data $\mathbf{X}$, we assume that the training outputs $\mathbf{y}$ are generated from (3.5), where the noise $\mathbf{d}$ is independent of $\mathbf{X}$ and has an empirical distribution which converges as

$$\lim_{N\to\infty} \{d_i\} \overset{PL(2)}{=} D. \tag{3.13}$$

Again, the limit (3.13) will be satisfied if $\{d_i\}_{i=1}^N$ are i.i.d. draws of random variable D with bounded second moments.

Test data: To measure the generalization error, we assume now that we are given a test point $\mathbf{x}_{\text{ts}}$, and we obtain the true output y_{ts} and predicted output $\widehat{y}_{\text{ts}}$ given by (3.3). We assume that the test data inputs are also Gaussian, i.e.,

$$\mathbf{x}_{\text{ts}}^\mathsf{T} = \mathbf{u}^\mathsf{T}\text{diag}(\mathbf{s}_{\text{ts}})\mathbf{V}_0, \tag{3.14}$$

where $\mathbf{u} \in \mathbb{R}^p$ has i.i.d. Gaussian components, $\mathcal{N}(0, \frac{1}{p})$, and $\mathbf{s}_{\text{ts}}$ and $\mathbf{V}_0$ are the eigenvalues and eigenvectors of the test data covariance matrix. That is, the test data sample has a

[2] Related to $\mathbf{S}_{\text{mp}}$ and $\mathbf{s}_{\text{mp}}$ from equation (3.11), we need to define two quantities $\mathbf{s}_{\text{mp}}^+ \in \mathbb{R}^N$ and $\mathbf{s}_{\text{mp}}^- \in \mathbb{R}^p$ that are zero-padded versions of the singular values $\mathbf{s}_{\text{mp}}$, so that for $n > \min\{N, p\}$, we set $s_{\text{mp},n}^{\pm} = 0$. Observe that $(\mathbf{s}_{\text{mp}}^+)^2$ are eigenvalues of $\mathbf{U}\mathbf{U}^\mathsf{T}$ whereas $(\mathbf{s}_{\text{mp}}^-)^2$ are eigenvalues of $\mathbf{U}^\mathsf{T}\mathbf{U}$. Since $\mathbf{s}_{\text{mp}}$ empirically converges to S_{mp} as given in (3.12), the vector $\mathbf{s}_{\text{mp}}^+$ empirically converges to random variable S_{mp}^+ whereas the vector $\mathbf{s}_{\text{mp}}^-$ empirically converges to random variable S_{mp}^-, where a mass is placed at 0 appropriately. Specifically, S_{mp}^+ has a point mass of $(1-\beta)_+\delta_{\{0\}}$ when $\beta < 1$, whereas S_{mp}^- has a point mass of $(1-\frac{1}{\beta})_+\delta_{\{0\}}$, when $\beta > 1$. In Section 2.2.1 (eqn. (2.10)), we provide the densities over positive parts of S_{mp}^+ and S_{mp}^-.

covariance matrix

$$\boldsymbol{\Sigma}_{\text{ts}} = \tfrac{1}{p}\mathbf{V}_0^{\mathsf{T}}\text{diag}(\mathbf{s}_{\text{ts}}^2)\mathbf{V}_0. \tag{3.15}$$

In comparison to (3.9), we see that we are assuming that the eigenvectors of the training and test data are the same, but the eigenvalues may be different. In this way, we can capture distributional mismatch between the training and test data. For example, we will be able to measure the generalization error when the test sample is outside a subspace explored by the training data.

To capture the relation between the training and test distributions, we assume that components of $\mathbf{s}_{\text{tr}}$ and $\mathbf{s}_{\text{ts}}$ converge as

$$\lim_{N\to\infty}\{(s_{\text{tr},i}, s_{\text{ts},i})\} \overset{PL(2)}{=} (S_{\text{tr}}, S_{\text{ts}}), \tag{3.16}$$

to some non-negative, bounded random vector $(S_{\text{tr}}, S_{\text{ts}})$. The joint distribution on $(S_{\text{tr}}, S_{\text{ts}})$ captures the relation between the training and test data.

When $S_{\text{tr}} = S_{\text{ts}}$, our model corresponds to the case when the training and test distribution are matched. Isotropic Gaussian features in both training and test data correspond to covariance matrices $\boldsymbol{\Sigma}_{\text{tr}} = \frac{1}{p}\sigma_{\text{tr}}^2\mathbf{I}$, $\boldsymbol{\Sigma}_{\text{ts}} = \frac{1}{p}\sigma_{\text{ts}}^2\mathbf{I}$, which can be modeled as $S_{\text{tr}} = \sigma_{\text{tr}}$, $S_{\text{ts}} = \sigma_{\text{ts}}$. We also require that the matrix $\mathbf{V}_0$ is uniformly distributed on the set of $p \times p$ orthogonal matrices.

Generalization error: From the training data, we obtain an estimate $\widehat{\mathbf{w}}$ via a regularized empirical risk minimization (3.2). Given a test sample $\mathbf{x}_{\text{ts}}$ and parameter estimate $\widehat{\mathbf{w}}$, the true output y_{ts} and predicted output $\widehat{y}_{\text{tr}}$ are given by equation (3.3). We assume the test noise is distributed as $d_{\text{ts}} \sim D$, following the same distribution as the training data. The postulated inverse-link function $\phi(\cdot)$ in (3.3) may be different from the true inverse-link function $\phi_{\text{out}}(\cdot)$.

The generalization error is defined as the asymptotic expected loss,

$$\mathcal{E}_{\text{ts}} := \lim_{N\to\infty} \mathbb{E} f_{\text{ts}}(\widehat{y}_{\text{ts}}, y_{\text{ts}}), \tag{3.17}$$

where $f_{\text{ts}}(\cdot)$ is some loss function relevant for the test error (which may be different from the training loss). The expectation in (3.17) is with respect to the randomness in the training as well as test data, and the noise. Our main result provides a formula for the generalization error (3.17).

3.3 Learning GLMs via ML-VAMP

There are many methods for solving the minimization problem (3.2). We apply the ML-VAMP algorithm of [51, 97]. This algorithm is not necessarily the most computationally efficient method. For our purposes, however, the algorithm serves as a constructive proof technique, i.e., it enables exact predictions for generalization error in the LSL as described above. Moreover, in the case when loss function (3.2) is strictly convex, the problem has a unique global minimum, whereby the generalization error of this minimum is agnostic to the choice of algorithm used to find this minimum. To that end, we start by reformulating (3.2) in a form that is amicable to the application of ML-VAMP, Algorithm 2.

Multi-Layer Representation. The first step in applying ML-VAMP to the GLM learning problem is to represent the mapping from the true parameters $\mathbf{w}^0$ to the output $\mathbf{y}$ as a certain multi-layer network. We combine (3.5), (3.10) and (3.11), so that the mapping $\mathbf{w}^0 \mapsto \mathbf{y}$ can be written as the following sequence of operations (as illustrated in Fig. 3.1):

$$\begin{aligned}
\mathbf{z}_0^0 &:= \mathbf{w}^0, & \mathbf{p}_0^0 &:= \mathbf{V}_0 \mathbf{z}_0^0, \\
\mathbf{z}_1^0 &:= \boldsymbol{\phi}_1(\mathbf{p}_0^0, \boldsymbol{\xi}_1), & \mathbf{p}_1^0 &:= \mathbf{V}_1 \mathbf{z}_1^0, \\
\mathbf{z}_2^0 &:= \boldsymbol{\phi}_2(\mathbf{p}_1^0, \boldsymbol{\xi}_2), & \mathbf{p}_2^0 &:= \mathbf{V}_2 \mathbf{z}_2^0, \\
\mathbf{z}_3^0 &:= \boldsymbol{\phi}_3(\mathbf{p}_2^0, \boldsymbol{\xi}_3) = \mathbf{y},
\end{aligned} \tag{3.18}$$

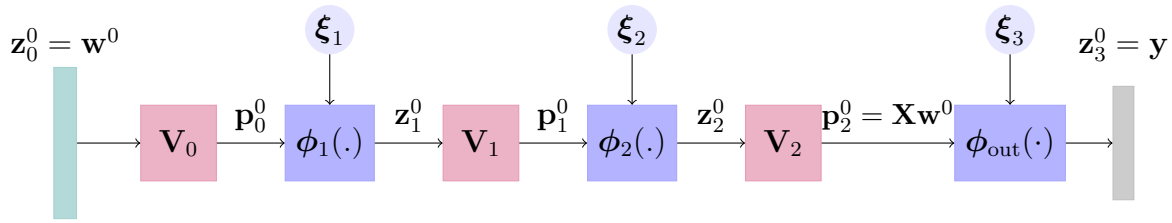

Figure 3.1: Sequence flow representing the mapping from the unknown parameter values $\mathbf{w}^0$ to the vector of responses $\mathbf{y}$ on the training features $\mathbf{X}$. $\mathbf{V}_\ell$ blocks represent multiplication by orthogonal operators and $\boldsymbol{\phi}_\ell(\cdot)$ blocks are non-linear functions acting coordinatewise. For the GLM learning problem we have $\boldsymbol{\xi}_1 = \mathbf{s}_{\text{tr}}$ and, $\boldsymbol{\xi}_2 = \mathbf{s}_{\text{mp}}, \boldsymbol{\xi}_3 = \mathbf{d}$. Also, $\boldsymbol{\phi}_1(\mathbf{p}_0, \mathbf{s}_{\text{tr}}) = \text{diag}(\mathbf{s}_{\text{tr}})\mathbf{p}_0, \boldsymbol{\phi}_2(\mathbf{p}_1, \mathbf{s}_{\text{mp}}) = \text{diag}(\mathbf{s}_{\text{mp}})\mathbf{p}_1$, and $\boldsymbol{\phi}_3(\mathbf{p}_2, \mathbf{d}) = \boldsymbol{\phi}_{\text{out}}(\mathbf{p}_2, \mathbf{d})$.

where $\boldsymbol{\xi}_\ell$ are the following vectors:

$$\boldsymbol{\xi}_1 := \mathbf{s}_{\text{tr}}, \quad \boldsymbol{\xi}_2 := \mathbf{s}_{\text{mp}}, \quad \boldsymbol{\xi}_3 := \mathbf{d}, \tag{3.19}$$

and the functions $\boldsymbol{\phi}_\ell(\cdot)$ are given by

$$\boldsymbol{\phi}_1(\mathbf{p}_0, \mathbf{s}_{\text{tr}}) := \text{diag}(\mathbf{s}_{\text{tr}})\mathbf{p}_0, \tag{3.20}$$

$$\boldsymbol{\phi}_2(\mathbf{p}_1, \mathbf{s}_{\text{mp}}) := \mathbf{S}_{\text{mp}}\mathbf{p}_1, \tag{3.21}$$

$$\boldsymbol{\phi}_3(\mathbf{p}_2, \mathbf{d}) := \boldsymbol{\phi}_{\text{out}}(\mathbf{p}_2, \mathbf{d}). \tag{3.22}$$

We see from Fig. 3.1 that the mapping of true parameters $\mathbf{w}^0 = \mathbf{z}_0^0$ to the observed response vector $\mathbf{y} = \mathbf{z}_3^0$ is described by a multi-layer network of alternating orthogonal operators $\mathbf{V}_\ell$ and non-linear functions $\boldsymbol{\phi}_\ell(\cdot)$. Let $L = 3$ denote the number of layers in this multi-layer network.

The minimization (3.2) can also be represented using a similar signal flow graph. Given a parameter candidate $\mathbf{w}$, the mapping $\mathbf{w} \mapsto \mathbf{X}\mathbf{w}$ can be written using the sequence of vectors

$$\begin{aligned} \mathbf{z}_0 &:= \mathbf{w}, & \mathbf{p}_0 &:= \mathbf{V}_0\mathbf{z}_0, \\ \mathbf{z}_1 &:= \mathbf{S}_{\text{tr}}\mathbf{p}_0, & \mathbf{p}_1 &:= \mathbf{V}_1\mathbf{z}_1, \\ \mathbf{z}_2 &:= \mathbf{S}_{\text{mp}}\mathbf{p}_1, & \mathbf{p}_2 &:= \mathbf{V}_2\mathbf{z}_2 = \mathbf{X}\mathbf{w}. \end{aligned} \tag{3.23}$$

There are $L = 3$ steps in this sequence, and we let

$$\mathbf{z} = \{\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2\}, \quad \mathbf{p} = \{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2\}$$

denote the sets of vectors across the steps. The minimization in (3.2) can then be written in the following equivalent form:

$$\begin{aligned} \min_{\mathbf{z},\mathbf{p}} \; & F_0(\mathbf{z}_0) + F_1(\mathbf{p}_0, \mathbf{z}_1) + F_2(\mathbf{p}_1, \mathbf{z}_1) + F_3(\mathbf{p}_2) \\ & \text{subject to} \quad \mathbf{p}_\ell = \mathbf{V}_\ell \mathbf{z}_\ell, \quad \ell = 0, 1, 2, \end{aligned} \tag{3.24}$$

where the *penalty functions* F_ℓ are defined as

$$\begin{aligned} F_0(\cdot) &= F_{\text{in}}(\cdot), & F_1(\cdot,\cdot) &= \delta_{\{\mathbf{z}_1 = \mathbf{S}_{\text{tr}}\mathbf{p}_0\}}(\cdot,\cdot), \\ F_2(\cdot,\cdot) &= \delta_{\{\mathbf{z}_2 = \mathbf{S}_{\text{mp}}\mathbf{p}_1\}}(\cdot,\cdot), & F_3(\cdot) &= F_{\text{out}}(\mathbf{y},\cdot), \end{aligned} \tag{3.25}$$

where $\delta_{\mathcal{A}}(\cdot)$ is 0 on the set $\mathcal{A}$, and $+\infty$ on $\mathcal{A}^c$.

ML-VAMP for GLM Learning. Using this multi-layer representation, we can now apply the ML-VAMP algorithm from [51, 97] to solve the optimization (3.24). The steps are shown in Algorithm 2. These steps are a special case of the "MAP version" of ML-VAMP in [97], but with a slightly different set-up for the GLM problem. We will call these steps the ML-VAMP GLM Learning Algorithm.

The algorithm operates in a set of iterations indexed by k. In each iteration, a "forward pass" through the layers generates estimates $\widehat{\mathbf{z}}_{k\ell}$ for the hidden variables $\mathbf{z}_\ell^0$, while a "backward pass" generates estimates $\widehat{\mathbf{p}}_{k\ell}$ for the variables $\mathbf{p}_\ell^0$. In each step, the estimates $\widehat{\mathbf{z}}_{k\ell}$ and $\widehat{\mathbf{p}}_{k\ell}$ are produced by functions $\mathbf{g}_\ell^+(\cdot)$ and $\mathbf{g}_\ell^-(\cdot)$ called *estimators* or *denoisers*.

For the MAP version of ML-VAMP algorithm in [97], the denoisers are essentially

Algorithm 2 ML-VAMP GLM Learning Algorithm

Require: Denoisers $\mathbf{g}_0^+$, $\mathbf{g}_L^-$, and $\mathbf{g}_\ell^+, \mathbf{g}_\ell^-$ for $\ell = 1, \ldots, L-1$

1: Initialize $\gamma_{0\ell}^- > 0$, $\mathbf{r}_{0\ell}^- = 0$ for $\ell = 0, \ldots, L-1$
2: **for** $k = 0, 1, \ldots$ **do**
3: // `Forward Pass`
4: **for** $\ell = 0, \ldots, L-1$ **do**
5: **if** $\ell = 0$ **then**
6: $\widehat{\mathbf{z}}_{k0} = \mathbf{g}_0^+(\mathbf{r}_{k0}^-, \gamma_{k0}^-)$
7: **else**
8: $\widehat{\mathbf{z}}_{k\ell} = \mathbf{g}_\ell^+(\mathbf{r}_{k,\ell-1}^+, \mathbf{r}_{k\ell}^-, \gamma_{k,\ell-1}^+, \gamma_{k\ell}^-)$
9: **end if**
10: $\alpha_{k\ell}^+ = \langle \partial \widehat{\mathbf{z}}_{k\ell} / \partial \mathbf{r}_{k\ell}^- \rangle$
11: $\mathbf{r}_{k\ell}^+ = \dfrac{\mathbf{V}_\ell(\widehat{\mathbf{z}}_{k\ell} - \alpha_{k\ell}^+ \mathbf{r}_{k\ell}^-)}{1 - \alpha_{k\ell}^+}$
12: $\gamma_{k\ell}^+ = (1/\alpha_{k\ell}^+ - 1)\gamma_{k\ell}^-$
13: **end for**
14: // `Backward Pass`
15: **for** $\ell = L, \ldots, 1$ **do**
16: **if** $\ell = L$ **then**
17: $\widehat{\mathbf{p}}_{k,L-1} = \mathbf{g}_L^-(\mathbf{r}_{k,L-1}^+, \gamma_{k,L-1}^+)$
18: **else**
19: $\widehat{\mathbf{p}}_{k,\ell-1} = \mathbf{g}_\ell^-(\mathbf{r}_{k,\ell-1}^+, \mathbf{r}_{k+1,\ell}^-, \gamma_{k,\ell-1}^+, \gamma_{k+1,\ell}^-)$
20: **end if**
21: $\alpha_{k,\ell-1}^- = \langle \partial \widehat{\mathbf{p}}_{k,\ell-1} / \partial \mathbf{r}_{k,\ell-1}^+ \rangle$
22: $\mathbf{r}_{k+1,\ell-1}^- = \dfrac{\mathbf{V}_{\ell-1}^{\mathsf{T}}(\widehat{\mathbf{p}}_{k,\ell-1} - \alpha_{k,\ell-1}^- \mathbf{r}_{k,\ell-1}^+)}{1 - \alpha_{k,\ell-1}^-}$
23: $\gamma_{k+1,\ell-1}^- = (1/\alpha_{k,\ell-1}^- - 1)\gamma_{k,\ell-1}^+$
24: **end for**
25: **end for**

proximal-type operators defined as

$$\text{prox}_{F/\gamma}(\boldsymbol{u}) := \underset{\mathbf{x}}{\text{argmin}} \; F(\mathbf{x}) + \tfrac{\gamma}{2} \|\mathbf{x} - \boldsymbol{u}\|^2 . \tag{3.26}$$

An important property of the proximal operator is that for separable functions F of the form (3.6), we have $[\text{prox}_{F/\gamma}(\boldsymbol{u})]_i = \text{prox}_{f/\gamma}(\boldsymbol{u}_i)$.

In the case of the GLM model, for $\ell = 0$ and L, on lines 6 and 17, the denoisers are

proximal operators given by

$$\mathbf{g}_0^+(\mathbf{r}_0^-, \gamma_0^-) = \text{prox}_{F_{\text{in}}/\gamma_0^-}(\mathbf{r}_0^-), \tag{3.27a}$$

$$\mathbf{g}_3^-(\mathbf{r}_2^+, \mathbf{y}, \gamma_2^+) = \text{prox}_{F_{\text{out}}/\gamma_2^+}(\mathbf{r}_2^+). \tag{3.27b}$$

Note that in (3.27b), there is a dependence on $\mathbf{y}$ through the term $F_{\text{out}}(\mathbf{y}, \cdot)$. For the *middle* terms, $\ell = 1, 2$, i.e., lines 8 and 19, the denoisers are given by

$$\mathbf{g}_\ell^+(\mathbf{r}_{\ell-1}^+, \mathbf{r}_\ell^-, \gamma_{\ell-1}^+, \gamma_\ell^-) := \widehat{\mathbf{z}}_\ell, \tag{3.28a}$$

$$\mathbf{g}_\ell^-(\mathbf{r}_{\ell-1}^+, \mathbf{r}_\ell^-, \gamma_{\ell-1}^+, \gamma_\ell^-) := \widehat{\mathbf{p}}_{\ell-1}, \tag{3.28b}$$

where $(\widehat{\mathbf{p}}_{\ell-1}, \widehat{\mathbf{z}}_\ell)$ are the solutions to the minimization

$$(\widehat{\mathbf{p}}_{\ell-1}, \widehat{\mathbf{z}}_\ell) := \underset{(\mathbf{p}_{\ell-1}, \mathbf{z}_\ell)}{\text{argmin}} \; F_\ell(\mathbf{p}_{\ell-1}, \mathbf{z}_\ell) + \frac{\gamma_\ell^-}{2}\|\mathbf{z}_\ell - \mathbf{r}_\ell^-\|^2 + \frac{\gamma_{\ell-1}^+}{2}\|\mathbf{p}_{\ell-1} - \mathbf{r}_{\ell-1}^+\|^2. \tag{3.29}$$

The quantity $\langle \partial \boldsymbol{v}/\partial \boldsymbol{u} \rangle$ on lines 10 and 21 denotes the empirical mean $\frac{1}{N}\sum_{n=1}^N \partial v_n/\partial u_n$.

Thus, the ML-VAMP algorithm in Algorithm 2 reduces the joint constrained minimization (3.24) over variables $(\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2)$ and $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ to a set of proximal operations on pairs of variables $(\mathbf{p}_{\ell-1}, \mathbf{z}_\ell)$. As discussed in [97], this type of minimization is similar to ADMM with adaptive step-sizes. Details of the denoisers $\mathbf{g}_\ell^\pm$ and other aspects of the algorithm are given in Section 3.5.

3.4 Main Result

We make two assumptions. The first assumption imposes certain regularity conditions on the functions f_{ts}, ϕ, ϕ_{out}, and maps $\mathbf{g}_\ell^\pm$ appearing in Algorithm 2. The precise definitions of pseudo-Lipschitz continuity and uniform Lipschitz continuity are given in Section 2.2.

Assumption 1. The denoisers and link functions satisfy the following continuity conditions:

(*a*) The proximal operators in (3.27),

$$\mathbf{g}_0^+(\mathbf{r}_0^-, \gamma_0^-), \quad \mathbf{g}_3^-(\mathbf{r}_2^+, \mathbf{y}, \gamma_2^+),$$

are uniformly Lipschitz continuous in $\mathbf{r}_0^-$ and $(\mathbf{r}_2^+, \mathbf{y})$ over parameters γ_0^- and γ_2^+.

(*b*) The link function $\phi_{\rm out}(p, d)$ is Lipschitz continuous in (p, d). The test error function $f_{\rm ts}(\phi(\widehat{z}), \phi_{\rm out}(z, d))$ is pseduo-Lipschitz continuous in $(\widehat{z}, z, d)$ of order 2.

Our second assumption is that the ML-VAMP algorithm converges. Specifically, let $\mathbf{x}_k = \mathbf{x}_k(N)$ be any set of outputs of Algorithm 2, at some iteration k and dimension N. For example, $\mathbf{x}_k(N)$ could be $\widehat{\mathbf{z}}_{k\ell}(N)$ or $\widehat{\mathbf{p}}_{k\ell}(N)$ for some ℓ, or a concatenation of signals such as $\begin{bmatrix} \mathbf{z}_\ell^0(N) & \widehat{\mathbf{z}}_{k\ell}(N) \end{bmatrix}$.

Assumption 2. Let $\mathbf{x}_k(N)$ be any finite set of outputs of the ML-VAMP algorithm as above. Then there exist limits

$$\mathbf{x}(N) = \lim_{k\to\infty} \mathbf{x}_k(N), \tag{3.30}$$

satisfying

$$\lim_{k\to\infty} \lim_{N\to\infty} \frac{1}{N} \|\mathbf{x}_k(N) - \mathbf{x}(N)\|^2 = 0. \tag{3.31}$$

We are now ready to state our main result.

Theorem 1. *Consider the GLM learning problem* (3.2) *solved by applying Algorithm 2 to the equivalent problem* (3.24) *under the assumptions of Section 3.2 along with Assumptions 1 and 2. Then, there exist constants* $\tau_0^-, \overline{\gamma}_0^+ > 0$ *and* $\mathbf{M} \in \mathbb{R}^{2\times 2}_{\succ 0}$ *such that the following hold:*

(a) The fixed points $\{\widehat{\mathbf{z}}_\ell, \widehat{\mathbf{p}}_\ell\}$, $\ell = 0, 1, 2$ *of Algorithm 2 satisfy the KKT conditions for the constrained optimization problem* (3.24). *Equivalently* $\widehat{\mathbf{w}} := \widehat{\mathbf{z}}_0$ *is a stationary point of* (3.2).

(b) The true parameter $\mathbf{w}^0$ *and its estimate* $\widehat{\mathbf{w}}$ *empirically converge as*

$$\lim_{N\to\infty} \{(w_i^0, \widehat{w}_i)\} \overset{PL(2)}{=} (W^0, \widehat{W}), \tag{3.32}$$

where W^0 *is the random variable from* (3.8) *and*

$$\widehat{W} = \mathrm{prox}_{f_{\mathrm{in}}/\overline{\gamma}_0^+}(W^0 + Q_0^-), \tag{3.33}$$

with $Q_0^- = \mathcal{N}(0, \tau_0^-)$ *independent of* W^0.

(c) The asymptotic generalization error (3.17) *with* $(y_{\mathrm{ts}}, \widehat{y}_{\mathrm{ts}})$ *defined as* (3.3) *is given by*

$$\mathcal{E}_{\mathrm{ts}} = \mathbb{E}\, f_{\mathrm{ts}}\Big(\phi_{\mathrm{out}}(Z_{\mathrm{ts}}, D), \phi(\widehat{Z}_{\mathrm{ts}})\Big), \tag{3.34}$$

where $(Z_{\mathrm{ts}}, \widehat{Z}_{\mathrm{ts}}) \sim \mathcal{N}(\mathbf{0}_2, \mathbf{M})$ *and independent of* D.

Part (a) shows that, similar to gradient descent, Algorithm 2 finds the stationary points of problem (3.2). These stationary points will be unique in strictly convex problems such as linear and logistic regression. Thus, in such cases, the same results will be true for any algorithm that finds such stationary points. Hence, the fact that we are using ML-VAMP is immaterial – our results apply to any solver for (3.2). Note that the convergence to the fixed points $\{\widehat{\mathbf{z}}_\ell, \widehat{\mathbf{p}}_\ell\}$ is assumed from Assumption 2.

Part (b) provides an exact description of the asymptotic statistical relation between the true parameter $\mathbf{w}^0$ and its estimate $\widehat{\mathbf{w}}$. The parameters $\tau_0^-, \overline{\gamma}_0^+ > 0$ and $\mathbf{M}$ can be explicitly computed using a set of recursive equations called the *state evolution* or SE described in Section 3.6.

We can use the expressions to compute a variety of relevant metrics. For example, the

$PL(2)$ convergence shows that the MSE on the parameter estimate is

$$\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N}(w_n^0 - \widehat{w}_n)^2 = \mathbb{E}(W^0 - \widehat{W})^2. \tag{3.35}$$

The expectation on the right hand side of (3.35) can then be computed via integration over the joint density of $(W^0, \widehat{W})$ from part (b). In this way, we have a simple and exact method to compute the parameter error. Other metrics such as parameter bias or variance, cosine angle or sparsity detection can also be computed.

Part (c) of Theorem 1 similarly exactly characterizes the asymptotic generalization error. In this case, we would compute the expectation over the three variables $(Z, \widehat{Z}, D)$. In this way, we have provided a methodology for exactly predicting the generalization error from the key parameters of the problems such as the sampling ratio $\beta = p/N$, the regularizer, the output function, and the distributions of the true weights and noise. We provide several examples such as linear regression, logistic regression and SVM in Section 3.7. We also recover the result by [59]. The detailed proof of Theorem 1 is given in Appendix B.2.

Remarks on Assumptions. Note that Assumption 1 is satisfied in many practical cases. For example, it can be verified that it is satisfied in the case when $f_{\text{in}}(\cdot)$ and $f_{\text{out}}(\cdot)$ are convex. Assumption 2 is somewhat more restrictive in that it requires that the ML-VAMP algorithm converges. The convergence properties of ML-VAMP are discussed in [49]. The ML-VAMP algorithm may not always converge, and characterizing conditions under which convergence is possible is an open question. However, experiments in [105] show that the algorithm does indeed often converge, and in these cases, our analysis applies. In any case, we will see below that the predictions from Theorem 1 agree closely with numerical experiments in several relevant cases. In some special cases equation (3.34) simplifies to yield quantitative insights for interesting modeling artifacts. We discuss these in Section 3.7.

3.5 ML-VAMP Denoisers

A key property of our analysis will be that the non-linear functions (3.20) and the denoisers $\mathbf{g}_\ell^{\pm}(\cdot)$ have simple forms.

Non-linear functions $\boldsymbol{\phi}_\ell(\cdot)$: The non-linear functions all act *componentwise*. For example, for $\boldsymbol{\phi}_1(\cdot)$ in (3.20), we have

$$\mathbf{z}_1 = \boldsymbol{\phi}_1(\mathbf{p}_0, \mathbf{s}_{\rm tr}) = \mathrm{diag}(\mathbf{s}_{\rm tr})\mathbf{p}_0 \Longleftrightarrow z_{1,n} = \phi_1(p_{0,n}, s_{{\rm tr},n}),$$

where $\phi_1(\cdot)$ is the scalar-valued function, $\phi_1(p_0, s) = sp_0$. Similarly, for $\boldsymbol{\phi}_2(\cdot)$,

$$\mathbf{z}_2 = \boldsymbol{\phi}_2(\mathbf{p}_1, \mathbf{s}_{\rm mp}^+) \Longleftrightarrow z_{2,n} = \phi_2(\mathbf{w}p_{1,n}, s_{{\rm mp},n}^+), \quad n < N$$

where $\mathbf{w}\mathbf{p}_1 \in \mathbb{R}^N$ is the zero-padded version of $\mathbf{p}_1$, and $\phi_2(p_1, s) = s\, p_1$. Finally, the function $\boldsymbol{\phi}_3(\cdot)$ in (3.20) acts component-wise with $\phi_3(p_2, d) = \phi_{\rm out}(p_2, d)$.

Input denoiser $\mathbf{g}_0^+(\cdot)$: Since $F_0(\mathbf{z}_0) = F_{\rm in}(\mathbf{z}_0)$, and $F_{\rm in}(\cdot)$ given in (3.6), the denoiser (3.27a) acts *componentwise* in that,

$$\widehat{\mathbf{z}}_0 = \mathbf{g}_0^+(\mathbf{r}_0^-, \gamma_0^-) \Longleftrightarrow \widehat{z}_{0,n} = g_0^+(r_{0,n}^-, \gamma_0^-),$$

where $g_0^+(\cdot)$ is the scalar-valued function,

$$g_0^+(r_0^-, \gamma_0^-) := \underset{z_0}{\mathrm{argmin}} \;\; f_{\rm in}(z_0) + \frac{\gamma_0^-}{2}(z_0 - r_0^-)^2. \tag{3.36}$$

Thus, the vector optimization in (3.27a) reduces to a set of scalar optimizations (3.36) on each component.

Output denoiser $\mathbf{g}_3^-(\cdot)$: The output penalty $F_3(\mathbf{p}_2, \mathbf{y}) = F_{\rm out}(\mathbf{p}_2, \mathbf{y})$ where $F_{\rm out}(\mathbf{p}_2, \mathbf{y})$ has the separable form (3.6). Thus, similar to the case of $\mathbf{g}_0(\cdot)$, the denoiser $\mathbf{g}_3(\cdot)$ in (3.27b) also acts

componentwise with the function,

$$g_3^-(r_2^+, \gamma_2^+, y) := \underset{p_2}{\operatorname{argmin}} \; f_{\text{out}}(p_2, y) + \tfrac{\gamma_2^+}{2}(p_2 - r_2^+)^2. \tag{3.37}$$

Linear denoiser $\mathbf{g}_1^{\pm}(\cdot)$: The expressions for both denoisers $g_1^{\pm}$ and $g_2^{\pm}$ are very similar and can be explained together. The penalty $F_1(\cdot)$ restricts $\mathbf{z}_1 = \mathbf{S}_{\text{tr}}\mathbf{p}_0$, where $\mathbf{S}_{\text{tr}}$ is a square matrix. Hence, for $\ell = 1$, the minimization in (3.29) is given by,

$$\widehat{\mathbf{p}}_0 := \underset{\mathbf{p}_0}{\operatorname{argmin}} \; \tfrac{\gamma_0^+}{2}\|\mathbf{p}_0 - \mathbf{r}_0^+\|^2 + \tfrac{\gamma_1^-}{2}\|\mathbf{S}_{\text{tr}}\mathbf{p}_0 - \mathbf{r}_1^-\|^2, \tag{3.38}$$

and $\widehat{\mathbf{z}}_1 = \mathbf{S}_{\text{tr}}\widehat{\mathbf{p}}_0$. This is a simple quadratic minimization and the components of $\widehat{\mathbf{p}}_0$ and $\widehat{\mathbf{z}}_1$ are given by

$$\begin{aligned}
\widehat{p}_{0,n} &= g_1^-(r_{0,n}^+, r_{1,n}^-, \gamma_0^+, \gamma_1^-, s_{\text{tr},n}), \\
\widehat{z}_{1,n} &= g_1^+(r_{0,n}^+, r_{1,n}^-, \gamma_0^+, \gamma_1^-, s_{\text{tr},n}),
\end{aligned}$$

where

$$g_1^-(r_0^+, r_1^-, \gamma_0^+, \gamma_1^-, s) := \frac{\gamma_0^+ r_0^+ + s\gamma_1^- r_1^-}{\gamma_0^+ + s^2\gamma_1^-}, \tag{3.39a}$$

$$g_1^+(r_0^+, r_1^-, \gamma_0^+, \gamma_1^-, s) := \frac{s(\gamma_0^+ r_0^+ + s\gamma_1^- r_1^-)}{\gamma_0^+ + s^2\gamma_1^-}. \tag{3.39b}$$

Linear denoiser $\mathbf{g}_2^{\pm}(\cdot)$: This denoiser is identical to the case $\mathbf{g}_1^{\pm}(\cdot)$ in that we need to impose the linear constraint $\mathbf{z}_2 = \mathbf{S}_{\text{mp}}\mathbf{p}_1$. However $\mathbf{S}_{\text{mp}}$ is in general a rectangular matrix and the two resulting cases of $\beta \lessgtr 1$ needs to be treated separately. Recall the definitions of vectors $\mathbf{s}_{\text{mp}}^+$ and $\mathbf{s}_{\text{mp}}^-$ at the beginning of this section. Then, for $\ell = 2$, with the penalty

$F_2(\mathbf{p}_1, \mathbf{z}_2) = \delta_{\{\mathbf{z}_2 = \mathbf{S}_{\rm mp}\mathbf{p}_1\}}$, the solution to (3.29) has components,

$$\widehat{p}_{1,n} = g_2^-(r_{1,n}^+, r_{2,n}^-, \gamma_1^+, \gamma_2^+, s_{{\rm mp},n}^-), \tag{3.40a}$$

$$\widehat{z}_{2,n} = g_2^+(r_{1,n}^+, r_{2,n}^-, \gamma_1^+, \gamma_2^+, s_{{\rm mp},n}^+), \tag{3.40b}$$

with the identical functions $g_2^- = g_1^-$ and $g_2^+ = g_1^+$ as given by (3.39a) and (3.39b). Note that in (3.40a), $n = 1, \ldots, p$ and in (3.40b), $n = 1, \ldots, N$.

3.6 State Evolution Analysis of ML-VAMP

A key property of the ML-VAMP algorithm is that its performance in the LSL can be exactly described by a *scalar equivalent system*. In the scalar equivalent system, the vector-valued outputs of the algorithm are replaced by scalar random variables representing the typical behavior of the components of the vectors in the large-scale-limit (LSL). Each of the random variables are described by a set of parameters, where the parameters are given by a set of deterministic equations called the *state evolution* or SE.

The SE for the general ML-VAMP algorithm are derived in [97] and the special case of the updates for ML-VAMP for GLM learning are shown in Algorithm 3 with details of functions $\mathbf{g}_\ell^\pm$ in Section 3.5. We see that the SE updates in Algorithm 3 parallel those in the ML-VAMP algorithm Algo. 2, except that vector quantities such as $\widehat{\mathbf{z}}_{k\ell}$, $\widehat{\mathbf{p}}_{k\ell}$, $\mathbf{r}_{k\ell}^+$ and $\mathbf{r}_{k\ell}^-$ are replaced by scalar random variables such as $\widehat{Z}_{k\ell}$, $\widehat{P}_{k\ell}$, $R_{k\ell}^+$ and $R_{k\ell}^-$. Each of these random variables are described by the deterministic parameters such as $\mathbf{K}_{k\ell} \in \mathbb{R}^{2\times 2}_{\succ 0}$, and τ_ℓ^0, $\tau_{k\ell}^- \in \mathbb{R}_+$.

The updates in the section labeled as "Initial", provide the scalar equivalent model for the true system (3.18). In these updates, Ξ_ℓ represent the limits of the vectors $\boldsymbol{\xi}_\ell$ in (3.19). That is,

$$\Xi_1 := S_{\rm tr}, \quad \Xi_2 := S_{\rm mp}^+, \quad \Xi_3 := D. \tag{3.41}$$

Due to assumptions in Section 3.2, we have that the components of $\boldsymbol{\xi}_\ell$ converge empirically

Algorithm 3 SE for ML-VAMP for GLM Learning

Require: Functions g_0^+, g_L^-, and g_ℓ^+, g_ℓ^- for $\ell = 1, \ldots, L-1$

Initialize $\overline{\gamma}_{0\ell}^- = \gamma_{0\ell}^-$ from Algorithm 2.
$Q_{0\ell}^- \sim \mathcal{N}(0, \tau_{0\ell}^-)$ for some $\tau_{0\ell}^- > 0$ for $\ell = 0, 1, 2$
$Z_0^0 = W^0$
for $\ell = 0, \ldots, L-1$ **do**
    $P_\ell^0 = \mathcal{N}(0, \tau_\ell^0), \quad \tau_\ell^0 = \mathrm{var}(Z_\ell^0)$
    $Z_{\ell+1}^0 = \phi_{\ell+1}(P_\ell^0, \Xi_{\ell+1})$
end for
for $k = 0, 1, \ldots$ **do**
    // `Forward Pass`
    for $\ell = 0, \ldots, L-1$ **do**
        if $\ell = 0$ **then**
            $R_{k0}^- = Z_\ell^0 + Q_{k0}^-$
            $\widehat{Z}_{k0} = g_0^+(R_{k0}^-, \overline{\gamma}_{k0}^-)$
        else
            $R_{k,\ell-1}^+ = P_{\ell-1}^0 + P_{k,\ell-1}^+$, $R_{k\ell}^- = Z_\ell^0 + Q_{k\ell}^-$
            $\widehat{Z}_{k\ell} = g_\ell^+(R_{k,\ell-1}^+, R_{k\ell}^-, \overline{\gamma}_{k,\ell-1}^+, \overline{\gamma}_{k\ell}^-, \Xi_\ell)$
        end if
        $\overline{\alpha}_{k\ell}^+ = \mathbb{E}\partial \widehat{Z}_{k\ell} / \partial Q_{k\ell}^-$
        $Q_{k\ell}^+ = \dfrac{\widehat{Z}_{k\ell} - Z_\ell^0 - \overline{\alpha}_{k\ell}^+ Q_{k\ell}^-}{1 - \overline{\alpha}_{k\ell}^+}$
        $\overline{\gamma}_{k\ell}^+ = (\frac{1}{\overline{\alpha}_{k\ell}^+} - 1)\overline{\gamma}_{k\ell}^-$
        $(P_\ell^0, P_{k\ell}^+) \sim \mathcal{N}(0, \mathbf{K}_{k\ell}^+), \ \mathbf{K}_{k\ell}^+ = \mathrm{cov}(Z_\ell^0, Q_{k\ell}^+)$
    end for
    // `Backward Pass`
    for $\ell = L, \ldots, 1$ **do**
        if $\ell = L$ **then**
            $R_{k,L-1}^+ = P_{L-1}^0 + P_{k,L-1}^+$
            $\widehat{P}_{k,L-1} = g_L^-(R_{k,L-1}^+, \overline{\gamma}_{k,L-1}^+, Z_L^0)$
        else
            $R_{k,\ell-1}^+ = P_{\ell-1}^0 + P_{k,\ell-1}^+$, $R_{k+1,\ell}^- = Z_\ell^0 + Q_{k+1,\ell}^-$
            $\widehat{P}_{k,\ell-1} = g_\ell^-(R_{k,\ell-1}^+, R_{k+1,\ell}^-, \overline{\gamma}_{k,\ell-1}^+, \overline{\gamma}_{k+1,\ell}^-, \Xi_\ell)$
        end if
        $\overline{\alpha}_{k,\ell-1}^- = \mathbb{E}\partial \widehat{P}_{k,\ell-1} / \partial P_{k,\ell-1}^+$
        $P_{k+1,\ell-1}^- = \dfrac{\widehat{P}_{k,\ell-1} - P_{\ell-1}^0 - \overline{\alpha}_{k,\ell-1}^- P_{k,\ell-1}^+}{1 - \overline{\alpha}_{k,\ell-1}^-}$
        $\overline{\gamma}_{k+1,\ell-1}^- = (\frac{1}{\overline{\alpha}_{k,\ell-1}^-} - 1)\overline{\gamma}_{k,\ell-1}^+$
        $Q_{k+1,\ell-1}^- \sim \mathcal{N}(0, \tau_{k+1,\ell-1}^-), \ \tau_{k,\ell-1}^- = \mathbb{E}(P_{k+1,\ell-1}^-)^2$
    end for
end for

as,

$$\lim_{N\to\infty}\{\xi_{\ell,i}\} \overset{PL(2)}{=} \Xi_\ell, \tag{3.42}$$

So, the Ξ_ℓ represent the asymptotic distribution of the components of the vectors $\boldsymbol{\xi}_\ell$.

The updates in sections labeled “Forward pass" and “Backward pass" in the SE equations in Algorithm 3 parallel those in Algorithm 2. The key quantities in these SE equations are the error variables,

$$\mathbf{p}^+_{k\ell} := \mathbf{r}^+_{k\ell} - \mathbf{p}^0_\ell, \quad \mathbf{q}^-_{k\ell} := \mathbf{r}^-_{k\ell} - \mathbf{z}^0_\ell,$$

which represent the errors of the estimates to the inputs of the denoisers. We will also be interested in their transforms,

$$\mathbf{q}^+_{k\ell} = \mathbf{V}^{\mathsf{T}}_\ell \mathbf{p}^+_{k,\ell+1}, \quad \mathbf{p}^-_{k\ell} = \mathbf{V}_\ell \mathbf{q}^-_{k\ell}.$$

The following Theorem is an adapted version of the main result from [97] to the iterates of Algorithms 2 and 3.

Theorem 2. *Consider the outputs of the ML-VAMP for GLM Learning Algorithm under the assumptions of Section 3.2. Assume the denoisers satisfy the continuity conditions in Assumption 1. Also, assume that the outputs of the SE satisfy*

$$\overline{\alpha}^{\pm}_{k\ell} \in (0,1),$$

for all k and ℓ. Suppose Algo. 2 is initialized so that the following convergence holds:

$$\lim_{N\to\infty}\{\mathbf{r}^-_{0\ell} - \mathbf{z}^0_\ell\} \overset{PL(2)}{=} Q^-_{0\ell},$$

where $(Q^-_{00}, Q^-_{01}, Q^-_{02})$ are independent zero-mean Gaussians, independent of $\{\Xi_\ell\}$. Then,

(a) For any fixed iteration $k \geq 0$ in the forward direction and layer $\ell = 1,\ldots,L-1$, we

have that, almost surely,

$$\lim_{N\to\infty}(\gamma^+_{k,\ell-1},\gamma^-_{k\ell})=(\overline{\gamma}^+_{k,\ell-1},\overline{\gamma}^-_{k\ell}),\quad \text{and,} \tag{3.43}$$

$$\lim_{N\to\infty}\{(\widehat{\mathbf{z}}^+_{k\ell},\mathbf{z}^0_\ell,\mathbf{p}^0_{\ell-1},\mathbf{r}^+_{k,\ell-1},\mathbf{r}^-_\ell)\}$$
$$\overset{PL(2)}{=}(\widehat{Z}^+_{k\ell},Z^0_\ell,P^0_{\ell-1},R^+_{k,\ell-1},R^-_\ell), \tag{3.44}$$

where the variables on the right-hand side are from the SE equations (3.43) *and* (3.44) *are the outputs of the SE equations in Algorithm 3. Similar equations hold for* $\ell=0$ *with the appropriate variables removed.*

(b) Similarly, in the reverse direction, For any fixed iteration $k\geq 0$ *and layer* $\ell=1,\ldots,L-2$*, we have that, almost surely,*

$$\lim_{N\to\infty}(\gamma^+_{k,\ell-1},\gamma^-_{k+1,\ell})=(\overline{\gamma}^+_{k,\ell-1},\overline{\gamma}^-_{k+1,\ell}),\quad \text{and} \tag{3.45}$$

$$\lim_{N\to\infty}\{(\widehat{\mathbf{p}}^+_{k+1,\ell-1},\mathbf{z}^0_\ell,\mathbf{p}^0_{\ell-1},\mathbf{r}^+_{k,\ell-1},\mathbf{r}^-_{k+1,\ell})\}$$
$$\overset{PL(2)}{=}(\widehat{P}^+_{k+1,\ell-1},Z^0_\ell,P^0_{\ell-1},R^+_{k,\ell-1},R^-_{k+1,\ell}). \tag{3.46}$$

Furthermore, $(P^0_{\ell-1},P^+_{k\ell-1})$ *and* $Q^-_{k\ell}$ *are independent.*

Proof. This is a direct application of Theorem 3 from [98] to the iterations of Algorithm 2. The convergence result in [98] requires the uniform Lipschitz continuity of functions $\mathbf{g}^\pm_\ell(\cdot)$. Assumption 1 provides the required uniform Lipschitz continuity assumption on $\mathbf{g}^+_0(\cdot)$ and $\mathbf{g}^-_3(\cdot)$. For the "middle" layers, $\ell=1,2$, the denoisers $\mathbf{g}^\pm_\ell(\cdot)$ are linear and the uniform continuity assumption is valid since we have made the additional assumption that the terms $\mathbf{s}_{\text{tr}}$ and $\mathbf{s}_{\text{mp}}$ are bounded almost surely. □

A key use of the Theorem is to compute asymptotic empirical limits. Specifically, for a componentwise function $\psi(\cdot)$, let $\langle\psi(\mathbf{x})\rangle$ denotes the average $\frac{1}{N}\sum_{n=1}^N\psi(x_n)$ The above theorem then states that for any componentwise pseudo-Lipschitz function $\psi(\cdot)$ of order 2, as

$N \to \infty$, we have the following two properties

$$\begin{aligned}
&\lim_{N\to\infty} \left\langle \psi(\widehat{\mathbf{z}}^+_{k\ell}, \mathbf{z}^0_\ell, \mathbf{p}^0_{\ell-1}, \mathbf{r}^+_{k,\ell-1}, \mathbf{r}^-_\ell) \right\rangle \\
&\qquad = \mathbb{E}\, \psi(\widehat{Z}^+_{k\ell}, Z^0_\ell, P^0_{\ell-1}, R^+_{k,\ell-1}, R^-_\ell), \\
&\lim_{N\to\infty} \left\langle \psi(\widehat{\mathbf{p}}^+_{k+1,\ell-1}, \mathbf{z}^0_\ell, \mathbf{p}^0_{\ell-1}, \mathbf{r}^+_{k,\ell-1}, \mathbf{r}^-_{k+1,\ell}) \right\rangle \\
&\qquad = \mathbb{E}\, \psi(\widehat{P}^+_{k+1,\ell-1}, Z^0_\ell, P^0_{\ell-1}, R^+_{k,\ell-1}, R^-_{k+1,\ell}).
\end{aligned}$$

That is, we can compute the empirical average over components with the expected value of the random variable limit. This convergence is key to proving Theorem 1.

3.7 Special Cases

In this section, we provide a few special cases for calculating the generalization error of the GLM problem (3.2).

3.7.1 Linear Output with Square Error

We first consider a linear output with additive Gaussian noise and a squared error training and test loss. Specifically, consider the model,

$$\mathbf{y} = \mathbf{X}\mathbf{w}^0 + \mathbf{d}. \tag{3.47}$$

We consider estimates of $\mathbf{w}^0$ such that:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \ \tfrac{1}{2} \left\|\mathbf{y} - \mathbf{X}\mathbf{w}\right\|^2 + \tfrac{\lambda}{2\beta} \left\|\mathbf{w}\right\|^2. \tag{3.48}$$

The factor β is added above since the two terms scale with a ratio of β. It does not change analysis. Consider the ML-VAMP GLM learning algorithm applied to this problem. The following corollary follows from the main result in Theorem 1.

Corollary 1 (Squared error)**.** *For linear regression,* i.e., $\phi(t) = t$, $\phi_{\text{out}}(t,d) = t + d$, $f_{\text{ts}}(y,\widehat{y}) = (y_{\text{ts}} - \widehat{y}_{\text{ts}})^2$, $F_{\text{out}}(\mathbf{p}_2) = \frac{1}{N}\|\boldsymbol{y} - \mathbf{p}_2\|^2$, *we have*

$$\mathcal{E}_{\text{ts}}^{\mathsf{LR}} = \mathbb{E}\left(\frac{\mathbf{w}\gamma_0^+ S_{\text{ts}}}{\mathbf{w}\gamma_0^+ + S_{\text{tr}}^2 \mathbf{w}\gamma_1^-}\right)^2 k_{22} + \mathbb{E}\left(\frac{\mathbf{w}\gamma_1^- S_{\text{tr}} S_{\text{ts}}}{\mathbf{w}\gamma_0^+ + S_{\text{tr}}^2 \mathbf{w}\gamma_1^-}\right)^2 \tau_1^- + \sigma_d^2.$$

The quantities k_{22}, $\tau_1^-, \overline{\gamma}_0^+, \overline{\gamma}_1^-$ *depend on the choice of regularizer* λ *and the covariance between features.*

Proof. See Appendix B.3. □

3.7.2 Ridge Regression with i.i.d. Covariates

We next look at the special case when the input features are independent, i.e., (3.48) where rows of $\mathbf{X}$ corresponding to the training data has i.i.d Gaussian features with covariance $\mathbf{P}_{\text{train}} = \frac{\sigma_{\text{tr}}^2}{p}\mathbf{I}$ and $S_{\text{tr}} = \sigma_{\text{tr}}$.

Although the solution to (3.48) exists in closed form $(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$, we can study the effect of the regularization parameter λ on the generalization error $\mathcal{E}_{\text{ts}}$ as detailed in the result below.

Corollary 2. *Consider the ridge regression problem* (3.48) *with regularization parameter* $\lambda > 0$. *For the squared loss* i.e., $f_{\text{ts}}(y,\widehat{y}) = (y - \widehat{y})^2$, *i.i.d Gaussian features without train-test mismatch,* i.e., $S_{\text{tr}} = S_{\text{ts}} = \sigma_{\text{tr}}$, *the generalization error* $\mathcal{E}_{\text{ts}}^{\mathsf{RR}}$ *is given by Corollary 1, with constants*

$$k_{22} = \text{Var}(W^0), \qquad \overline{\gamma}_0^+ = \lambda/\beta,$$

$$\overline{\gamma}_1^- = \begin{cases} \frac{1}{G} - \frac{\lambda}{\sigma_{\text{tr}}^2} & \beta < 1 \\ \frac{\lambda}{\sigma_{\text{tr}}^2\beta}\left(\frac{1}{G} - \frac{\lambda}{\sigma_{\text{tr}}^2\beta}\right) & \\ \frac{\beta - 1}{G} + \frac{\lambda}{\sigma_{\text{tr}}^2\beta} & \beta > 1 \end{cases},$$

where $G = G_{\text{mp}}(-\frac{\lambda}{\sigma_{\text{tr}}^2\beta})$, *with* G_{mp} *given in Section 2.2.1, and* $\tau_1^- = \mathbb{E}(P_1^-)^2$ *where* P_1^- *is given in equation* (B.39) *in the proof in the Appendix B.3.*

Proof. See Appendix B.3. □

3.7.3 Ridgeless Linear Regression

Here we consider the case of Ridge regression (3.48) when $\lambda \to 0^+$. Note that the solution to the problem (3.48) is $(\mathbf{X}^\mathsf{T}\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$ remains unique since $\lambda > 0$. The following result was stated in [59], and can be recovered using our methodology. Note however, that we calculate the generalization error whereas they have calculated the squared error, whereby we obtain an additional additive factor of σ_d^2. The result explains the double-descent phenomenon for Ridgeless linear regression.

Corollary 3. *For ridgeless linear regression, we have*

$$\lim_{\lambda\to 0^+} \mathcal{E}_{\text{ts}}^{\mathsf{RR}} = \begin{cases} \frac{1}{1-\beta}\sigma_d^2 & \beta < 1 \\ \frac{\beta}{\beta-1}\sigma_d^2 + (1-\frac{1}{\beta})\sigma_{\text{tr}}^2 \text{Var}(W^0) & \beta \geq 1 \end{cases}$$

Proof. See Appendix B.3 □

3.7.4 Train-Test Mismatch

Observe that our formulation allows for analyzing the effect of mismatch in the training and test distribution. One can consider arbitrary joint distributions over $(S_{\text{tr}}, S_{\text{ts}})$ that model the mismatch between training and test features. Here we give a simple example which highlights the effect of this mismatch.

Definition 1 (Bernoulli ε-mismatch)**.** $(S_{\text{ts}}, S_{\text{tr}})$ has a bivariate Bernoulli distribution with

- $\Pr\{S_{\text{tr}}=S_{\text{ts}}=0\} = \mathbb{P}\{S_{\text{tr}}=S_{\text{ts}}=1\} = (1-\varepsilon)/2$
- $\Pr\{S_{\text{tr}}=0, S_{\text{ts}}=1\} = \mathbb{P}\{S_{\text{tr}}=1, S_{\text{ts}}=0\} = \varepsilon/2$

Notice that the marginal distribution of the S_{tr} in the Bernoulli $\varepsilon-$mismatch model is such that $\mathbb{P}(S_{\text{tr}} \neq 0) = \frac{1}{2}$. Hence half of the features extracted by the matrix V_0 are relevant

during training. Similarly, $\mathbb{P}(S_{\text{ts}} \neq 0) = \frac{1}{2}$. However the features spanned by the test data do not exactly overlap with the features captured in the training data, and the fraction of features common to both the training and test data is $1 - \varepsilon$. Hence for $\varepsilon = 0$, there is no training-test mismatch, whereas for $\varepsilon = 1$ there is a complete mismatch.

The following result shows that the generalization error increases linearly with the mismatch parameter ε.

Corollary 4 (Mismatch)**.** *Consider the problem of Linear Regression* (3.48) *under the conditions of Corollary 1. Additionally suppose we have Bernoulli ε-mismatch between the training and test distributions. Then*

$$\mathcal{E}_{\text{ts}} = \tfrac{k_{22}}{2}((1-\varepsilon)\gamma^{*2} + \varepsilon) + \tfrac{\tau_1^-}{2}(1-\gamma^*)(1-\varepsilon) + \sigma_d^2,$$

where $\gamma^* := \frac{\mathbf{w}\gamma_0^+}{\mathbf{w}\gamma_0^+ + \mathbf{w}\gamma_1^-}$. *The terms $k_{22}, \tau_1^-, \gamma^*$ are independent of ε.*

Proof. This follows directly by calculating the expectations of the terms in Corollary 1, with the joint distribution of $(S_{\text{tr}}, S_{\text{ts}})$ given in Definition 1. □

The quantities k_{22} and τ_1^- in the result above can be calculated similar to the derivation in the proof of Corollary 2 and can in general depend on the regularization parameter λ and overparameterization parameter β.

3.7.5 Logistic Regression

The precise analysis for the special case of regularized logistic regression estimator with i.i.d Gaussian features is provided in [110]. Consider the logistic regression model,

$$\mathbb{P}(y_i = 1|\mathbf{x}_i) := \rho(\mathbf{x}_i^\mathsf{T}\mathbf{w}) \quad \text{for } i = 1, \cdots, N$$

where $\rho(x) = \frac{1}{1+e^{-x}}$ is the standard logistic function.

In this problem we consider estimates of $\mathbf{w}^0$ such that

$$\widehat{\mathbf{w}} := \underset{\mathbf{w}}{\operatorname{argmin}} \mathbf{1}^{\mathsf{T}} \log(1 + e^{\mathbf{X}\mathbf{w}}) - \mathbf{y}^{\mathsf{T}}\mathbf{X}\mathbf{w} + F_{\text{in}}(\mathbf{w}).$$

where F_{in} is the reguralization function. This is a special case of optimization problem (3.2) where

$$F_{\text{out}}(\mathbf{y}, \mathbf{X}\mathbf{w}) = \mathbf{1}^{\mathsf{T}} \log(1 + e^{\mathbf{X}\mathbf{w}}) - \mathbf{y}^{\mathsf{T}}\mathbf{X}\mathbf{w}. \tag{3.49}$$

Similar to the linear regression model, using the ML-VAMP GLM learning algorithm, we can characterize the generalization error for this model with quantities $\mathbf{K}_0^+, \tau_1^-, \overline{\gamma}_0^+, \overline{\gamma}_1^-$ given by algorithm 3. We note that in this case, the output non-linearity is

$$\phi_{\text{out}}(p_2, d) = \mathbb{1}_{\{\rho(p_2) > d\}} \tag{3.50}$$

where $d \sim \text{Unif}(0, 1)$. Also, the denoisers g_0^+, and g_3^- can be derived as the proximal operators of F_{in}, and F_{out} defined in (3.27).

3.7.6 Support Vector Machines

The asymptotic generalization error for support vector machine (SVM) is provided in [36]. Our model can also handle SVMs. Similar to logistic regression, SVM finds a linear classifier using the hinge loss instead of logistic loss. Assuming the class labels are $y = \pm 1$ the hinge loss is

$$\ell_{\text{hinge}}(y, \widehat{y}) = \max(0, 1 - y\widehat{y}). \tag{3.51}$$

Therefore, if we take

$$F_{\text{out}}(\mathbf{y}, \mathbf{X}\mathbf{w}) = \sum_i \max(0, 1 - y_i \mathbf{X}_i \mathbf{w}), \tag{3.52}$$

where $\mathbf{X}_i$ is the i^{th} row of the data matrix, the ML-VAMP algorithm for GLMs finds the SVM classifier. The algorithm would have proximal map of hinge loss and our theory provides

exact predictions for the estimation and prediction error of SVM.

As with all other models considered in this work, the true underlying data generating model could be anything that can be represented by the graphical model in Figure 3.1, e.g. logistic or probit model, and our theory is able to exactly predict the error when SVM is applied to learn such linear classifiers in the large system limit.

3.8 Numerical Experiments

Training and Test Distributions. Our theoretical results are validated on a number of synthetic data experiments. For all the experiments, the training and test data is generated following the model in Section 3.2. We generate the training and test eigenvalues as i.i.d. with lognormal distributions,

$$S_{\text{tr}}^2 = A(10)^{0.1u_{\text{tr}}}, \quad S_{\text{ts}}^2 = A(10)^{0.1u_{\text{ts}}},$$

where $(u_{\text{tr}}, u_{\text{ts}})$ are bivariate zero-mean Gaussian with

$$\text{cov}(u_{\text{tr}}, u_{\text{ts}}) = \sigma_u^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}.$$

In the case when $\sigma_u^2 = 0$, we obtain eigenvalues that are equal, corresponding to the i.i.d. case. With $\sigma_u^2 > 0$ we can model correlated features. Also, when the correlation coefficient $\rho = 1$, $S_{\text{tr}} = S_{\text{ts}}$, so there is no training and test mismatch. However, we can also select $\rho < 1$ to experiment with cases when the training and test distributions differ. In the examples below, we consider the following three cases:

(1) i.i.d. features ($\sigma_u = 0$);

(2) correlated features with matching training and test distributions ($\sigma_u = 3$ dB, $\rho = 1$); and

(3) correlated features with train-test mismatch ($\sigma_u = 3$ dB, $\rho = 0.5$).

For all experiments below, the true model coefficients are generated as i.i.d. Gaussian $w_j^0 \sim \mathcal{N}(0,1)$ and we use standard L_2-regularization, $f_{\text{in}}(w) = \lambda w^2/2$ for some $\lambda > 0$. Our framework can incorporate arbitrary i.i.d. distributions on w_j and regularizers, but we will illustrate just the Gaussian case with L_2-regularization here.

Under-regularized linear regression. We first consider the case of under-regularized linear regression where the output channel is $\phi_{\text{out}}(p,d) = p + d$ with $d \sim \mathcal{N}(0,\sigma_d^2)$. The noise variance σ_d^2 is set for an SNR level of 10 dB. We use a standard mean-square error (MSE) output loss, $f_{\text{out}}(y,p) = (y-p)^2/(2\sigma_d^2)$. Since we are using the L_2-regularizer, $f_{\text{in}}(w) = \lambda w^2/2$, the minimization (3.2) is standard ridge regression. Moreover, if we were to select $\lambda = 1/\mathbb{E}(w_j^0)^2$, then the ridge regression estimate would correspond to the minimum mean-squared error (MMSE) estimate of the coefficients $\mathbf{w}^0$. However, to study the under-regularized regime, we take $\lambda = 10^{-4}/\mathbb{E}(w_j^0)^2$.

Fig. 3.2 plots the test MSE for the three cases described above for the linear model. In the figure, we take $p = 1000$ features and vary the number of samples n from $0.2p$ (over-parameterized) to $3p$ (under-parameterized). For each value of n, we take 100 random instances of the model and compute the ridge regression estimate using the sklearn package and measure the test MSE on the 1000 independent test samples. The simulated values in Fig. 3.2 are the median test error over the 100 random trials. The test MSE is plotted in a normalized dB scale,

$$\text{Test MSE (dB)} = 10\log_{10}\left(\frac{\mathbb{E}(\widehat{y}_{\text{ts}} - y_{\text{ts}})^2}{\mathbb{E}y_{\text{ts}}^2}\right).$$

Also plotted is the state evolution (SE) theoretical test MSE from Theorem 1.

In all three cases in Fig. 3.2, the SE theory exactly matches the simulated values for the test MSE. Note that the case of match training and test distributions for this problem was studied in [59, 82, 88] and we see the *double descent* phenomenon described in their work.

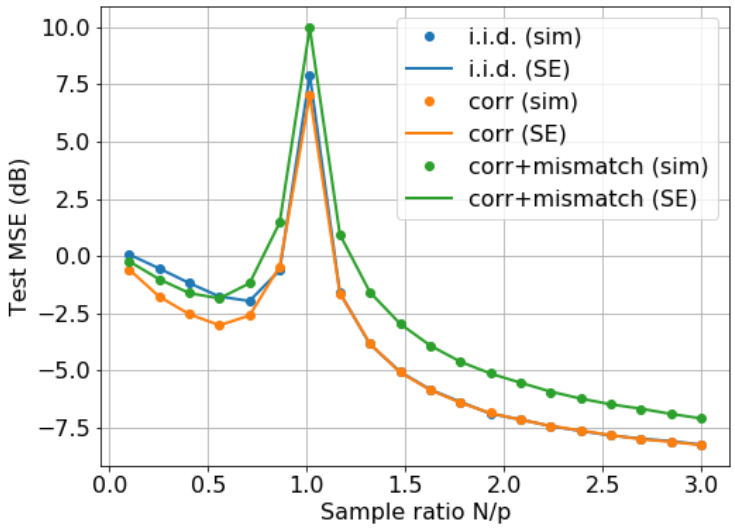

Figure 3.2: Test error for under-regularized linear regression under various train and test distributions. Noise variance σ_d^2 is set to SNR = 10 dB. The number of features $p = 1000$, and the number of samples n vary from $0.2\times p$ (over-parameterized) to $3\times p$ (under-parameterized). The experiment is averaged over 100 random instances of the model for each n and test MSE is calculated with 1000 independent test samples.

Specifically, with highly under-regularized linear regression, the test MSE actually *increases* with more samples n in the over-parametrized regime ($n/p < 1$) and then decreases again in the under-parametrized regime ($n/p > 1$).

Our SE theory can also provide predictions for the correlated feature case. In this particular setting, we see that in the correlated case the test error is slightly lower in the over-parametrized regime since the energy of data is concentrated in a smaller sub-space. Interestingly, there is minimal difference between the correlated and i.i.d. cases for the under-parametrized regime when the training and test data match. When the training and test data are not matched, the test error increases. In all cases, the SE theory can accurately predict these effects.

Logistic Regression. Fig. 3.3 shows a similar plot as Fig. 3.2 for a logistic model. Specifically, we use a logistic output $P(y = 1) = 1/(1 + e^{-p})$, a binary cross entropy output loss $f_{\text{out}}(y, p)$, and ℓ_2-regularization level λ, so that the output corresponds to the MAP estimate

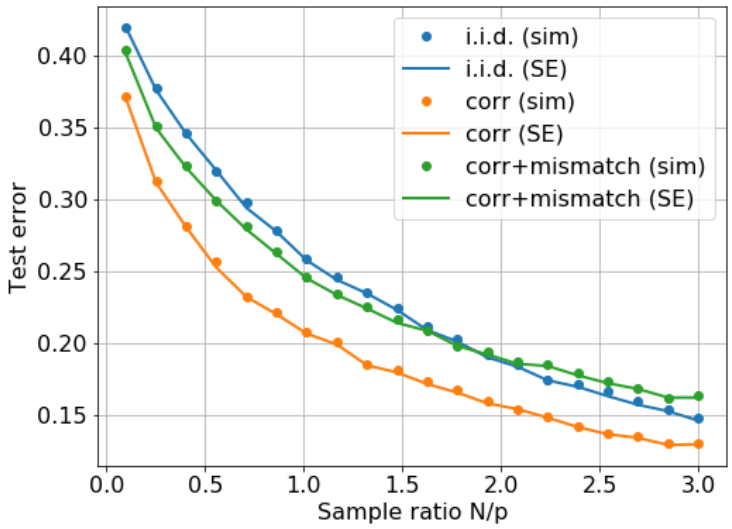

Figure 3.3: Classification error rate for logistic regression under various train and test distributions. F_{out} is the Binary cross-entropy loss, and F_{in} is the ridge penalty. The median error rate (1- accuracy) is estimated from 1000 new test samples.

(we do not perform ridgeless regression in this case). Other values of λ would correspond to M-estimators with a mismatched prior.

The mean of the training and test eigenvalues $\mathbb{E}S_{\rm tr}^2 = \mathbb{E}S_{\rm ts}^2$ are selected such that, if the true coefficients $\mathbf{w}^0$ were known, we could obtain a 5% prediction error. As in the linear case, we generate random instances of the model, use the sklearn package to perform the logistic regression, and evaluate the estimates on 1000 new test samples. We compute the median error rate (1− accuracy) and compare the simulated values with the SE theoretical estimates. The i.i.d. case was considered in [110]. Fig. 3.3 shows that our SE theory is able to predict the test error rate exactly in i.i.d. cases along with a correlated case and a case with training and test mismatch.

Nonlinear Regression. The SE framework can also consider non-convex problems. As an example, we consider a non-linear regression problem where the output function is

$$\phi_{\rm out}(p, d) = \tanh(p) + d, \quad d \sim \mathcal{N}(0, \sigma_d^2). \tag{3.53}$$

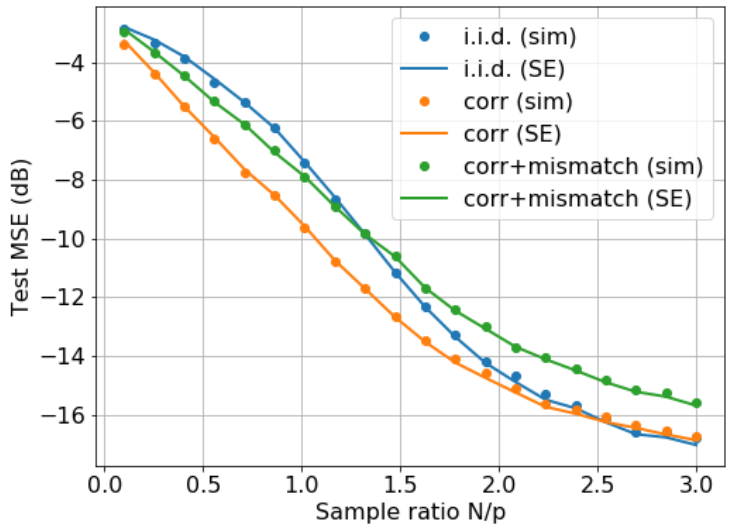

Figure 3.4: Test MSE under a non-linear least square estimation. The $\tanh(\cdot)$ output function is used with ℓ_2-regularization. Noise variance $\sigma_d^2 = 0.01$. The ADAM optimizer is used for simulations.

The $\tanh(p)$ models saturation in the output. Corresponding to this output, we use a non-linear MSE output loss

$$f_{\text{out}}(y - p) = \frac{1}{2\sigma_d^2}(y - \tanh(p))^2. \tag{3.54}$$

This output loss is non-convex. The data is generated as in the previous experiments and we scale the data matrix so that the input $\mathbb{E}(p^2) = 9$ so that the $\tanh(p)$ is driven well into the non-linear regime. We also take $\sigma_d^2 = 0.01$.

For the simulation, the non-convex loss is minimized using TensorFlow, where the non-linear model is described as a two-layer model. We use the ADAM optimizer [68] with 200 epochs to approach a local minimum of the objective (3.2). Fig. 3.4 plots the median test MSE for the estimate along with the SE theoretical test MSE. We again see that the SE theory is able to predict the test MSE in all cases even for this non-convex problem. Note that Figures 3.3 and 3.4 do not show a double descent because we apply regularization in those experiments.

3.9 Summary

In this chapter we provided a procedure for exactly computing the asymptotic generalization error of a solution in a generalized linear model (GLM). This procedure is based on scalar quantities which are fixed points of a recursive iteration. The formula holds for a large class of generalization metrics, loss functions, and regularization schemes. Our formula allows analysis of important modeling effects such as over-parameterization, dependence between covariates, and mismatch between train and test distributions, which play a significant role in the analysis and design of machine learning systems. We experimentally validated our theoretical results for linear as well as non-linear regression and logistic regression, where a strong agreement is seen between our formula and simulated results.

Chapter 4

Input-Output Equivalence of Unitary and Contractive Recurrent Neural Networks

[1] Unitary recurrent neural networks (URNNs) have been proposed as a method to overcome the vanishing and exploding gradient problem in modeling data with long-term dependencies. A basic question is how restrictive the unitary constraint is on the possible input-output mappings of such a network. This chapter shows that for any contractive RNN with ReLU activations, there is a URNN with at most twice the number of hidden states and the identical input-output mapping. Hence, with ReLU activations, URNNs are as expressive as general RNNs. In contrast, for certain smooth activations, we show that the input-output mapping of an RNN cannot be matched with a URNN, even with an arbitrary number of states. The theoretical results are supported by experiments on modeling of slowly-varying dynamical systems.

[1] This chapter is based on the work [46] coauthored with Mojtaba Sahraee-Ardakan, Sundeep Rangan and Alyson K. Fletcher.

4.1 Introduction

Recurrent neural networks (RNNs) – originally proposed in the late 1980s [45, 109] – refer to a widely-used and powerful class of models for time series and sequential data. In recent years, RNNs have become particularly important in speech recognition [58, 61] and natural language processing [9, 32, 119] tasks.

A well-known challenge in training recurrent neural networks is the *vanishing* and *exploding* gradient problem [20, 100]. RNNs have a transition matrix that maps the hidden state at one time to the next time. When the transition matrix has an induced norm greater than one, the RNN may become unstable. In this case, small perturbations of the input at some time can result in a change in the output that grows exponentially over the subsequent time. This instability leads to a so-called exploding gradient. Conversely, when the norm is less than one, perturbations can decay exponentially so inputs at one time have negligible effect in the distant future. As a result, the loss surface associated with RNNs can have steep walls that may be difficult to minimize. Such problems are particularly acute in systems with long-term dependencies, where the output sequence can depend strongly on the input sequence many time steps in the past.

Unitary RNNs (URNNs) [6] is a simple and commonly-used approach to mitigate the vanishing and exploding gradient problem. The basic idea is to restrict the transition matrix to be unitary (an orthogonal matrix for the real-valued case). The unitary transitional matrix is then combined with a non-expansive activation such as a ReLU or sigmoid. As a result, the overall transition mapping cannot amplify the hidden states, thereby eliminating the exploding gradient problem. In addition, since all the singular values of a unitary matrix equal 1, the transition matrix does not attenuate the hidden state, potentially mitigating the vanishing gradient problem as well. (Due to activation, the hidden state may still be attenuated). Some early work in URNNs suggested that they could be more effective than other methods, such as long short-term memory (LSTM) architectures and standard RNNs, for certain learning tasks involving long-term dependencies [6, 65] – see a short summary

below.

Although URNNs may improve the stability of the network for the purpose of optimization, a basic issue with URNNs is that the unitary constraint may potentially reduce the set of input-output mappings that the network can model. Here, we seeks to rigorously characterize how restrictive the unitary constraint is on an RNN. We evaluate this restriction by comparing the set of input-output mappings achievable with URNNs with the set of mappings from all RNNs. As described below, we restrict our attention to RNNs that are contractive in order to avoid unstable systems.

4.1.1 Key Contributions

We show that, given any contractive RNN with n hidden states and ReLU activations, there exists a URNN with at most $2n$ hidden states and the identical input-ouput mapping. This result is tight in the sense that, given any $n > 0$, there exists at least one contractive RNN such that any URNN with the same input-output mapping must have at least $2n$ states. We also demonstrate that the equivalence of URNNs and RNNs depends on the activation. For example, we prove that there exists a contractive RNN with *sigmoid* activations such that there is no URNN with any finite number of states that exactly matches the input-output mapping. The implication of this result is that, for RNNs with ReLU activations, there is no loss in the *expressiveness* of model when imposing the unitary constraint. As we discuss later, the penalty is a two-fold increase in the number of parameters.

Of course, the expressiveness of a class of models is only one factor in their real performance. Based on these results alone, one cannot determine if URNNs will outperform RNNs in any particular task. Earlier works have found examples where URNNs offer some benefits over LSTMs and RNNs [6, 128]. But in our simulations concerning modeling slowly-varying nonlinear dynamical systems, we see that URNNs with $2n$ states perform approximately equally to RNNs with n states.

4.1.2 Prior Work

The vanishing and exploding gradient problem in RNNs has been known almost as early as RNNs themselves [20, 100]. It is part of a larger problem of training models that can capture long-term dependencies, and several proposed methods address this issue. Most approaches use some form of gate vectors to control the information flow inside the hidden states, the most widely-used being LSTM networks [62]. Other gated models include Highway networks [116] and gated recurrent units (GRUs) [30]. L_1/L_2 penalization on gradient norms and gradient clipping were proposed to solve the exploding gradient problem in [100]. With L_1/L_2 penalization, capturing long-term dependencies is still challenging since the regularization term quickly kills the information in the model. A more recent work [101] has successfully trained very deep networks by carefully adjusting the initial conditions to impose an approximate unitary structure of many layers.

Unitary evolution RNNs (URNNs) are a more recent approach first proposed in [6]. One of the technical difficulties is to efficiently parametrize the set of unitary matrices. The numerical simulations presented here focus on relatively small networks, where the parameterization is not a significant computational issue. Nevertheless, for larger numbers of hidden states, several approaches have been proposed. The model in [6] parametrizes the transition matrix as a product of reflection, diagonal, permutation, and Fourier transform matrices. This model spans a subspace of the whole unitary space, thereby limiting the expressive power of RNNs. The work [128] overcomes this issue by optimizing over full-capacity unitary matrices. A key limitation this result, however, is that the projection of weights on to the unitary space is not computationally efficient. A tunable, efficient parametrization of unitary matrices is proposed in [65]. This model provides the computational complexity of $O(1)$ per parameter. The unitary matrix is represented as a product of rotation matrices and a diagonal matrix. By grouping specific rotation matrices, the model provides tunability of the span of the unitary space and enables using different capacities for different tasks. Combining the parametrization in [65] for unitary matrices and the "forget" ability of the GRU structure, [30, 64] presented

an architecture that outperforms conventional models in several long-term dependency tasks. Other methods such as orthogonal RNNs proposed by [84] showed that the unitary constraint is a special case of the orthogonal constraint. By representing an orthogonal matrix as a product of Householder reflectors, we are able span the entire space of orthogonal matrices. Imposing hard orthogonality constraints on the transition matrix limits the expressiveness of the model and speed of convergence and performance may degrade [126].

4.2 RNNs and Input-Output Equivalence

4.2.1 RNNs

We consider recurrent neural networks (RNNs) representing sequence-to-sequence mappings of the form

$$\mathbf{h}_t = \phi(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{F}\mathbf{x}_t + \mathbf{b}), \quad \mathbf{h}_{(-1)} = \mathbf{h}_{-1}, \tag{4.1a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{h}_t, \tag{4.1b}$$

parameterized by $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{F}, \mathbf{b}, \mathbf{C}, \mathbf{h}_{-1})$. The system is shown in Fig. 4.1. The system maps a sequence of inputs $\mathbf{x}_t \in \mathbb{R}^m$, $t = 0, 1, \ldots, T-1$ to a sequence of outputs $\mathbf{y}_t \in \mathbb{R}^p$. In equation (4.1), ϕ is the activation function (e.g. sigmoid or ReLU); $\mathbf{h}_t \in \mathbb{R}^n$ is an internal or hidden state; $\mathbf{W} \in \mathbb{R}^{n \times n}, \mathbf{F} \in \mathbb{R}^{n \times m}$, and $\mathbf{C} \in \mathbb{R}^{p \times n}$ are the hidden-to-hidden, input-to-hidden, and hidden-to-output weight matrices respectively; and $\mathbf{b}$ is the bias vector. We have considered the initial condition, $\mathbf{h}_{-1}$, as part of the parameters, although we will often take $\mathbf{h}_{-1} = \mathbf{0}$. Given a set of parameters $\boldsymbol{\theta}$, we will let

$$\mathbf{y} = G(\mathbf{x}, \boldsymbol{\theta}) \tag{4.2}$$

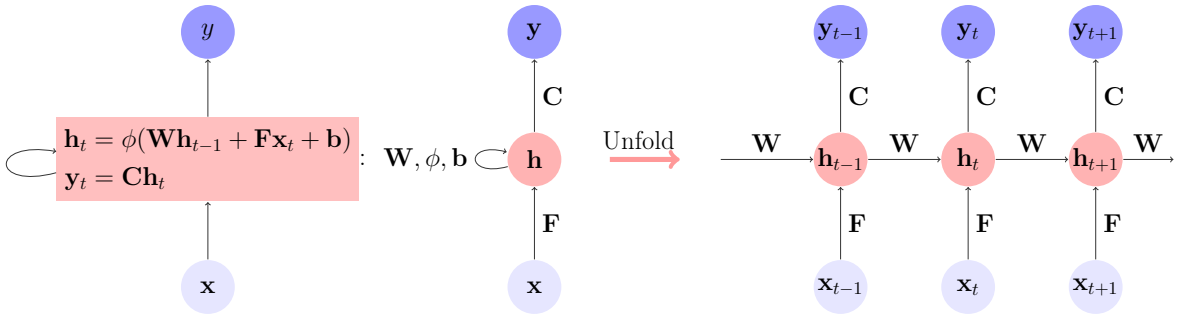

Figure 4.1: Recurrent Neural Networks representation

denote the resulting sequence-to-sequence mapping. Note that the number of time samples, T, is fixed throughout our discussion.

Recall [118] that a matrix $\mathbf{W}$ is *unitary* if $\mathbf{W}^{\mathsf{H}}\mathbf{W} = \mathbf{W}\mathbf{W}^{\mathsf{H}} = \mathbf{I}$. When a unitary matrix is real-valued, it is also called *orthogonal*. In this work, we will restrict our attention to real-valued matrices, but still use the term unitary for consistency with the URNN literature. A *Unitary RNN* or URNN is simply an RNN (4.1) with a unitary state-to-state transition matrix $\mathbf{W}$. A key property of unitary matrices is that they are *norm-preserving*, meaning that $\|\mathbf{W}\mathbf{h}_t\|_2 = \|\mathbf{h}_t\|_2$. In the context of (4.1a), the unitary constraint implies that the transition matrix does not amplify the state.

4.2.2 Equivalence of RNNs

Our goal is to understand the extent to which the unitary constraint in a URNN restricts the set of input-output mappings. To this end, we say that the RNNs for two parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are *input-output equivalent* if the sequence-to-sequence mappings are identical,

$$G(\mathbf{x}, \boldsymbol{\theta}_1) = G(\mathbf{x}, \boldsymbol{\theta}_2) \text{ for all } \mathbf{x} = (\boldsymbol{x}_{(0)}, \ldots, \boldsymbol{x}_{(T-1)}). \tag{4.3}$$

That is, for all input sequences $\mathbf{x}$, the two systems have the same output sequence. Note that the hidden internal states $\mathbf{h}_t$ in the two systems may be different. We will also say that two RNNs are *equivalent on a set of* $\mathcal{X}$ of inputs if (4.3) holds for all $\mathbf{x} \in \mathcal{X}$.

It is important to recognize that input-output equivalence does *not* imply that the parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are identical. For example, consider the case of linear RNNs where the activation in (4.1) is the identity, $\phi(\mathbf{z}) = \mathbf{z}$. Then, for any invertible S, the transformation

$$\mathbf{W} \to \mathbf{S}\mathbf{W}\mathbf{S}^{-1}, \quad \mathbf{C} \to \mathbf{C}\mathbf{S}^{-1}, \quad \mathbf{F} \to \mathbf{S}\mathbf{F}, \quad \mathbf{h}_{-1} \to \mathbf{S}\mathbf{h}_{-1}, \tag{4.4}$$

results in the same input-output mapping. However, the internal states $\mathbf{h}_t$ will be mapped to $\mathbf{S}\mathbf{h}_t$. The fact that many parameters can lead to identical input-output mappings will be key to finding equivalent RNNs and URNNs.

4.2.3 Contractive RNNs

The *spectral norm* [118] of a matrix $\mathbf{W}$ is the maximum gain of the matrix $\|\mathbf{W}\| := \max_{\mathbf{h}\neq\mathbf{0}} \frac{\|\mathbf{W}\mathbf{h}\|_2}{\|\mathbf{h}\|_2}$. In an RNN (4.1), the spectral norm $\|\mathbf{W}\|$ measures how much the transition matrix can amplify the hidden state. For URNNs, $\|\mathbf{W}\| = 1$. We will say an RNN is *contractive* if $\|\mathbf{W}\| < 1$, *expansive* if $\|\mathbf{W}\| > 1$, and *non-expansive* if $\|\mathbf{W}\| \leq 1$. In the sequel, we will restrict our attention to contractive and non-expansive RNNs. In general, given an expansive RNN, we cannot expect to find an equivalent URNN. For example, suppose $\mathbf{h}_t = h_t$ is scalar. Then, the transition matrix $\mathbf{W}$ is also scalar $\mathbf{W} = w$ and w is expansive if and only if $|w| > 1$. Now suppose the activation is a ReLU $\phi(h) = \max\{0, h\}$. Then, it is possible that a constant input $x_t = x_0$ can result in an output that grows exponentially with time: $y_t = \text{const} \times w^t$. Such an exponential increase is not possible with a URNN. We consider only non-expansive RNNs in the remainder of the chapter. Some of our results will also need the assumption that the activation function $\phi(\cdot)$ in (4.1) is non-expansive:

$$\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2, \qquad \text{for all } \mathbf{x} \text{ and } \mathbf{y}.$$

This property is satisfied by the two most common activations, sigmoids and ReLUs.

4.2.4 Equivalence of Linear RNNs.

To get an intuition of equivalence, it is useful to briefly review the concept in the case of linear systems [66]. Linear systems are RNNs (4.1) in the special case where the activation function is identity, $\phi(\mathbf{z}) = \mathbf{z}$; the initial condition is zero, $\mathbf{h}_{-1} = \mathbf{0}$; and the bias is zero, $\mathbf{b} = \mathbf{0}$. In this case, it is well-known that two systems are input-output equivalent if and only if they have the same *transfer function*,

$$H(s) := \mathbf{C}(s\mathbf{I} - \mathbf{W})^{-1}\mathbf{F}. \tag{4.5}$$

In the case of scalar inputs and outputs, $H(s)$ is a rational function of the complex variable s with numerator and denominator degree of at most n, the dimension of the hidden state $\mathbf{h}_t$. Any state-space system (4.1) that achieves a particular transfer function is called a *realization* of the transfer function. Hence two linear systems are equivalent if and only if they are the realizations of the same transfer function.

A realization is called *minimal* if it is not equivalent some linear system with fewer hidden states. A basic property of realizations of linear systems is that they are minimal if and only if they are *controllable* and *observable*. The formal definition is in any linear systems text, e.g. [66]. Loosely, controllable implies that all internal states can be reached with an appropriate input and observable implies that all hidden states can be observed from the ouptut. In absence of controllability and observability, some hidden states can be removed while maintaining input-output equivalence.

4.3 Equivalence Results for RNNs with ReLU Activations

Our first results consider contractive RNNs with ReLU activations. For the remainder of the section, we will restrict our attention to the case of zero initial conditions, $\mathbf{h}_{-1} = 0$ in (4.1).

Theorem 3. *Let* $\mathbf{y} = G(\mathbf{x}, \boldsymbol{\theta}_c)$ *be a contractive RNN with ReLU activation and states of*

dimension n. Fix $M > 0$ and let $\mathcal{X}$ be the set of all sequences such that $\|\mathbf{x}_t\|_2 \leq M < \infty$ for all t. Then there exists a URNN with state dimension $2n$ and parameters $\boldsymbol{\theta}_u = (\mathbf{W}_u, \mathbf{F}_u, \mathbf{b}_u, \mathbf{C}_u)$ such that for all $\mathbf{x} \in \mathcal{X}$, $G(\mathbf{x}, \boldsymbol{\theta}_c) = G(\mathbf{x}, \boldsymbol{\theta}_u)$. Hence the input-output mapping is matched for bounded inputs.

Proof. The basic idea is to construct a URNN with $2n$ states such that first n states match the states of RNN and the last n states are always zero. To this end, consider any contractive RNN,

$$\mathbf{h}_{c,t} = \phi(\mathbf{W}_c\ \mathbf{h}_{c,t-1} + \mathbf{F}_c\ \mathbf{x}_t + \mathbf{b}_c), \quad \mathbf{y}_t = \mathbf{C}_c\ \mathbf{h}_{c,t},$$

where $\mathbf{h}_t \in \mathbb{R}^n$. Note that the subscript c indicates the parameters/signals for the contractive network and u for unitary network. The second subscript, t, for the hidden states indicates the time index. Since $\mathbf{W}$ is contractive, we have $\|\mathbf{W}\| \leq \rho$ for some $\rho < 1$. Also, for a ReLU activation, $\|\phi(\mathbf{z})\| \leq \|\mathbf{z}\|$ for all pre-activation inputs $\mathbf{z}$. Hence,

$$\begin{aligned}\|\mathbf{h}_{c,t}\|_2 &= \|\phi(\mathbf{W}_c\mathbf{h}_{c,t-1} + \mathbf{F}_c\mathbf{x}_t + \mathbf{b}_c)\|_2 \leq \|\mathbf{W}_c\mathbf{h}_{c,t-1} + \mathbf{F}_c\mathbf{x}_t + \mathbf{b}_c\|_2 \\ &\leq \rho\|\mathbf{h}_{c,t-1}\|_2 + \|\mathbf{F}_c\|\|\mathbf{x}_t\|_2 + \|\mathbf{b}_c\|_2.\end{aligned}$$

Therefore, with bounded inputs, $\|\mathbf{x}_t\| \leq M$, we have the state is bounded,

$$\|\mathbf{h}_t\|_2 \leq \frac{1}{1-\rho}\left[\|\mathbf{F}_c\|M + \|\mathbf{b}_c\|_2\right] =: M_h. \tag{4.6}$$

We construct a URNN as,

$$\mathbf{h}_{u,t} = \phi(\mathbf{W}_u\ \mathbf{h}_{u,t-1} + \mathbf{F}_u\ \mathbf{x}_t + \mathbf{b}_u), \quad \mathbf{y}_t = \mathbf{C}_u\ \mathbf{h}_{u,t}$$

where the parameters are of the form,

$$\mathbf{h}_u = \begin{bmatrix}\mathbf{h}_1 \\ \mathbf{h}_2\end{bmatrix} \in \mathbb{R}^{2n}, \quad \mathbf{W}_u = \begin{bmatrix}\mathbf{W}_1, \mathbf{W}_2 \\ \mathbf{W}_3, \mathbf{W}_4\end{bmatrix}, \quad \mathbf{F}_u = \begin{bmatrix}\mathbf{F}_c \\ \mathbf{0}\end{bmatrix}, \quad \mathbf{b}_u = \begin{bmatrix}\mathbf{b}_c \\ \mathbf{b}_2\end{bmatrix}. \tag{4.7}$$

Let $\mathbf{W}_1 = \mathbf{W}_c$. Since $\|\mathbf{W}_c\| < 1$, we have $\mathbf{I} - \mathbf{W}_c^\mathsf{T}\mathbf{W}_c \succeq 0$. Therefore, there exists W_3 such that $\mathbf{W}_3^\mathsf{T}\mathbf{W}_3 = \mathbf{I} - \mathbf{W}_c^\mathsf{T}\mathbf{W}_c$. With this choice of $\mathbf{W}_3$, the first n columns of $\mathbf{W}_u$ are orthonormal. Let $\begin{bmatrix}\mathbf{W}_2 \\ \mathbf{W}_4\end{bmatrix}$ extend these to an orthonormal basis for $\mathbb{R}^{2n}$. Then, the matrix $\mathbf{W}_u$ will be orthonormal.

Next, let $\mathbf{b}_2 = -M_h\mathbf{1}_{n\times 1}$, where M_h is defined in (4.6). We show by induction that for all k,

$$\mathbf{h}_{1,t} = \mathbf{h}_{c,t}, \quad \mathbf{h}_{2,t} = \mathbf{0}. \tag{4.8}$$

If both systems are initialized at zero, (4.8) is satisfied at $t = -1$. Now, suppose this holds up to time $t-1$. Then,

$$\begin{aligned}\mathbf{h}_{1,t} &= \phi(\mathbf{W}_1\mathbf{h}_{1,t-1} + \mathbf{W}_2\mathbf{h}_{2,t-1} + \mathbf{F}_c\mathbf{x}_t + \mathbf{b}_c) \\ &= \phi(\mathbf{W}_1\mathbf{h}_{1,t-1} + \mathbf{F}_c\mathbf{x}_t + \mathbf{b}_c) = \mathbf{h}_{c,t},\end{aligned}$$

where we have used the induction hypothesis that $\mathbf{h}_{2,t-1} = \mathbf{0}$. For $\mathbf{h}_{2,t}$, note that

$$\|\mathbf{W}_3\mathbf{h}_{1,t-1}\|_\infty \leq \|\mathbf{W}_3\mathbf{h}_{1,t-1}\|_2 \leq \|\mathbf{h}_{1,t-1}\| \leq M_h, \tag{4.9}$$

where the last step follows from (4.6). Therefore,

$$\mathbf{W}_3\mathbf{h}_{1,t-1} + \mathbf{W}_4\mathbf{h}_{2,t-1} + \mathbf{b}_2 = \mathbf{W}_3\mathbf{h}_{1,t-1} - M\mathbf{1}_{n\times 1} \leq \mathbf{0}. \tag{4.10}$$

Hence with ReLU activation $\mathbf{h}_{2,t} = \phi(\mathbf{W}_3\mathbf{h}_{1,t-1} + \mathbf{W}_4\mathbf{h}_{2,t-1} + \mathbf{b}_2) = \mathbf{0}$. By induction, (4.8) holds for all t. Then, if we define $\mathbf{C}_u = [\mathbf{C}_c\ \mathbf{0}]$, we have the output of the URNN and RNN systems are identical

$$\mathbf{y}_{u,t} = \mathbf{C}_u\mathbf{h}_{u,t} = \mathbf{C}_c\mathbf{h}_{1,t} = \mathbf{y}_{c,t}.$$

This shows that the systems are equivalent. □

Theorem 3 shows that for any contractive RNN with ReLU activations, there exists a URNN with at most twice the number of hidden states and the identical input-output mapping. Thus, there is no loss in the set of input-output mappings with URNNs relative to general contractive RNNs on bounded inputs.

The penalty for using RNNs is the two-fold increase in state dimension, which in turn increases the number of parameters to be learned. We can estimate this increase in parameters as follows: The raw number of parameters for an RNN (4.1) with n hidden states, p outputs and m inputs is $n^2 + (p + m + 1)n$. However, for ReLU activations, the RNNs are equivalent under the transformations (4.4) using diagonal positive $\mathbf{S}$. Hence, the number of degrees of freedom of a general RNN is at most $d_{\text{rnn}} = n^2 + (p + m)n$. We can compare this value to a URNN with $2n$ hidden states. The set of $2n \times 2n$ unitary $\mathbf{W}$ has $2n(2n - 1)/2$ degrees of freedom [117]. Hence, the total degrees of freedom in a URNN with $2n$ states is at most $d_{\text{urnn}} = n(2n - 1) + 2n(p + m)$. We conclude that a URNN with $2n$ hidden states has slightly fewer than twice the number of parameters as an RNN with n hidden states. We next show a converse result.

Theorem 4. *For any state dimension n, there exists at least one contractive RNN with ReLU non-linearity such that if a URNN is equivalent to the RNN, it has at least $2n$ states.*

Proof. See Appendix C.1. □

The result shows that the $2n$ achievability bound in Theorem 3 is tight, at least in the worst case. In addition, the RNN constructed in the proof of Theorem 4 is not particularly pathological. We will show in our simulations in Section 4.5 that URNNs typically need twice the number of hidden states to achieve comparable modeling error as an RNN.

4.4 Equivalence Results for RNNs with Sigmoid Activations

Equivalence between RNNs and URNNs depends on the particular activation. Our next result shows that with sigmoid activations, URNNs are, in general, never exactly equivalent to RNNs, even with an arbitrary number of states.

We need the following technical definition: Consider an RNN (4.1) with a sigmoid activation $\phi(z) = 1/(1 + e^{-z})$. If $\mathbf{W}$ is non-expansive, then a simple application of the contraction mapping principle shows that for any constant input $\mathbf{x}_t = \mathbf{x}^*$, there is a fixed point in the hidden state $\mathbf{h}^* = \phi(\mathbf{W}\mathbf{h}^* + \mathbf{F}\mathbf{x}^* + \mathbf{b})$. We will say that the RNN is controllable and observable at $\mathbf{x}^*$ if the linearization of the RNN around $(\mathbf{x}^*, \mathbf{h}^*)$ is controllable and observable.

Theorem 5. *There exists a contractive RNN with sigmoid activation function ϕ with the following property: If a URNN is controllable and observable at any point $\mathbf{x}^*$, then the URNN cannot be equivalent to the RNN for inputs $\mathbf{x}$ in the neighborhood of $\mathbf{x}^*$.*

Proof. See Appendix C.2. □

The result provides a converse on equivalence: Contractive RNNs with sigmoid activations are not in general equivalent to URNNs, even if we allow the URNN to have an arbitrary number of hidden states. Of course, the approximation error between the URNN and RNN may go to zero as the URNN hidden dimension goes to infinity (e.g., similar to the approximation results in [52]). However, exact equivalence is not possible with sigmoid activations, unlike with ReLU activations. Thus, there is fundamental difference in equivalence for smooth and non-smooth activations.

4.5 Numerical Experiments

In this section, we numerically compare the modeling ability of RNNs and URNNs where the true system is a contractive RNN with long-term dependencies. Specifically, we generate data from multiple instances of a synthetic RNN where the parameters in (4.1) are randomly generated. For the true system, we use $m = 2$ input units, $p = 2$ output units, and $n = 4$ hidden units at each time step. The matrices $\mathbf{F}$, $\mathbf{C}$ and $\mathbf{b}$ are generated as i.i.d. Gaussians. We use a random transition matrix,

$$\mathbf{W} = \mathbf{I} - \epsilon \mathbf{A}^{\mathsf{T}} \mathbf{A} / \|\mathbf{A}\|^2, \tag{4.11}$$

where $\mathbf{A}$ is Gaussian i.i.d. matrix and ϵ is a small value, taken here to be $\epsilon = 0.01$. The matrix (4.11) will be contractive with singular values in $(1 - \epsilon, 1)$. By making ϵ small, the states of the system will vary slowly, hence creating long-term dependencies. In analogy with linear systems, the time constant will be approximately $1/\epsilon = 100$ time steps. We use ReLU activations. To avoid degenerate cases where the outputs are always zero, the biases $\mathbf{b}$ are adjusted to ensure that the each hidden state is on some target 60% of the time using a similar procedure as in [51].

The trials have $T = 1000$ time steps, which corresponds to 10 times the time constant $1/\epsilon = 100$ of the system. We added noise to the output of this system such that the signal-to-noise ratio (SNR) is 15 dB or 20 dB. In each trial, we generate 700 training samples and 300 test sequences from this system.

Given the input and the output data of this contractive RNN, we attempt to learn the system with: (i) standard RNNs, (ii) URNNs, and (iii) LSTMs. The hidden states in the model are varied in the range $n = [2, 4, 6, 8, 10, 12, 14]$, which include values both above and below the true number of hidden states $n_{\text{true}} = 4$. We used mean-squared error as the loss function. Optimization is performed using Adam [68] optimization with a batch size = 10 and learning rate = 0.01. All models are implemented in the Keras package in Tensorflow.

The experiments are done over 30 realizations of the original contractive system.

For the URNN learning, of all the proposed algorithms for enforcing the unitary constraints on transition matrices during training [6, 65, 84, 128], we chose to project the transition matrix on the full space of unitary matrices after each iteration using singular value decomposition (SVD). Although SVD requires $\mathcal{O}(n^3)$ computation for each projection, for our choices of hidden states it performed faster than the aforementioned methods.

Since we have training noise and since optimization algorithms can get stuck in local minima, we cannot expect "exact" equivalence between the learned model and true system as in the theorems. So, instead, we look at the test error as a measure of the closeness of the learned model to the true system. Figure 4.2 on the left shows the test R^2 for a Gaussian i.i.d. input and output with SNR = 20 dB for RNNs, URNNs, and LSTMs. The red dashed line corresponds to the optimal R^2 achievable at the given noise level.

Note that even though the true RNN has $n_{\text{true}} = 4$ hidden states, the RNN model does not obtain the optimal test R^2 at $n = 4$. This is not due to training noise, since the RNN is able to capture the full dynamics when we over-parametrize the system to $n \approx 8$ hidden states. The test error in the RNN at lower numbers of hidden states is likely due to the optimization being caught in a local minima.

What is important for this result though is to compare the URNN test error with that of the RNN. We observe that URNN requires approximately twice the number of hidden states to obtain the same test error as achieved by an RNN. To make this clear, the right plot shows the same performance data with number of states adjusted for URNN. Since our theory indicates that a URNN with $2n$ hidden states is as powerful as an RNN with n hidden states, we compare a URNN with $2n$ hidden units directly with an RNN with n hidden units. We call this the adjusted hidden units. We see that the URNN and RNN have similar test error when we appropriately scale the number of hidden units as predicted by the theory.

For completeness, the left plot in Figure 4.2 also shows the test error with an LSTM. It is important to note that the URNN has almost the same performance as an LSTM with

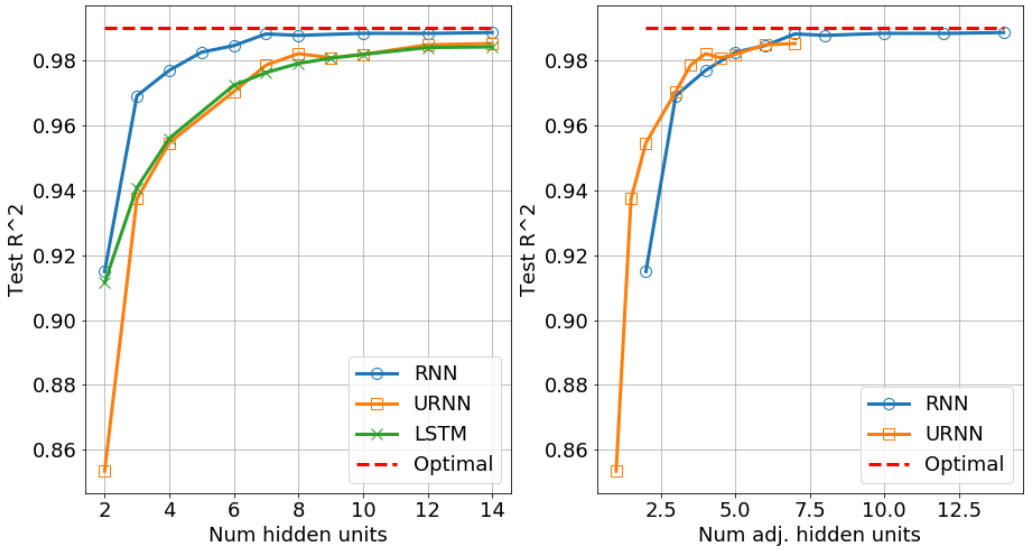

Figure 4.2: Test R^2 on synthetic data for a Gaussian i.i.d. input and output SNR=20 dB.

considerably smaller number of parameters.

Figure 4.3 shows similar results for the same task with SNR = 15 dB. For this task, the input is *sparse* Gaussian i.i.d., i.e. Gaussian with some probability $p = 0.02$ and 0 with probability $1 - p$. The left plot shows the R^2 vs. the number of hidden units for RNNs and URNNs and the right plot shows the same results once the number of hidden units for URNN is adjusted.

We also compared the modeling ability of URNNs and RNNs using the Pixel-Permuted MNIST task. Each MNIST image is a 28×28 grayscale image with a label between 0 and 9. A fixed random permutation is applied to the pixels and each pixel is fed to the network in each time step as the input and the output is the predicted label for each image [6, 65, 126].

We evaluated various models on the Pixel-Permuted MNIST task using validation based early stopping. Without imposing a contractivity constraint during learning, the RNN is either unstable or requires a slow learning rate. Imposing a contractivity constraint improves the performance. Incidentally, using a URNN improves the performance further. Thus, contractivity can improve learning for RNNs with sufficiently large numbers of time steps.

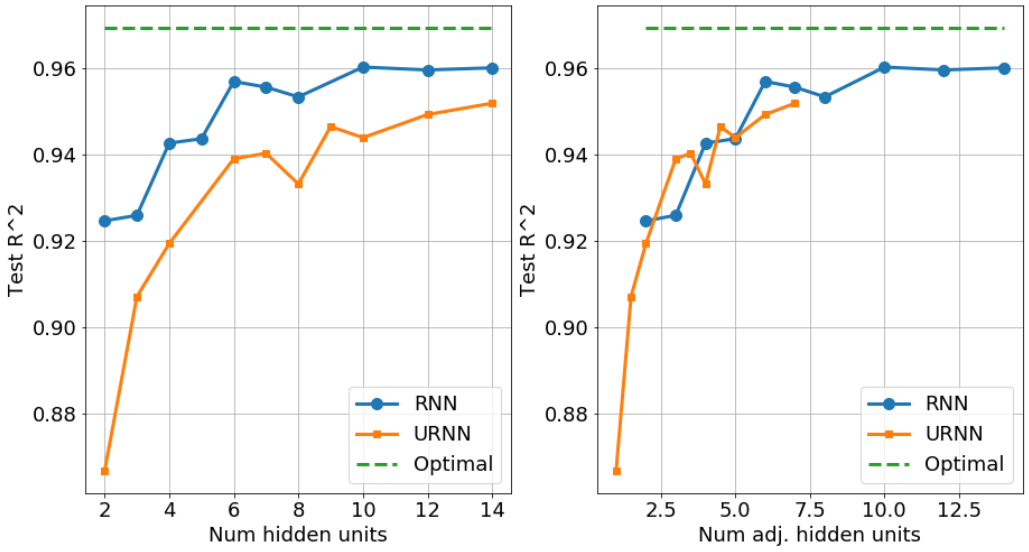

Figure 4.3: Test R^2 on sparse synthetic data for a Gaussian i.i.d. input and output SNR=15 dB.

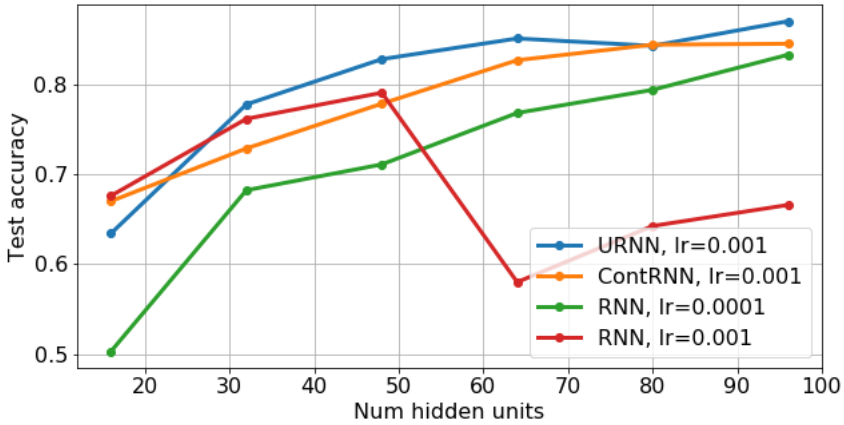

Figure 4.4: Accuracy on Permuted MNIST task for various models trained with RMSProp, validation-based early termination, and initial learning rate `lr`. (1) URNN model: RNN model with unitary constraint; (2) ContRNN: RNN with a contractivity constraint; (3 & 4) RNN model with no contractivity or unitary constraint (two learning rates). We see contractivity improves performance, and unitary constraints improve performance further.

4.6 Summary

Several works empirically show that using unitary recurrent neural networks improves the stability and performance of the RNNs. In this chapter, we studied how restrictive it is to use URNNs instead of RNNs. We showed that URNNs are at least as powerful as contractive RNNs in modeling input-output mappings if enough hidden units are used. More specifically, for any contractive RNN we explicitly constructed a URNN with twice the number of states of the RNN and identical input-output mapping. We also provided converse results for the number of states and the activation function needed for exact matching. We emphasize that although it has been shown that URNNs outperform standard RNNs and LSTM in many tasks that involve long-term dependencies, our main goal was to show that from an approximation viewpoint, URNNs are as expressive as general contractive RNNs. By a two-fold increase in the number of parameters, we can use the stability benefits they bring for optimization of neural networks.

Chapter 5

Implicit Bias of Recurrent Neural Networks

[1] In this chapter, we aim to understand the *implicit bias* behaviour of Recurrent Neural Networks (RNN). Several machine learning tasks require dealing with sequential data with possibly varying lengths of input sequences. Example tasks include automatic speech recognition, language translation, and image captioning, among others. A well-known shortcoming of standard RNNs trained using gradient descent is their poor performance at tasks requiring long-term dependence [21]. As a simple starting point, we seek to precisely understand the training and memory of *linear* RNNs. Even such linear models have been difficult to analyze completely due to their non-linear parameterization. Our main result shows rigorously that these models have an implicit bias toward short-term memory, confirming the qualitative empirically observed behavior of these networks.

Our results are based on the key observation that linear RNNs are functionally equivalent to a 1D-convolutional model which is feed-forward in nature. Further, due to the Neural Tangent Kernel (NTK) regime based analysis [63], we are able to show that RNNs trained using gradient descent are implicitly biased towards learning tasks with short-term contexts.

[1] Most parts of this chapter is based on the work [48] and is coauthered with Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K. Fletcher.

Our result holds in a certain wide limit regime where the number of hidden units in RNN goes to infinity.

5.1 Introduction

5.1.1 Key Contributions

We explicitly compute the NTK for a linear RNN. This is challenging due to the weight sharing in RNNs which leads to statistical dependencies across time. We calculate this NTK using a conditioning technique as in [16] to deal with the dependencies. This NTK is also calculated in [3] using the Tensor program results of [129]. Then, We show that the linear RNN NTK is equivalent to the NTK of a scaled convolutional model with certain scaling coefficients. This means that in the wide limit regime (number of hidden units in RNN $\to \infty$), gradient descent training of a linear RNN with non-linear parameterization is identical to the training of an appropriately scaled convolutional model. The above results rigorously show that there is an implicit bias in using the non-linear parameterization associated with a linear RNN. In particular, training linear RNNs with non-linear parameterization using gradient descent is implicitly biased towards short memory. Finally, We demonstrate the bias-variance trade-off of linear RNNs in experiments on synthetic and real data.

5.1.2 Prior Work

The connection between kernel methods and infinite width neural networks was first introduced in [92]. Neural networks in the infinite width limit are equivalent to Gaussian processes at initialization and several papers have investigated the correspondence to kernel methods for a variety of architectures [34, 54, 71, 81, 95, 129]. In particular, [34] introduced a framework to link a reproducing kernel to the neural network and stochastic gradient descent was shown to learn any function in the corresponding RKHS if the network is sufficiently wide [33].

A recent line of work has shown that gradient descent on over-parameterized networks

can achieve zero training error with parameters very close to their initialization [5, 42, 43, 74, 132]. The analysis of the generalization error in this high dimensional regime led to exact characterizations of the test error for different architectures [12, 17, 56, 59, 88]. In addition to convergence to a global minimum for an over-parameterized two-layer neural network, [44] also showed that the resulting functions are uniformly close to the ones found with the kernel regime. It was shown by [63] that the behavior of an infinitely wide fully-connected neural network trained by gradient descent is characterized by the so-called Neural Tangent Kernel (NTK) which is essentially the linearization of the network around its initialization. The NTK was later extended for different architectures [3, 7, 129, 130].

A different line of papers investigated the over-parameterized neural networks from the mean field viewpoint [40, 77, 83, 108, 114, 127]. For recurrent neural networks in particular, [28] has provided a theory for signal propagation in these networks which could predict their trainability. The authors also give a closed-form initialization to improve the conditioning of input-output Jacobian.

Note that in an RNN the states are correlated due to weight sharing. Previous work such as [28], has simplified the setting by assuming an independence over RNN weights (ignoring the correlation) to show that the pre-activations are Gaussian distributed. In this Chapter, taking into account these dependencies, we use techniques used in [16, 106] to characterize the behaviour of RNNs at initialization. Similar techniques have also been explored in [129].

We should mention that learning the weight matrices of a linear RNN using data is essentially a system identification task. There is a large body of literature in control theory that consider the system identification problem and propose many different methods to find a system that matches the input-output behavior of a given system. These methods include the prediction error method (PEM), subspace methods, empirical transfer function estimate (ETFE), correlation method, spectral analysis method, and sequential Monte Carlo method to name a few. For a more comprehensive list of system identification methods and details see [67, 73, 75, 76, 102, 115, 123] Even though it would be interesting to see how different

system identification methods can be incorporated into neural network training pipeline, the vast majority of works currently learn the weights directly by optimizing a loss function via gradient descent or its variants. As such, here we solely focus on training of linear RNNs using gradient descent.

5.2 Linear RNN and Convolutional Models

5.2.1 Linear RNNs

We fix a time period T and consider a linear RNN mapping an input sequence $\mathbf{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1})$ to an ouptut sequence $\mathbf{y} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{T-1})$ via the updates

$$\begin{aligned} \mathbf{h}_t &= \frac{1}{\sqrt{n}}\mathbf{W}\mathbf{h}_{t-1} + \mathbf{F}\boldsymbol{x}_t, \\ \boldsymbol{y}_t &= \frac{1}{\sqrt{n}}\mathbf{C}\mathbf{h}_t, \quad t = 0, \ldots, T-1, \end{aligned} \tag{5.1}$$

with the initial condition $\mathbf{h}_{-1} = \mathbf{0}$. We let n_x, n_y, and n be the dimension at each time of the input, $\boldsymbol{x}_t$, output, $\boldsymbol{y}_t$, and hidden state $\mathbf{h}_t$ respectively. Note that a bias term can be added for $\mathbf{h}_t$ by extending $\boldsymbol{x}_t$ and $\mathbf{F}$. We will let

$$\mathbf{y} = f_{\mathrm{RNN}}(\mathbf{x}, \boldsymbol{\theta}_{\mathrm{RNN}}) \tag{5.2}$$

denote the mapping (5.1) where θ_{RNN} are the parameters

$$\boldsymbol{\theta}_{\mathrm{RNN}} = (\mathbf{W}, \mathbf{F}, \mathbf{C}). \tag{5.3}$$

The goal is to learn parameters $\boldsymbol{\theta}_{\mathrm{RNN}}$ for the system from N training data samples $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, N$. In this case, each sample $(\mathbf{x}_i, \mathbf{y}_i)$ is a T-length input-output pair.

5.2.2 Wide System Limit

We wish to understand learning of this system in the *wide-system limit* where the number of hidden units $n \to \infty$ while the dimensions n_x, n_y and number of time steps T are fixed. Since the parameterization of the RNN is non-linear, the initialization is critical. For each n, we will assume that the parameters $\mathbf{W}, \mathbf{F}, \mathbf{C}$ are initialized with i.i.d. components,

$$W_{ij} \sim \mathcal{N}(0, \nu_W), \quad F_{ij} \sim N(0, \nu_F), \quad C_{ki} \sim N(0, \nu_C), \tag{5.4}$$

for constants ν_W, ν_F, ν_C.

5.2.3 Stability

In the initialization (5.4), ν_W is the variance of the components of the kernel matrix $\mathbf{W}$. One critical aspect in selecting ν_W is the *stability* of the system. A standard result in linear systems theory (see, e.g. [66]) is that the system (5.1) is stable if and only if $\frac{1}{\sqrt{n}}\lambda_{max}(\mathbf{W}) < 1$ where $\lambda_{\max}(\mathbf{W})$ is the maximum absolute eigenvalue of W (i.e. the spectral radius). Stable W are generally necessary for linear RNNs: Otherwise bounded inputs $\boldsymbol{x}_t$ can result in outputs $\boldsymbol{y}_t$ that grow unbounded with time t. Hence, training will be numerically unstable. Now, a classic result in random matrix theory [10] is that, since the entries of $\mathbf{W}$ are i.i.d. Gaussian $\mathcal{N}(0, \nu_W)$,

$$\lim_{n\to\infty} \frac{1}{\sqrt{n}}\lambda_{\max}(\mathbf{W}) = \nu_W$$

almost surely. Hence, for stability we need to select $\nu_W < 1$. As we will see below, it is this constraint that will limit the ability of the linear RNN to learn long-term memory.

5.2.4 Scaled 1D Convolutional Equivalent Systems:

Our main result will draw an equivalence between the learning of linear RNNs and certain types of linear convolutional models. Specifically, consider a *linear convolutional model* of

the form,

$$\boldsymbol{y}_t = \sum_{j=0}^{t} \mathbf{L}_j \boldsymbol{x}_{t-j}, \tag{5.5}$$

where $\mathbf{L}_j \in \mathbb{R}^{n_y \times n_x}$ are the filter coefficient matrices. In neural network terminology, the model (5.5) is a simply a linear 1D convolutional network with n_x input channels, n_y output channels and T-wide kernels.

Both the linear RNN model (5.1) and the 1D convolutional model (5.5) define linear mappings of T-length input sequences $\mathbf{x}$ to T-length output sequences $\mathbf{y}$. To state our equivalence result between these models, we need to introduce a certain *scaled parametrization*: Fix a set of scaling factors $\boldsymbol{\rho} = (\rho_0, \ldots, \rho_{T-1})$ where $\rho_j > 0$ for all j. Define the parameters

$$\boldsymbol{\theta}_{\text{conv}} = (\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{T-1}), \tag{5.6}$$

where $\boldsymbol{\theta}_j \in \mathbb{R}^{n_y \times n_x}$. Given any $\boldsymbol{\theta}$, let the impulse response coefficients be

$$\mathbf{L}_j = \sqrt{\rho_j}\boldsymbol{\theta}_j. \tag{5.7}$$

As we will see below, the effect of the weighting is to favor certain coefficients $\mathbf{L}_j$ over others during training: For coefficients j where ρ_j is large, the fitting will tend to select $\mathbf{L}_j$ large if needed. This scaling will be fundamental in understanding implicit bias.

Now, given a set of weights $\boldsymbol{\rho} = (\rho_0, \ldots, \rho_{T-1})$, let

$$\mathbf{y} = f_{\text{conv}}(\mathbf{x}, \boldsymbol{\theta}_{\text{conv}}) := \left\{ \sum_{j=0}^{t} \sqrt{\rho_j}\boldsymbol{\theta}_j \boldsymbol{x}_{t-j} \right\}_{t=0}^{T-1}, \tag{5.8}$$

denote the mapping of the input $\mathbf{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1})$ through the convolutional filter $\boldsymbol{\theta}_{\text{conv}}$ with filter coefficients ρ to produce the output sequence $\mathbf{y} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{T-1})$.

It is well-know that the RNN and convolutional models define the same total set of input-output mappings as given by the following standard result:

Proposition 1. *Consider the linear RNN model* (5.2) *and the 1D convolutional model* (5.8).

(a) *Given a linear RNN parameters* $\boldsymbol{\theta}_{\text{RNN}} = (\mathbf{W}, \mathbf{F}, \mathbf{C})$*, a 1D convolutional filter with coefficient matrices*

$$\mathbf{L}_j = \frac{1}{n^{\frac{j+1}{2}}} \mathbf{C}\mathbf{W}^j\mathbf{F}, \quad j = 0, \ldots, T-1, \tag{5.9}$$

will have an identical input-output mapping. That is, there exists parameters $\boldsymbol{\theta}_{\text{conv}}$ *such that*

$$f_{\text{RNN}}(\mathbf{x}, \boldsymbol{\theta}_{\text{RNN}}) = f_{\text{conv}}(\mathbf{x}, \boldsymbol{\theta}_{\text{conv}}), \tag{5.10}$$

for all inputs $\mathbf{x}$*.*

(b) *Conversely, given any* T *filter coefficients* $\{\mathbf{L}_j\}_{j=0}^{T-1}$*, there exists RNN model with* $n \leq Tn_xn_y$ *hidden states such that the RNN and 1D convolutional model have identical input-output mappings over* T*-length sequences.*

Proof. These are standard results from linear systems theory [66]. In the linear systems theory, the coefficients $\mathbf{L}_i$ are together called the *matrix impulse response*. Part (b) follows by finding $\mathbf{C}$, $\mathbf{W}$ and $\mathbf{F}$ to match the equations (5.9). Details are given in the Appendix D.1. □

5.2.5 Linear and Non-Linear Parametrizations:

Proposition 1 shows that linear RNNs with sufficient width can represent the same input-output mappings as any linear convolution system. The difference between the models is in the parameterizations. The output of the convolutional model is linear in the parameters $\boldsymbol{\theta}_{\text{conv}}$ whereas it is non-linear in $\boldsymbol{\theta}_{\text{RNN}}$. As we will see below, the non-linear parameterization of the RNN results in certain implicit biases.

5.3 NTKs of Linear RNNs and Scaled Convolutional Models

5.3.1 Neural Tangent Kernel Background

To state our first set of results, we briefly review the neural tangent kernel (NTK) theory from [7, 63]. An overview of the kernel methods and the NTK is given in Chapter 2 Section 2.4.3. We also briefly review the main definitions and results that we need below: Consider the problem of learning a (possibly non-linear) model of the form

$$\widehat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta}), \tag{5.11}$$

where $\mathbf{x} \in \mathbb{R}^{m_x}$ is an input, $f(\cdot)$ is a model function differentiable with respect to parameters θ, and $\widehat{\mathbf{y}}$ is some prediction of an output $\mathbf{y} \in \mathbb{R}^{m_y}$. The problem is to learn the parameters $\boldsymbol{\theta}$ from training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$. For sequence problems, we use the convention that each $\mathbf{x}_i$ and $\mathbf{y}_i$ represent one entire input-output sequence pair. Hence, the dimensions will be $m_x = n_x T$ and $m_y = n_y T$.

Now, given the training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ and an initial parameter estimate $\boldsymbol{\theta}^0$, the *neural tangent kernel (NTK) model* is the linear model

$$\widehat{\mathbf{y}} = f^{\text{lin}}(\mathbf{x}, \boldsymbol{\alpha}) := f(\mathbf{x}, \theta^0) + \sum_{j=1}^{N} K(\mathbf{x}_j, \mathbf{x})\boldsymbol{\alpha}_j, \tag{5.12}$$

where $K(\mathbf{x}, \mathbf{x}')$ is the so-called NTK,

$$[K(\mathbf{x}, \mathbf{x}')]_{ij} := \left\langle \frac{\partial f_i(\mathbf{x}, \boldsymbol{\theta}^0)}{\partial \boldsymbol{\theta}}, \frac{\partial f_j(\mathbf{x}', \boldsymbol{\theta}^0)}{\partial \boldsymbol{\theta}} \right\rangle. \tag{5.13}$$

and $\boldsymbol{\alpha}$ is a vector of dual coefficients,

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N), \quad \boldsymbol{\alpha}_j \in \mathbb{R}^{m_x}. \tag{5.14}$$

Note that $K(\mathbf{x}, \mathbf{x}')$ depends implicitly on $\boldsymbol{\theta}^0$. Also, for a fixed initial condition, $\boldsymbol{\theta}^0$, the model (D.35) is linear in the parameters $\boldsymbol{\alpha}$, and hence potentially easier to analyze than the original non-linear model (5.11). The key result in NTKs is that, for certain wide neural networks with random initializations, (full-batch) gradient descent training of the non-linear and linear models are *asymptotically identical*. For example, the results in [72] and [3] provide the following proposition:

Proposition 2. *Suppose that $f_n(\mathbf{x}, \boldsymbol{\theta})$ is a sequence of recurrent neural networks with n hidden states and non-linear activation function $\sigma(\cdot)$. Let $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ be some fixed training data contained in a compact set. Let $\widehat{\boldsymbol{\theta}}_n^0$ denote a random initial condition generated as* (5.4) *and let $\widehat{\boldsymbol{\theta}}_n^\ell$ denote the parameter estimate after ℓ steps of (full-batch) gradient descent with some learning rate η. Let $K_n(\mathbf{x}, \mathbf{x}')$ denote the NTK of the RNN and $f_n^{\text{lin}}(\mathbf{x}, \boldsymbol{\alpha})$ denote the corresponding linear NTK model* (D.35)*. Let $\widehat{\boldsymbol{\alpha}}_n^\ell$ denote the parameter estimate obtained with GD with the same learning rate. We further assume that the non-linear activation σ satisfies*

$$|\sigma(0)|, \quad \|\sigma'\|_\infty, \quad \sup_{\mathbf{x} \neq \mathbf{x}'} |\sigma'(\mathbf{x}) - \sigma'(\mathbf{x}')|/|\mathbf{x} - \mathbf{x}'| < \infty.$$

Then, for all $\mathbf{x}$ and $\mathbf{x}'$,

$$\lim_n K_n(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') \ \ a.s. \tag{5.15}$$

for some deterministic positive semi-definite matrix $K(\mathbf{x}, \mathbf{x}')$. Moreover, if $\lambda_{\min}(K) > 0$, then for sufficiently small learning rate η and any new sample $\mathbf{x}$,

$$\lim_{n \to \infty} \sup_{\ell \geq 0} \| f_n(\mathbf{x}, \widehat{\boldsymbol{\theta}}_n^\ell) - f_n^{\text{lin}}(\mathbf{x}, \widehat{\boldsymbol{\alpha}}_n^\ell) \| = 0, \tag{5.16}$$

where the convergence is in probability.

The consequence of this result is that the behavior of certain infinitely-wide neural networks on new samples $\mathbf{x}$ is identical to the behavior of the linearized network around its initialization. This essentially means that as $n \to \infty$, the learning dynamics for the original

and the linearized networks match during training.

5.3.2 NTK for the Convolutional Model

Having defined the NTK, we first compute the NTK of the scaled convolutional model (5.8).

Theorem 6. *Fix a time period T and consider the convolutional model* (5.8) *for a given set of scale factors $\boldsymbol{\rho} = (\rho_0, \ldots, \rho_{T-1})$. Then, for any initial condition, and any two input sequences $\mathbf{x}$ and $\mathbf{x}'$, the NTK for this model is,*

$$K(\mathbf{x}, \mathbf{x}') = \mathcal{T}(\mathbf{x})^\mathsf{T} D(\boldsymbol{\rho}) \mathcal{T}(\mathbf{x}') \otimes \mathbf{I}_{n_y}, \tag{5.17}$$

where $\mathcal{T}(\mathbf{x})$ is the Toeplitz matrix,

$$\mathcal{T}(\mathbf{x}) := \begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_{T-1} \\ 0 & \boldsymbol{x}_0 & \cdots & \boldsymbol{x}_{T-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{x}_0 \end{bmatrix} \in \mathbb{R}^{Tn_x \times T}. \tag{5.18}$$

and $D(\boldsymbol{\rho})$ is the diagonal matrix,

$$D(\boldsymbol{\rho}) := \operatorname{diag}(\rho_0 \mathbf{I}_{n_x}, \cdots, \rho_{T-1} \mathbf{I}_{n_x}). \tag{5.19}$$

Proof. See Appendix D.2 for proof. □

5.3.3 NTK for the RNN Parametrization

We now compare the NTK of the scaled convolutional model to the NTK for the RNN parameterization.

Theorem 7. *Fix a time period T and consider the RNN model* (5.1) *mapping an input sequence $\mathbf{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1})$ to an ouptut sequence $\mathbf{y} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{T-1})$ with the parameters*

$(\mathbf{W}, \mathbf{F}, \mathbf{C})$. *Assume the parameters are initialized as* (5.4) *for some constants* $\nu_W, \nu_F, \nu_C > 0$. *In the limit as the number of hidden states* $n \to \infty$*:*

(a) *The impulse response coefficients* $\mathbf{L}_j$ *in* (5.9) *converge in distribution to independent Gaussians, where the components of* $\mathbf{L}_j$ *are i.i.d.* $\mathcal{N}(0, \nu_C \nu_F \nu_W^j)$.

(b) *Given any input sequences* $\mathbf{x}$ *and* $\mathbf{x}'$*, the NTK converges almost surely to the deterministic limit:*

$$K(\mathbf{x}, \mathbf{x}') = \mathcal{T}(\mathbf{x})^{\mathsf{T}} D(\boldsymbol{\rho}) \mathcal{T}(\mathbf{x}') \otimes \mathbf{I}_{n_y}, \tag{5.20}$$

where $\mathcal{T}(\mathbf{x})$ *and* $D(\boldsymbol{\rho})$ *are given in* (5.18) *and* (5.19) *and*

$$\rho_j = \nu_C (j \nu_F \nu_W^{j-1} + \nu_W^j) + \nu_F \nu_W^j. \tag{5.21}$$

Proof. See Appendix D.3 for the proof. □

Comparing Theorems 6 and 7, we see that the NTK for linear RNN is identical to that of an scaled convolution model when the scalings are chosen as (5.21). From Proposition 2, we see that, in the wide limit regime where $n \to \infty$, gradient descent training of the linear RNN with the nonlinear parametrization $\boldsymbol{\theta}_{\text{RNN}} = (\mathbf{W}, \mathbf{F}, \mathbf{C})$ in (5.3) is identical to the training of the convolutional model (5.5) where the linear parameters $\mathbf{L}_j$ are initialized as i.i.d. Gaussians and then trained with certain scaling factors (5.21).

Moreover, the scaling factors have a geometric decay. Recall from Section 5.2 that, for stability of the linear RNN we require that $\nu_W < 1$. When $\nu_W < 1$ and $j > 1$, the scaling factors (5.21) can be bounded as

$$\rho_j \leq \rho_{\max} \nu_W^{j-1}, \quad \rho_{\max} := \nu_C (T \nu_F + 1) + \nu_F. \tag{5.22}$$

Consequently, the scale factors decay geometrically with ν_W^{j-1}. This implies that, in the training of the scaled convolutional model, the coefficients with higher delay j will be given

lower weight.

5.4 Implicit Bias of Linear RNNs

An importance consequence of the geometric decay of the scaling factors ρ_j in (5.22) is the *implicit bias* of GD training of linear RNNs towards networks with short-term memory. To state this precisely, fix input and output training data $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, N$. For each n, consider the RNN model (5.2) with n hidden states and parameters $\boldsymbol{\theta}_{\text{RNN}}$ in (5.3). Assume the parameters are initialized as $\boldsymbol{\theta}^0_{\text{RNN}} = (\mathbf{W}^0, \mathbf{F}^0, \mathbf{C}^0)$ in (5.4) for some $\nu_W, \nu_F, \nu_C > 0$. Let

$$\boldsymbol{\theta}^\ell_{\text{RNN}} = (\mathbf{W}^\ell, \mathbf{F}^\ell, \mathbf{C}^\ell),$$

denote the parameter after ℓ steps of (full batch) gradient descent with some learning rate η. Let $\mathbf{L}^\ell_{\text{RNN}}$ be the resulting impulse response coefficients (5.9),

$$\mathbf{L}^\ell_{\text{RNN},j} = n^{-(j+1)/2}\mathbf{C}^\ell(\mathbf{W}^\ell)^j\mathbf{F}^\ell, \quad j = 0, \ldots, T-1. \tag{5.23}$$

We then have the following bound.

Theorem 8. *Under the above assumptions, the norm of the impulse response coefficients of the RNN at the initial iteration $\ell = 0$ are given by*

$$\lim_{n\to\infty} \mathbb{E}\|\mathbf{L}^0_{\text{RNN},j}\|_F^2 = n_x n_y \nu_C \nu_F \nu_W^j. \tag{5.24}$$

Also, there exists constants B_1 and B_2 such that if the learning rate satisfies $\eta < B_1$, then for all iterations ℓ,

$$\limsup_{n\to\infty} \|\mathbf{L}^\ell_{\text{RNN},j} - \mathbf{L}^0_{\text{RNN},j}\|_F \le B_2\rho_j\eta\ell, \tag{5.25}$$

where the convergence is in probability. Moreover, the constants B_1 and B_2 can be selected independent of ν_W.

Proof. See Appendix D.4 for proof. □

To understand the significance of the theorem, observe that in the convolutional model (5.5), $\mathbf{L}_j$ relates the input time samples $\boldsymbol{x}_{t-j}$ to the output $\boldsymbol{y}_t$. The coefficient $\mathbf{L}_j$ thus describes the influence of the inputs samples on the output samples j time steps later. Combining (5.24), (5.25), and (5.22), we see that these coefficients decay as

$$\mathbf{L}^{\ell}_{\text{RNN},j} = O(\nu_W^{j/2} + \ell\nu_W^j).$$

Also, as discussed in Section 5.2, we need $\nu_W < 1$ for stability. Hence, the magnitude of these coefficients decay geometrically with ν_W^{j-1}. Therefore, for a fixed number of training steps, the effect of the input on the output at a lag of j would be exponentially small in j. In this sense, training linear RNNs with the non-linear parameterization $\boldsymbol{\theta}_{\text{RNN}} = (\mathbf{W}, \mathbf{F}, \mathbf{C})$ is implicitly biased to short memory.

It is useful to compare the performance of an unscaled convolutional model with the linear RNN. The convolutional model can fit any linear time-invariant system with an arbitrary delay. We have seen in Proposition 1 that, in principle, the linear RNN can also fit any such system with a sufficient number of hidden states. However, the above theorem shows that unless the number of gradient steps grows exponentially with the desired delay, the parameterization of the linear RNN will strongly bias the solutions to systems with short memory. This restriction will create bias error on systems that have long-term memory. On the other hand, due to the implicit constraint of the linear RNN, the parameterization will reduce the variance error.

5.5 Numerical Experiments

We validate our theoretical results on a number of synthetic and real data experiments.

Synthetic data In section 5.3, we showed that the NTK for a linear RNN given in (5.1) with parameters $\boldsymbol{\theta}_{\text{RNN}}$ in (5.3) is equivalent to the NTK for its convolutional parameterization with parameters $\boldsymbol{\theta}_{\text{conv}}$. In order to validate this, we compared the training dynamics of a linear RNN, eq (5.1), with a large number of hidden states, n, and a scaled convolutional model (5.8) with scale coefficients ρ_j defined in (5.21).

We generated data from a synthetic (teacher) RNN with random parameters. For the data generation system we used a linear RNN with 4 hidden units and $n_x = n_y = 1$. Matrices $\mathbf{W}$, $\mathbf{F}$, and $\mathbf{C}$ are generated as i.i.d random Gaussian with $\nu_W = 0.3$ and $\nu_F = \nu_C = 1$. We added noise to the output of this system such that the signal-to-noise ratio (SNR) is 20 dB. We generated 50 training sequences and 50 test sequences in total and each sequence has $T = 10$ time steps.

Given the training and test data, we train (i) a (student) linear RNN with $n = 1000$ hidden units, $\nu_W = 0.3$, and $\nu_F = \nu_C = 1$; and (ii) a 1D-convolutional model with scale coefficients ρ_j calculated in (5.21) using $\nu_W = 0.3$, $\nu_F = \nu_C = 1$. We used mean-squared error as the loss function for both models and applied full-batch gradient descent with learning rate $lr = 10^{-4}$. Fig. 5.1 shows the identical dynamics of training for both models. Fig. 5.2 shows a zoomed-in version of the dynamics for training error. We see, as the theory predicts, the RNN and scaled Convolution 1D model have an identical performance.

To evaluate the performance of these models for a task with long-term dependencies, we created a dataset where we manually added different delay steps τ to the output of a true linear RNN system i.e. $\boldsymbol{y}_t = \boldsymbol{x}_{t-\tau}$. We have chosen longer ($T = 20$) true sequences for this task. We then learned this data using the aforementioned linear RNN and scaled 1D convolutional models. We also trained an unscaled 1D convolutional model with this data to compare performances. With unscaled convolutional model, we exactly learn the impulse response coefficients $\mathbf{L}_j$ defined in (5.5) during training.

Fig. 5.3 shows the test error with respect to delay steps for all three models. Observe that the performance of the scaled convolutional and the linear RNN models match during

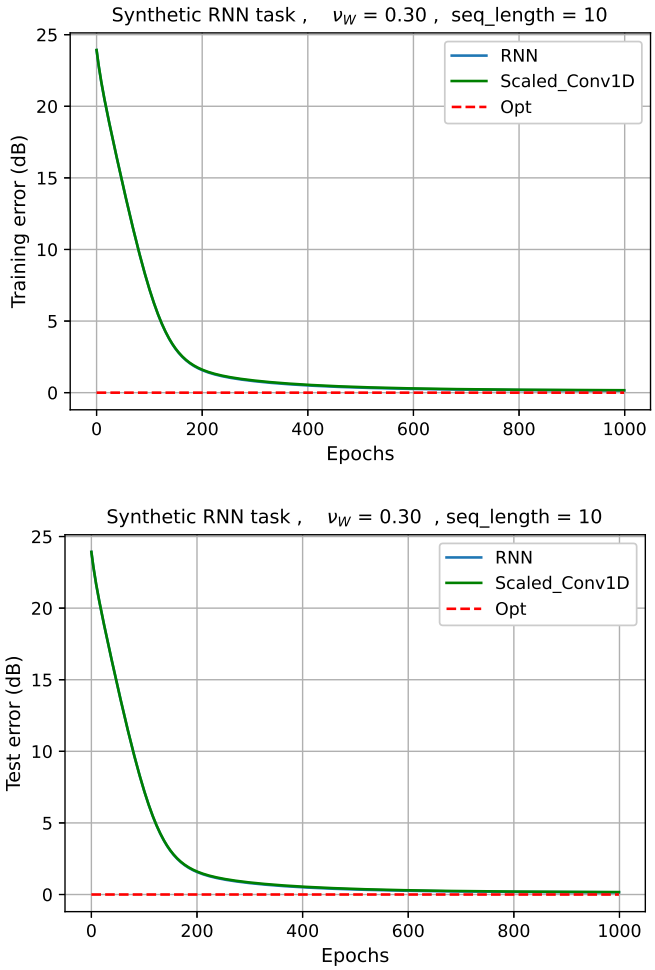

Figure 5.1: Dynamics of an RNN and its equivalent scaled Conv-1D in learning a synthetic task. The data is generated from a synthetic RNN with $n_x = n_y = 1$ and $nh = 4$. Noise is added to the output with SNR= 20 dB The sequence length T=10. Training and test samples are $N_{\text{tr}} = N_{\text{ts}} = 50$. Full batch gradient descent is used with $lr = 10^{-4}$. The dynamics of these models perfectly match.

training. Due to the bias of these models against the delay, the test error increases as we increase the delay steps in our system. On the other hand, the performance of the unscaled convolutional model stays almost the same with increasing delay, slightly changing at larger delays as there is less data to track. We thus see the effect of implicit bias: When the true system does not have high delay, the implicit bias of the RNN and scaled convolutional model against delay helps. As the true delay increases, the implicit bias causes bias error not present in the unscaled convolutional model.

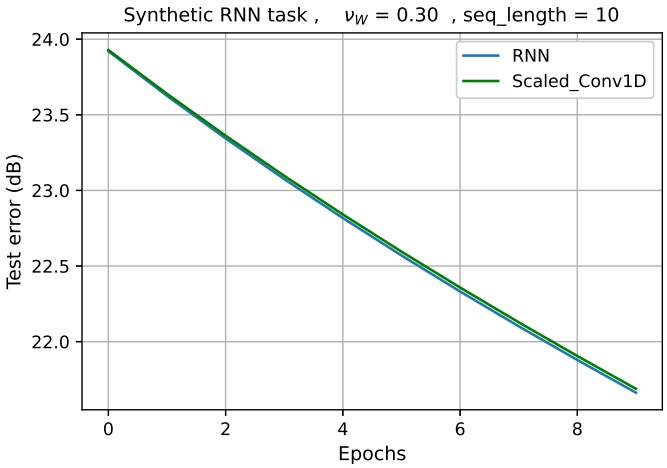

Figure 5.2: The first 10 epochs in Fig. 5.1. Note that to be theoretically accurate, we need $lr \to 0$ to be in the kernel regime.

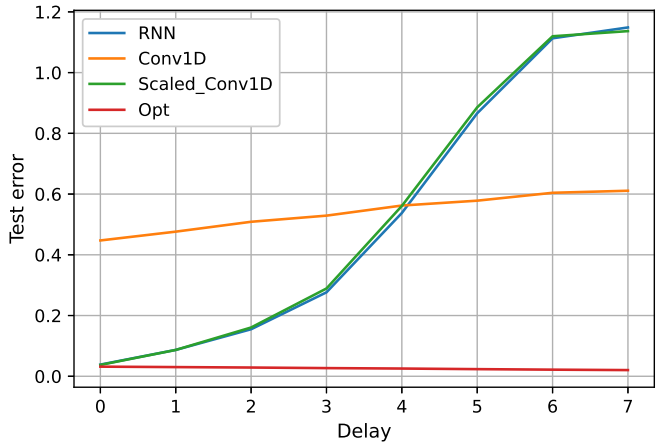

Figure 5.3: Test performance with respect to delay. For this task we have $n_h = 1000$, $N_{\text{tr}} = N_{\text{ts}} = 10$, $n_x = 15$, $n_y = 1$, and $T = 20$. The delay is added manually by shifting i.e. $\boldsymbol{y}_t = \boldsymbol{x}_{t-delay}$ and the output SNR = 20 dB.

Our theoretical results hold in the asymptotic regime where the number of hidden units of the RNN goes to infinity. To test the applicability of our results to real-world RNNs that have a finite number of hidden units, we run an experiment where we change the number of hidden units and test how closely the RNN training follows the kernel training, i.e. the equivalent scaled convolutional model. In this experiment, the true RNN that generates the synthetic data has 20 hidden units and we train different RNNs with 10, 40, and 200 hidden

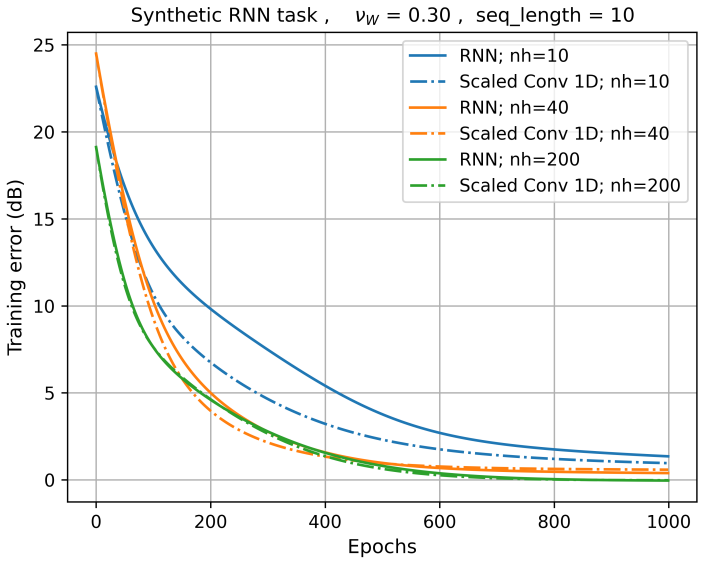

Figure 5.4: Training error over the course of training for RNNs with different number of hidden units. Solid curves show the error of RNN whereas the dashed curves show the error of equivalent scaled convolutional model for each epoch. Our theoretical results show that in the asymptotic regime of $n_h \to \infty$ with inifinitesimal learning rate, the solid curves and dashed curves should exactly match. Here, we see that as we increase n_h the two curves get closer and even for $n_h = 200$ our theoretical prediction almost perfectly matches what we observe in practice.

units to learn the synthetic data. The results are shown in Figure 5.4. For each n_h, the RNN (solid lines) along with the equivalent scaled convolutional model (dashed line) are trained. Our theory claims that when the learning rate of gradient descent goes to zero and the number of hidden units goes to infinity, the training error of these two models should be exactly the same over the course of the optimization, i.e. the solid curves and dashed curves should match. We see that it is indeed the case. As we increase n_h the two curves become closer and for $n_h = 200$ they almost perfectly match. This suggests that our results should be applicable in practice to RNNs of moderates size with more that 200 hidden units.

Real data We also validated our theory using spikes rate data from the macaque primary somatosensory cortex (S1) [22]. Somatosensory cortex is a part of the brain responsible for receiving sensations of touch, pain, etc from the entire body. The data is recorded during a

two-dimensional reaching task. In this task, a macaque was rewarded for positioning a cursor on a series of randomly generated targets on the screen using a handle. The data is from a single recording of 51 minutes and includes 52 neurons. The mean and median firing rates are 9.3 and 6.3 spikes/sec.Similar to the previous experiments, we also trained an unscaled 1D convolutional model with this data and compared the performances with the linear RNN and the scaled convolutional models.

We compared the performances on two sets of experiments. We first used only the 4.5 minutes of the total recorded data. The purpose of this experiment is to compare the performances in limited data circumstances. With this limited data, we expect the scaled convolutional model (and thus the RNN) to perform better than the unscaled model due to the implicit bias of the towards short-term memory and the fact that the effective number of parameters is smaller in the scaled model which leads to a smaller variance. In the second experiment, we trained our models using all the available data ($\approx$ 51 mins). In this case, the scaled model (and the RNN) performs worse because of the increased bias error. In our experiments setup, the linear RNN has $n = 1000$ hidden states and the sequence length $T = 15$. Also, $\nu_W = 0.3$ and $\nu_F = \nu_C = 1$.

Fig. 5.5 shows the R^2 scores for x and y directions of all three models for this task. Observe that, the dynamics of the linear RNN and scaled convolutional model are identical during training using either the entire recording or a part of it. For the case of limited data, as discussed earlier, we observe the implicit bias of the RNN and scaled convolutional model in the figures (a). This bias leads to better performance of these two models compared to the unscaled model. Using the total available data, the unscaled convolutional model performs better because of the increased bias error in the other two models (figures in (b)). Table 5.1 shows the test R^2-score of the final trained models for all three cases.

Table 5.1: R^2-score on test data for x and y directions in the two dimensional reaching task described in section 5.5

	RNN	Scaled Conv-1D	Conv-1D
R^2_x	0.6462	0.6442	0.6565
R^2_y	0.5911	0.5860	0.6027
R^2_x (limited data)	0.6043	0.6046	0.5856
R^2_y (limited data)	0.4257	0.4234	0.3918

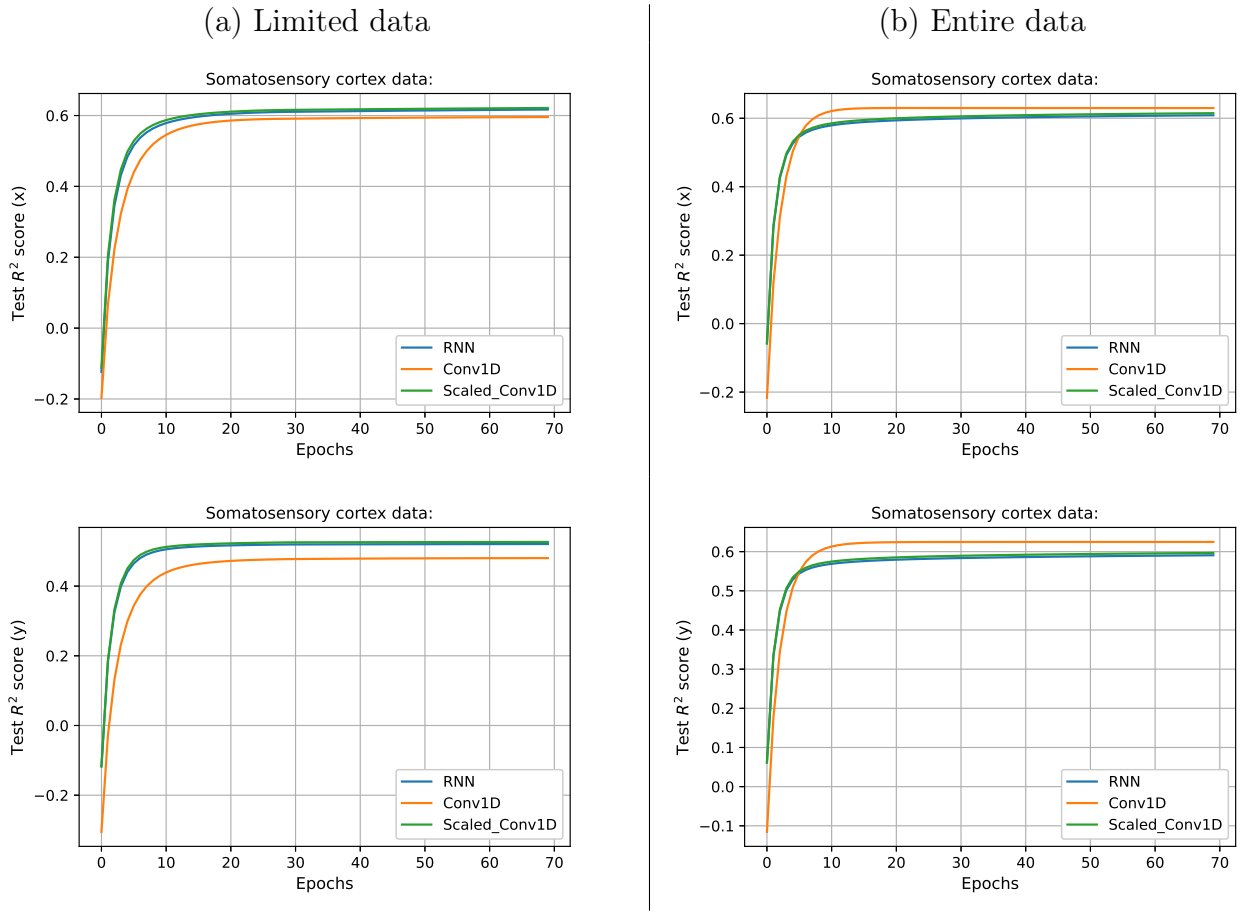

Figure 5.5: R^2 score for the two dimensional reaching task described in section 5.5 . The data is recorded from the primary somatosensory cortex of macaques. (a) Limited data: The models are trained on 4.5 minutes of recorded data. (b) Entire data: the whole recording ($\approx$ 51 mins) is used to compare the performances. For both cases we used mini-batch (batch size = 128) gradient descent with $lr = 10^{-4}$.

5.6 Implicit Bias of Non-Linear RNNs

In this section we investigate how the same kernel analysis works for a non-linear system. Consider a general non-linear system:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}), \quad \boldsymbol{y}_t = g(\mathbf{h}_t), \tag{5.26}$$

with the Lipschitz continuity condition that at $\boldsymbol{\theta} = \boldsymbol{\theta}^0$, i.e., there exists constants with $\rho \in (0,1)$, $B > 0$ and $C > 0$ such that for all $\boldsymbol{x}_0$, $\boldsymbol{x}_1$,

$$\begin{aligned} \|f(\mathbf{h}_1, \boldsymbol{x}_1, \boldsymbol{\theta}^0) - f(\mathbf{h}_0, \boldsymbol{x}_0, \boldsymbol{\theta}^0)\| &\leq \rho \|\mathbf{h}_1 - \mathbf{h}_0\| + B\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|, \\ \|g(\mathbf{h}_1, \boldsymbol{\theta}^0) - g(\mathbf{h}_0, \boldsymbol{\theta}^0)\| &\leq C\|\mathbf{h}_1 - \mathbf{h}_0\|. \end{aligned} \tag{5.27}$$

Let $\mathbf{y} = F(\mathbf{x}, \boldsymbol{\theta})$ denote the sequence-to-sequence mapping defined by the recursive equations (5.26). Let $G : \boldsymbol{\delta\theta} \mapsto \boldsymbol{\delta y}$ denote the gradient mapping:

$$G_{\mathbf{x}}(\boldsymbol{\delta\theta}) = G(\mathbf{x}, \boldsymbol{\theta}^0)(\boldsymbol{\delta\theta}) := \frac{\partial}{\partial \boldsymbol{\theta}} F(\mathbf{x}, \boldsymbol{\theta})\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \cdot \boldsymbol{\delta\theta}. \tag{5.28}$$

This mapping is given by the linear time-varying system,

$$\boldsymbol{\delta}\mathbf{h}_t = \mathbf{W}_t \, \boldsymbol{\delta}\mathbf{h}_{t-1} + \mathbf{B}_t \, \boldsymbol{\delta\theta}, \quad \boldsymbol{\delta y}_t = \mathbf{C}_t \, \boldsymbol{\delta}\mathbf{h}_t, \tag{5.29}$$

where

$$\mathbf{W}_t = \nabla_h f(\mathbf{h}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^0), \quad \mathbf{B}_t = \nabla_\theta f(\mathbf{h}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^0), \quad \mathbf{C}_t = \nabla_h g(\mathbf{h}_t, \boldsymbol{\theta}^0).$$

By the assumption (5.27),

$$\|\nabla_h f(\mathbf{h}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^0)\| \leq \rho, \quad \|\nabla_h g(\mathbf{h}_t, \boldsymbol{\theta}^0)\| \leq C, \quad \|\nabla_x f(\mathbf{h}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^0)\| \leq B. \tag{5.30}$$

For a given $\mathbf{x}, \mathbf{x}'$, the NTK in this case is defined as

$$K(\mathbf{x}, \mathbf{x}') = \sum_{\boldsymbol{\delta\theta} \in \mathcal{T}} G_{\mathbf{x}}(\boldsymbol{\delta\theta})\ G_{\mathbf{x}'}(\boldsymbol{\delta\theta})^*$$

where $\mathcal{T}$ is the standard basis for parameter space. The following theorem shows the implicit bias of non-linear RNNs in gradient descent training:

Theorem 9. *Let* $\mathbf{x}$ *be any input sequence and* $\mathbf{y} = F(\mathbf{x}, \boldsymbol{\theta}^\ell)$ *the estimate after step* ℓ *of the gradient descent. Assuming the Lipschitz continuity of* (5.27)*, we have*

$$\left\| \frac{\partial \boldsymbol{y}_t}{\partial \boldsymbol{x}_s} \right\| \leq C\ell\rho^{t-s} \tag{5.31}$$

for $t \geq s$*.*

Proof. See Appendix D.5. □

Note that the exact equivalence with the convolutional model does not hold in this case and the proof involves considering a perturbed system similar to (5.29) and evaluating the changes in the output at time t from the disturbance of the input at time $s < t$.

5.7 Summary

In this chapter, we focused on the special class of linear RNNs and observed a functional equivalence between linear RNNs and 1D convolutional models. Using the kernel regime framework, we showed that the training of a linear RNN is identical to the training of a certain scaled convolutional model. We further provided an analysis for an inductive bias in linear RNNs towards short-term memory. We showed that this bias is driven by the variances of RNN parameters at random initialization. We briefly touched on the analysis of the NTK regime for non-linear RNNs as well. Our theory is validated by both synthetic and real data experiments.

Appendices

Appendix A

Proofs for Chapter 2

A.1 Proof of Lemma 1

We prove this by induction. Let $\mathcal{M}_t$ be the hypothesis that this result is true up to iteration t. We show that $\mathcal{M}_0$ is true and that $\mathcal{M}_t$ implies $\mathcal{M}_{t+1}$.

Base case ($\mathcal{M}_0$): Define $\mathbf{z}_{0\ell} = \overline{G}_\ell(\mathbf{q}_0, \mathbf{u}_0)$. We have that rows of $\mathbf{z}_{0\ell}$ converge $PL(2)$ to $Z_{0\ell} = \overline{G}_\ell(Q_0, U_0)$. Now, let $\mu_{0\ell} = \mathbb{E}(Z_{0\ell})$ and define the following:

$$\mathbf{d}_\ell = \frac{1}{\sqrt{n}}\mathbf{A}_\ell \mathbf{1}, \quad \widetilde{\mathbf{z}}_{0\ell} = \mathbf{z}_{0\ell} - \mathbf{1}\mu_{0\ell} \tag{A.1}$$

$$\widetilde{\mathbf{r}}_{1\ell} = \frac{1}{\sqrt{n}}\mathbf{A}_\ell \widetilde{\mathbf{z}}_{0\ell}, \quad \mathbf{r}_{1\ell} = \widetilde{\mathbf{r}}_{1\ell} + \mathbf{d}_\ell \mu_{0\ell}, \quad \mathbf{q}_1 = \sum_{\ell=1}^{L} \mathbf{r}_{1\ell}. \tag{A.2}$$

We know that

$$\mathbf{d}_\ell \overset{PL(2)}{=} D_\ell \sim \mathcal{N}(0, \nu_\ell), \qquad \widetilde{\mathbf{z}}_{0\ell} \overset{PL(2)}{=} \widetilde{Z}_{0\ell} = Z_{0\ell} - \mu_{0\ell}. \tag{A.3}$$

where $D_\ell \in \mathbb{R}$ and $Z_{0\ell}, \widetilde{Z}_{0\ell} \in \mathbb{R}^{1 \times d_q}$.

Note that $\widetilde{Z}_{0\ell}$ are zero mean. Now since $\mathbf{A}_\ell$ are i.i.d Gaussian matrices, rows of $\widetilde{\mathbf{r}}_{1\ell}$

converge PL(2) to the random variable

$$\widetilde{R}_{1\ell} \sim \mathcal{N}(0, P_{1\ell}) \qquad \text{where,} \quad P_{1\ell} = \lim_{n\to\infty} \frac{1}{n} \widetilde{\mathbf{z}}_{0\ell}^{\mathsf{T}} \widetilde{\mathbf{z}}_{0\ell} \overset{a.s.}{=} \mathbb{E}(\widetilde{Z}_{0\ell}^{\mathsf{T}} \widetilde{Z}_{0\ell}) \tag{A.4}$$

Furthermore, one can show that $\mathbb{E}(\widetilde{R}_{1\ell_1} \widetilde{R}_{1\ell_2}) = \mathbb{E}(\widetilde{Z}_{0\ell_1}^{\mathsf{T}} \widetilde{Z}_{0\ell_2}) = 0$, and $\widetilde{R}_{1\ell_1}$ and $\widetilde{R}_{1\ell_2}$ are independent Therefore,

$$\mathbf{q}_1 \overset{PL(2)}{=} Q_1 = \sum_{\ell=1}^{L} \left[\widetilde{R}_{1\ell} + D_\ell \mu_{0\ell} \right] \tag{A.5}$$

This proves $\mathcal{M}_0$ holds true.

Induction recursion: We next assume that the state evolution system is true up to iteration t. We write the recursions as

$$\mathbf{d}_\ell = \frac{1}{\sqrt{n}} \mathbf{A}_\ell \mathbf{1} \in \mathbb{R}^n \tag{A.6a}$$

$$\mathbf{z}_{t\ell} = \overline{G}_\ell(\mathbf{q}_t, \mathbf{u}_t), \quad \widetilde{\mathbf{z}}_{t\ell} = \mathbf{z}_{t\ell} - \mathbf{1}\mu_{t\ell} \tag{A.6b}$$

$$\widetilde{\mathbf{r}}_{t+1,\ell} = \frac{1}{\sqrt{n}} \mathbf{A}_\ell \widetilde{\mathbf{z}}_{t\ell}, \quad \mathbf{r}_{t+1,\ell} = \widetilde{\mathbf{r}}_{t+1,\ell} + \mathbf{d}_\ell \mu_{t\ell}, \quad \mathbf{q}_{t+1} = \sum_{\ell=1}^{L} \mathbf{r}_{t+1,\ell}. \tag{A.6c}$$

The main issue in dealing with a recursion of the form Equation (A.6) is that for $t \geq 1$, matrices $\{\mathbf{A}_\ell\}_{\ell=1}^L$ and $\{\widetilde{\mathbf{r}}_{t\ell}\}_{\ell=1}^L$ are no longer independent. The key idea is to use a conditioning technique (Bolthausen conditioning) as in [16] to deal with this dependence. Instead of conditioning $\widetilde{\mathbf{r}}_{t\ell}$ on $\mathbf{A}_\ell$, we condition $\mathbf{A}_\ell$ on the event

$$\mathcal{E}_{t,\ell} = \{\widetilde{\mathbf{r}}_{t'+1,\ell} = \frac{1}{\sqrt{n}} \mathbf{A}_\ell \widetilde{\mathbf{z}}_{t'\ell}, t' = 0, \ldots, t-1\}. \tag{A.7}$$

Note that this event is a set of linear constraints, and i.i.d. Gaussian random variables conditioned on linear constraints have Gaussian densities that we can track.

Let $\widetilde{\mathcal{H}}_{t\ell}$ be the linear operator

$$\widetilde{\mathcal{H}}_{t\ell} : \mathbf{A}_\ell \mapsto (\widetilde{\mathbf{r}}_{1\ell}, \ldots, \widetilde{\mathbf{r}}_{t\ell}). \tag{A.8}$$

With these definitions, we have

$$\mathbf{A}_\ell|_{\varepsilon_{t,\ell}} \stackrel{d}{=} \widetilde{\mathcal{H}}^{\dagger}_{t\ell}(\widetilde{\mathbf{r}}_{1\ell}, \ldots, \widetilde{\mathbf{r}}_{t\ell}) + \widetilde{\mathcal{H}}^{\perp}_{t\ell}(\widetilde{\mathbf{A}}_\ell), \tag{A.9}$$

where $\widetilde{\mathcal{H}}^{\dagger}_{t,\ell}$ is the Moore-Penrose pseudo-inverse operator of $\widetilde{\mathcal{H}}_{t,\ell}$, $\widetilde{\mathcal{H}}^{\perp}_{t,\ell}$ is the orthogonal projection operator onto the subspace orthogonal to the kernel of $\widetilde{\mathcal{H}}_t$, and $\widetilde{\mathbf{A}}_\ell$ is an independent copy of $\mathbf{A}_\ell$. Therefore, we can write $\widetilde{\mathbf{r}}_{t+1,\ell}$ as sum of two terms

$$\widetilde{\mathbf{r}}_{t+1,\ell} = \widetilde{\mathbf{r}}^{\text{det}}_{t+1,\ell} + \widetilde{\mathbf{r}}^{\text{ran}}_{t+1,\ell}, \tag{A.10}$$

where $\widetilde{\mathbf{r}}^{\text{det}}_{t+1,\ell}$ is what we call the deterministic part:

$$\widetilde{\mathbf{r}}^{\text{det}}_{t+1,\ell} = \frac{1}{\sqrt{n}} \widetilde{\mathcal{H}}^{\dagger}_{t\ell}(\widetilde{\mathbf{r}}_1, \ldots, \widetilde{\mathbf{r}}_t)\ \widetilde{\mathbf{z}}_{t\ell} \tag{A.11}$$

and $\widetilde{\mathbf{r}}^{\text{ran}}_{t+1,\ell}$ is the random part:

$$\widetilde{\mathbf{r}}^{\text{ran}}_{t+1,\ell} = \frac{1}{\sqrt{n}} \widetilde{\mathcal{H}}^{\perp}_{t\ell}(\widetilde{\mathbf{A}}_\ell)\ \widetilde{\mathbf{z}}_{t\ell}. \tag{A.12}$$

It is helpful to write the linear operators defined in this section in matrix form for derivations that follow

$$\widetilde{\mathcal{H}}_{t\ell}(\mathbf{A}_\ell) = \frac{1}{\sqrt{n}} [\mathbf{A}_\ell] \begin{bmatrix} \widetilde{\mathbf{z}}_{0\ell} & \ldots & \widetilde{\mathbf{z}}_{t-1,\ell} \end{bmatrix}. \tag{A.13}$$

Deterministic part: We first characterizes the limiting behavior of $\widetilde{\mathbf{r}}^{\text{det}}_{t+1,\ell}$. It is easy to show that if the functions $\overline{G}_\ell$ are non-constant, then the operator $\widetilde{\mathcal{H}}_{t\ell}\widetilde{\mathcal{H}}^{\mathsf{T}}_{t\ell}$ where $\widetilde{\mathcal{H}}^{\mathsf{T}}_{t\ell}$ is the

adjoint of $\widetilde{\mathcal{H}}_{t\ell}$, is full-rank almost surely for any finite t. Thus, we have

$$\widetilde{\mathcal{H}}_{t\ell}^{\dagger} = \widetilde{\mathcal{H}}_{t\ell}^{\mathsf{T}}(\widetilde{\mathcal{H}}_{t\ell}\widetilde{\mathcal{H}}_{t\ell}^{\mathsf{T}})^{-1} \tag{A.14}$$

Form equation (A.8) we have

$$\widetilde{\mathcal{H}}_{t\ell}^{\mathsf{T}}(\widetilde{\mathbf{r}}_{1\ell}, \ldots, \widetilde{\mathbf{r}}_{t\ell}) = \frac{1}{\sqrt{n}} \sum_{t'=1}^{t} \widetilde{\mathbf{r}}_{t'\ell}(\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}}. \tag{A.15}$$

Combining (A.15) and (A.8) we get

$$\left(\widetilde{\mathcal{H}}_{t\ell}(\widetilde{\mathcal{H}}_{t\ell}^{\mathsf{T}})(\widetilde{\mathbf{r}}_{1\ell}, \ldots, \widetilde{\mathbf{r}}_{t\ell})\right)_s = \frac{1}{n} \sum_{t'=1}^{t} \widetilde{\mathbf{r}}_{t'\ell}\ (\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}} \widetilde{\mathbf{z}}_{s-1,\ell}. \tag{A.16}$$

Now, under the induction hypothesis, using the definition of PL(2) convergence we have

$$R_{\widetilde{Z}\ell}(t', s) := \lim_{n\to\infty} \frac{1}{n} (\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}} \widetilde{\mathbf{z}}_{s-1,\ell} \overset{a.s.}{=} \mathbb{E}\left((\widetilde{Z}_{t'-1,\ell})^{\mathsf{T}} \widetilde{Z}_{s-1,\ell}\right). \tag{A.17}$$

Therefore, we have

$$\widetilde{\mathcal{H}}_{t\ell}\widetilde{\mathcal{H}}_{t\ell}^{\mathsf{T}}(\widetilde{\mathbf{r}}_{1\ell}, \ldots, \widetilde{\mathbf{r}}_{t\ell}) = [\widetilde{\mathbf{r}}_{1\ell} \quad \ldots \quad \widetilde{\mathbf{r}}_{t\ell}] \underbrace{\begin{bmatrix} R_{\widetilde{Z}\ell}(0,0) & R_{\widetilde{Z}\ell}(0,1) & \ldots & R_{\widetilde{Z}\ell}(0,t-1) \\ R_{\widetilde{Z}\ell}(1,0) & R_{\widetilde{Z}\ell}(1,1) & \ldots & R_{\widetilde{Z}\ell}(1,t-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_{\widetilde{Z}\ell}(t-1,0) & R_{\widetilde{Z}\ell}(t-1,1) & \ldots & R_{\widetilde{Z}\ell}(t-1,t-1) \end{bmatrix}}_{\mathcal{R}_{\widetilde{Z}\ell}}. \tag{A.18}$$

Let $\mathcal{R}_{\widetilde{Z}\ell}^{-1}$ denote the inverse of $\mathcal{R}_{\widetilde{Z}\ell}$ and index its blocks similarly to $\mathcal{R}_{\widetilde{Z}\ell}$. Then, the pseudo-inverse is

$$\widetilde{\mathcal{H}}_{t\ell}^{\dagger}(\widetilde{\mathbf{r}}_{1\ell}, \ldots, \widetilde{\mathbf{r}}_{t\ell}) = \frac{1}{\sqrt{n}} \sum_{t'=1}^{t} \sum_{t''=1}^{t} \widetilde{\mathbf{r}}_{t''\ell} \mathcal{R}_{\widetilde{Z}\ell}^{-1}(t''-1, t'-1)(\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}} + o(\frac{1}{n}). \tag{A.19}$$

Define $\widetilde{P}_{t\ell} := \widetilde{Z}_{t\ell} - \sum_{j=1}^{t} \widetilde{Z}_{t-j,\ell} F_{tj\ell}$, , where $F_{t,:,\ell}$ are defined in (2.47d). Using equation (A.11) we get:

$$\widetilde{\mathbf{r}}^{\text{det}}_{t+1,\ell} = \frac{1}{n} \sum_{t''=1}^{t} \widetilde{\mathbf{r}}_{t''\ell} \sum_{t'=1}^{t} \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1, t'-1) (\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}} \widetilde{\mathbf{z}}_{t,\ell} + o(\frac{1}{n}) \tag{A.20a}$$

$$\overset{a.s.}{=} \sum_{t''=1}^{t} \widetilde{\mathbf{r}}_{t''\ell} \sum_{t'=1}^{t} \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1, t'-1) \; \mathbb{E}\left((\widetilde{Z}_{t'-1,\ell})^{\mathsf{T}} \widetilde{Z}_{t,\ell} \right) + o(\frac{1}{n}) \tag{A.20b}$$

$$= \sum_{t''=1}^{t} \widetilde{\mathbf{r}}_{t''\ell} \sum_{t'=1}^{t} \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1, t'-1) \; \mathbb{E}\left((\widetilde{Z}_{t'-1,\ell})^{\mathsf{T}} (\widetilde{P}_{t\ell} + \sum_{j=1}^{t} \widetilde{Z}_{t-j,\ell} F_{t,j,\ell}) \right) + o(\frac{1}{n}) \tag{A.20c}$$

$$= \sum_{t''=1}^{t} \widetilde{\mathbf{r}}_{t''\ell} \sum_{j=1}^{t} \underbrace{\sum_{t'=1}^{t} \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1, t'-1) \; \mathcal{R}_{\widetilde{Z}\ell}(t'-1, t-j)}_{I\delta(t''=t-j+1)} F_{t,j,\ell} + o(\frac{1}{n}) \tag{A.20d}$$

$$= \sum_{j=1}^{t} \widetilde{\mathbf{r}}_{t-j+1,\ell} F_{t,j,\ell} + o(\frac{1}{n}), \tag{A.20e}$$

where (a) follows from the fact that $\mathbb{E}(\widetilde{Z}^{\mathsf{T}}_{t'\ell} \widetilde{P}_{t\ell}) = 0$ for $t' = 0, \ldots, t-1$. Now by induction hypothesis we know that $\widetilde{\mathbf{r}}_{t-j+1,\ell} \overset{PL(2)}{=} \widetilde{R}_{t-j+1,\ell}$, therefore,

$$\widetilde{\mathbf{r}}^{\text{det}}_{t+1,\ell} \overset{PL(2)}{=} \widetilde{R}^{\text{det}}_{t+1,\ell} = \sum_{j=1}^{t} \widetilde{R}_{t-j+1,\ell} F_{t,j,\ell}. \tag{A.21}$$

Random part We next consider the random part:

$$\widetilde{\mathbf{r}}^{\text{ran}}_{t+1,\ell} = \frac{1}{\sqrt{n}} \widetilde{\mathcal{H}}^{\perp}_{t\ell}(\widetilde{\mathbf{A}}_{\ell}) \widetilde{\mathbf{z}}_{t\ell} \tag{A.22}$$

$$= \frac{1}{\sqrt{n}} (\widetilde{\mathbf{A}}_{\ell} \widetilde{\mathbf{z}}_{t\ell} - \widetilde{\mathcal{H}}^{\dagger}_{t\ell} \widetilde{\mathcal{H}}_{t\ell}(\widetilde{\mathbf{A}}_{\ell}) \widetilde{\mathbf{z}}_{t\ell}). \tag{A.23}$$

We know that,

$$\widetilde{\mathcal{H}}^{\dagger}_{t} \widetilde{\mathcal{H}}_{t}(\widetilde{\mathbf{A}}_{\ell}) = \frac{1}{n} \sum_{t'=1}^{t} \sum_{t''=1}^{t} \widetilde{\mathbf{A}}_{\ell} \widetilde{\mathbf{z}}_{t''-1,\ell} \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1, t'-1) (\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}} + o(\frac{1}{n}). \tag{A.24}$$

Then, we have

$$\widetilde{\mathbf{r}}^{\text{ran}}_{t+1,\ell} = \frac{1}{\sqrt{n}}\widetilde{\mathbf{A}}_\ell \widetilde{\mathbf{z}}_{t\ell} - \frac{1}{\sqrt{n}}\sum_{t'=1}^{t}\sum_{t''=1}^{t} \widetilde{\mathbf{A}}_\ell \widetilde{\mathbf{z}}_{t''-1,\ell}\ \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1,t'-1)\left(\frac{1}{n}(\widetilde{\mathbf{z}}_{t'-1,\ell})^{\mathsf{T}}\widetilde{\mathbf{z}}_{t\ell}\right) + o(\frac{1}{n}) \tag{A.25}$$

$$= \frac{1}{\sqrt{n}}\widetilde{\mathbf{A}}_\ell \widetilde{\mathbf{z}}_{t\ell} - \frac{1}{\sqrt{n}}\sum_{t''=1}^{t} \widetilde{\mathbf{A}}_\ell \widetilde{\mathbf{z}}_{t''-1,\ell}\ \sum_{j=1}^{t}\sum_{t'=1}^{t} \mathcal{R}^{-1}_{\widetilde{Z}\ell}(t''-1,t'-1)\ \mathcal{R}_{\widetilde{Z}\ell}(t'-1,t-j)F_{t,j,\ell} + o(\frac{1}{n}) \tag{A.26}$$

$$= \frac{1}{\sqrt{n}}\widetilde{\mathbf{A}}_\ell(\widetilde{\mathbf{z}}_{t\ell} - \sum_{j=1}^{t}\widetilde{\mathbf{z}}_{t-j,\ell}F_{t,j,\ell}) + o(\frac{1}{n}) \tag{A.27}$$

Therefore, since $\widetilde{\mathbf{A}}_\ell$ are i.i.d. Gaussian matrices, $\widetilde{\mathbf{r}}^{\text{ran}}_{t+1,\ell}$ converges PL(2) to a Gaussian random variable $\widetilde{R}^{\text{ran}}_{t+1,\ell} \sim \mathcal{N}(0, P_{t+1,\ell})$ such that,

$$P_{t+1,\ell} = \mathbb{E}(\widetilde{Z}_{t\ell} - \sum_{j=1}^{t}\widetilde{Z}_{t-j,\ell}F_{tj\ell})^{\mathsf{T}}(\widetilde{Z}_{t\ell} - \sum_{j=1}^{t}\widetilde{Z}_{t-j,\ell}F_{tj\ell}) \tag{A.28}$$

We can now write $\widetilde{R}_{t+1,\ell}$ as,

$$\widetilde{R}_{t+1,\ell} = \widetilde{R}^{\text{det}}_{t+1,\ell} + \widetilde{R}^{\text{ran}}_{t+1,\ell} = \sum_{j=1}^{t}\widetilde{R}_{t-j+1,\ell}F_{t,j,\ell} + \mathcal{N}(0, P_{t+1,\ell}), \tag{A.29}$$

and by equation (A.6) we have

$$Q_{t+1} = \sum_{\ell=1}^{L} R_{t+1,\ell}, \qquad R_{t+1,\ell} = \widetilde{R}_{t+1,\ell} + D_\ell \mu_{t\ell}.$$

This proves $\mathcal{M}_t$ implies $\mathcal{M}_{t+1}$.

Appendix B

Proofs for Chapter 3

B.1 Empirical Convergence of Fixed Points

A consequence of Assumption 2 is that we can take the limit $k \to \infty$ of the random variables in the SE algorithm. Specifically, let $\mathbf{x}_k = \mathbf{x}_k(N)$ be any set of d outputs from the ML-VAMP for GLM Learning Algorithm under the assumptions of Theorem 2. Under Assumption 2, for each N, there exists a vector

$$\mathbf{x}(N) = \lim_{k\to\infty} \mathbf{x}_k(N), \tag{B.1}$$

representing the limit over k. For each k, Theorem 2 shows there also exists a random vector limit,

$$\lim_{N\to\infty} \{\mathbf{x}_{k,i}(N)\} \overset{PL(2)}{=} X_k, \tag{B.2}$$

representing the limit over N. The following proposition shows that we can take the limits of the random variables X_k.

Proposition 3. *Consider the outputs of the ML-VAMP for GLM Learning Algorithm under the assumptions of Theorem 2 and Assumption 2. Let $\mathbf{x}_k = \mathbf{x}_k(N)$ be any set of d outputs from the algorithm and let $\mathbf{x}(N)$ be its limit from* (B.1) *and let X_k be the random variable limit* (B.2)*. Then, there exists a random variable $X \in \mathbb{R}^d$ such that, for any pseudo-Lipschitz*

continuous $f : \mathbb{R}^d \to \mathbb{R}$,

$$\lim_{k\to\infty} \mathbb{E}f(X_k) = \mathbb{E}f(X) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i(N)). \tag{B.3}$$

In addition, the SE parameter limits $\overline{\alpha}_{k\ell}^{\pm}$ *and* $\overline{\gamma}_{k\ell}^{\pm}$ *converge to limits,*

$$\lim_{k\to\infty} \overline{\alpha}_{k\ell}^{\pm} = \overline{\alpha}_{\ell}^{\pm}, \quad \lim_{k\to\infty} \overline{\gamma}_{k\ell}^{\pm} = \overline{\gamma}_{\ell}^{\pm}. \tag{B.4}$$

The proposition shows that, under the convergence assumption, Assumption 2, we can take the limits as $k \to \infty$ of the random variables from the SE. To prove the proposition we first need the following simple lemma.

Lemma 2. *If* α_N *and* $\beta_k \in \mathbb{R}$ *are sequences such that*

$$\lim_{k\to\infty} \lim_{N\to\infty} |\alpha_N - \beta_k| = 0, \tag{B.5}$$

then, there exists a constant C *such that,*

$$\lim_{N\to\infty} \alpha_N = \lim_{k\to\infty} \beta_k = C. \tag{B.6}$$

In particular, the two limits in (B.6) *exist.*

Proof. For any $\epsilon > 0$, the limit (B.5) implies that there exists a $k_\epsilon (\uparrow \infty$ as $\epsilon \downarrow 0)$ such that for all $k > k_\epsilon$,

$$\lim_{N\to\infty} |\alpha_N - \beta_k| < \epsilon,$$

from which we can conclude,

$$\liminf_{N\to\infty} \alpha_N > \beta_k - \epsilon$$

for all $k > k_\epsilon$. Therefore,

$$\liminf_{N\to\infty} \alpha_N \geq \sup_{k\geq k_\epsilon} \beta_k - \epsilon.$$

Since this is true for all $\epsilon > 0$, it follows that

$$\liminf_{N\to\infty} \alpha_N \geq \limsup_{k\to\infty} \beta_k. \tag{B.7}$$

Similarly, $\limsup_{N\to\infty} \alpha_N \leq \inf_{k>k_\epsilon} \beta_k + \epsilon$, whereby

$$\limsup_{N\to\infty} \alpha_N \leq \liminf_{k\to\infty} \beta_k. \tag{B.8}$$

Equations (B.7) and (B.8) together show that the limits in (B.6) exists and are equal. □

Proof of Proposition 3 Let $f : \mathbb{R}^d \to \mathbb{R}$ be any pseudo-Lipschitz function of order 2, and define,

$$\alpha_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i(N)), \quad \beta_k = \mathbb{E}f(X_k). \tag{B.9}$$

Their difference can be written as,

$$\alpha_N - \beta_k = A_{N,k} + B_{N,k}, \tag{B.10}$$

where

$$A_{N,k} := \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i(N)) - f(\mathbf{x}_{k,i}(N)), \tag{B.11}$$

$$B_{N,k} := \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_{k,i}(N)) - \mathbb{E}f(X_k). \tag{B.12}$$

Since $\{x_{k,i}(N)\}$ converges $PL(2)$ to X_k, we have,

$$\lim_{N\to\infty} B_{N,k} = 0. \tag{B.13}$$

For the term $A_{N,k}$,

$$\begin{aligned}
|A_{N,k}| &\overset{(a)}{\leq} \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} |f(\mathbf{x}_i(N)) - f(\mathbf{x}_{k,i}(N))| \\
&\overset{(b)}{\leq} \lim_{N\to\infty} \frac{C}{N}\sum_{i=1}^{N} a_{ki}(N)(1+a_{ki}(N)) \\
&\overset{(c)}{\leq} C \lim_{N\to\infty} \sqrt{\frac{1}{N}\sum_{i=1}^{N} a_{ki}^2(N)} + \frac{1}{N}\sum_{i=1}^{N} a_{ki}^2(N) \\
&= C \lim_{N\to\infty} \epsilon_k(N)(1+\epsilon_k(N)), \qquad \text{(B.14)}
\end{aligned}$$

where (a) follows from applying the triangle inequality to the definition of $A_{N,k}$ in (B.11); (b) follows from the definition of pseudo-Lipschitz continuity in Definition 2.2, $C > 0$ is the Lipschitz contant and

$$a_{ki}(N) := \|\mathbf{x}_{k,i}(N) - \mathbf{x}_i(N)\|_2,$$

and (c) follows from the RMS-AM inequality:

$$\left(\frac{1}{N}\sum_{i=1}^{N} a_{ki}(N)\right)^2 \leq \frac{1}{N}\sum_{i=1}^{N} a_{ki}^2(N) =: \epsilon_k^2(N).$$

By (3.31), we have that,

$$\lim_{k\to\infty}\lim_{N\to\infty} \epsilon_k(N) = 0.$$

Hence, from (B.14), it follows that,

$$\lim_{k\to\infty}\lim_{N\to\infty} A_{N,k} = 0. \qquad \text{(B.15)}$$

Substituting (B.13) and (B.15) into (B.10) show that α_N and β_k satisfy (B.5). Therefore, applying Lemma 2 we have that for any pseudo-Lipschitz function $f(\cdot)$, there exists a limit

$\Phi(f)$ such that,

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} f(x_i(N)) = \lim_{k\to\infty}\mathbb{E}f(X_k) = \Phi(f). \tag{B.16}$$

In particular, the two limits in (B.16) exists. When restricted to the continuous, bounded functions with the $\|f\|_\infty$ norm, it is easy verified that $\Phi(f)$ is a positive, linear, bounded function of f, with $\Phi(1) = 1$. Therefore, by the Riesz representation theorem, there exists a random variable X such that $\Phi(f) = \mathbb{E}f(X)$. This fact in combination with (B.16) shows (B.3).

It remains to prove the parameter limits in (B.4). We prove the result for the parameter $\mathbf{w}\alpha_{k\ell}^+$. The proof for the other parameters are similar. Using Stein's lemma, it is shown in [97] that

$$\mathbf{w}\alpha_{k\ell}^+ = \frac{\mathbb{E}(\widehat{Z}_{k\ell}Q_{k\ell}^-)}{\mathbb{E}(Q_\ell^-)^2}. \tag{B.17}$$

Since the numerator and denominator of (B.17) are $PL(2)$ functions we have that the limit,

$$\begin{aligned}\overline{\alpha}_\ell^+ &:= \lim_{k\to\infty}\mathbf{w}\alpha_{k\ell}^+ = \lim_{k\to\infty}\frac{\mathbb{E}(\widehat{Z}_{k\ell}Q_{k\ell}^-)}{\mathbb{E}(Q_{k\ell}^-)^2}\\ &= \frac{\mathbb{E}(\widehat{Z}_\ell Q_\ell^-)}{\mathbb{E}(Q_\ell^-)^2},\end{aligned} \tag{B.18}$$

where $\widehat{Z}_\ell$ and Q_ℓ^- are the limits of $\widehat{Z}_{k\ell}$ and $Q_{k\ell}^-$. This completes the proof. □

B.2 Proof of Theorem 1

From Assumption 2, we know that for every N, every group of vectors $\mathbf{x}_k$ converge to limits, $\mathbf{x} := \lim_{k\to\infty}\mathbf{x}_k$. The parameters, $\gamma_{k\ell}^\pm$, also converge to limits $\overline{\gamma}_\ell^\pm := \lim_{k\to\infty}\gamma_{k\ell}^\pm$ for all ℓ. By the continuity assumptions on the functions $\mathbf{g}_\ell^\pm(\cdot)$, the limits $\mathbf{x}$ and $\overline{\gamma}_\ell^\pm$ are fixed points of the algorithms. For details of empirical convergence of fixed points, please see Appendix B.1

A proof similar to that in [98] shows that the fixed points $\widehat{\mathbf{z}}_\ell$ and $\widehat{\mathbf{p}}_\ell$ satisfy the KKT condition of the constrained optimization (3.24). This proves part (a).

The estimate $\widehat{\mathbf{w}}$ is the limit,

$$\widehat{\mathbf{w}} = \widehat{\mathbf{z}}_0 = \lim_{k\to\infty} \widehat{\mathbf{z}}_{k0}.$$

Also, the true parameter is $\mathbf{z}_0^0 = \mathbf{w}^0$. By Proposition 3 (see Appendix B.1 for details on convergence of fixed points), we have that the $PL(2)$ limits of these variables are

$$\lim_{N\to\infty} \{(\widehat{\mathbf{w}}, \mathbf{w}_0)\} \overset{PL(2)}{=} (\widehat{W}, W_0) := (\widehat{Z}_0, Z_0^0).$$

From line 13 of the SE Algorithm 3, we have

$$\widehat{W} = \widehat{Z}_0 = g_0^+(R_0^-, \overline{\gamma}_0^-) = \mathrm{prox}_{f_{\mathrm{in}}/\overline{\gamma}_0^-}(W^0 + Q_0^-).$$

This proves part (b).

To prove part (c), we use the limit

$$\lim_{N\to\infty} \{p_{0,n}^0, \widehat{p}_{0,n}\} \overset{PL(2)}{=} (P_0^0, \widehat{P}_0). \tag{B.19}$$

Since the fixed points are critical points of the constrained optimization (3.24), $\widehat{\mathbf{p}}_0 = \mathbf{V}_0\widehat{\mathbf{w}}$. We also have $\mathbf{p}_0^0 = \mathbf{V}_0\mathbf{w}^0$. Therefore,

$$\begin{aligned}\left[z_{\mathrm{ts}}^{(N)}\ \widehat{z}_{\mathrm{ts}}^{(N)}\right] &:= \mathbf{u}^{\mathsf{T}}\,\mathrm{Diag}(\mathbf{s}_{\mathrm{ts}})\mathbf{V}_0[\mathbf{w}^0\ \widehat{\mathbf{w}}] \\ &= \mathbf{u}^{\mathsf{T}}\,\mathrm{Diag}(\mathbf{s}_{\mathrm{ts}})[\mathbf{p}_0^0\ \widehat{\mathbf{p}}_0].\end{aligned} \tag{B.20}$$

Here, (N) in the subscript denotes the dependence on N. Since $\mathbf{u} \sim \mathcal{N}(0, \frac{1}{p}\mathbf{I})$, $[z_{\mathrm{ts}}^{(N)}\ \widehat{z}_{\mathrm{ts}}^{(N)}]$ is a zero-mean bivariate Gaussian with covariance matrix

$$\mathbf{M}^{(N)} = \tfrac{1}{p}\sum_{n=1}^{p} \begin{bmatrix} s_{\mathrm{ts},n}^2 p_{0,n}^0 p_{0,n}^0 & s_{\mathrm{ts},n}^2 p_{0,n}^0 \widehat{p}_{0,n} \\ s_{\mathrm{ts},n}^2 p_{0,n}^0 \widehat{p}_{0,n} & s_{\mathrm{ts},n}^2 \widehat{p}_{0,n}\widehat{p}_{0,n} \end{bmatrix}$$

The empirical convergence (B.19) yields the following limit,

$$\lim_{N\to\infty} \mathbf{M}^{(N)} = \mathbf{M} := \mathbb{E}\, S_{\text{ts}}^2 \begin{bmatrix} P_0^0 P_0^0 & P_0^0 \widehat{P}_0 \\ P_0^0 \widehat{P}_0 & \widehat{P}_0 \widehat{P}_0 \end{bmatrix}. \tag{B.21}$$

It suffices to show that the distribution of $[z_{\text{ts}}^{(N)}\, \widehat{z}_{\text{ts}}^{(N)}]$ converges to the distribution of $[Z_{\text{ts}}\, \widehat{Z}_{\text{ts}}]$ in the Wasserstein-2 metric as $N \to \infty$. See Section 2.2 on the equivalence of convergence in Wasserstein-2 metric and PL(2) convergence.

Now, Wassestein-2 distance between between two probability measures ν_1 and ν_2 is defined as

$$W_2(\nu_1, \nu_2) = \left(\inf_{\gamma\in\Gamma} \mathbb{E}_\gamma \left\| X_1 - X_2 \right\|^2 \right)^{1/2}, \tag{B.22}$$

where Γ is the set of probability distributions on the product space with marginals consistent with ν_1 and ν_2. For Gaussian measures $\nu_1 = \mathcal{N}(\mathbf{0}, \Sigma_1)$ and $\nu_2 = \mathcal{N}(\mathbf{0}, \Sigma_2)$ we have [57]

$$W_2^2(\nu_1, \nu_2) = \operatorname{tr}(\Sigma_1 - 2(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2})^{1/2} + \Sigma_2).$$

Therefore, for Gaussian distributions $\nu_1^{(N)} = \mathcal{N}(\mathbf{0}, \mathbf{M}^{(N)})$, and $\nu_2 = \mathcal{N}(\mathbf{0}, \mathbf{M})$, the convergence (B.21) implies $W_2(\nu_1^{(N)}, \nu_2) \to 0$, i.e., convergence in Wasserstein-2 distance. Hence,

$$(z_{\text{ts}}^{(N)}, \widehat{z}_{\text{ts}}^{(N)}) \xrightarrow{W_2} (Z_{\text{ts}}, \widehat{Z}_{\text{ts}}) \sim \mathcal{N}(0, \mathbf{M}),$$

where $\mathbf{M}$ is the covariance matrix in (B.21). Hence the convergence holds in the PL(2) sense.

Hence the asymptotic generalization error (3.17) is

$$\begin{aligned} \mathcal{E}_{\text{ts}} &:= \lim_{N\to\infty} \mathbb{E} f_{\text{ts}}(\widehat{y}_{\text{ts}}, y_{\text{ts}}) \\ &\overset{(a)}{=} \lim_{N\to\infty} \mathbb{E} f_{\text{ts}}(\phi_{\text{out}}(z_{\text{ts}}^{(N)}, D), \phi(\widehat{z}_{\text{ts}}^{(N)})) \\ &\overset{(b)}{=} \mathbb{E} f_{\text{ts}}(\phi_{\text{out}}(Z_{\text{ts}}, D), \phi(\widehat{Z}_{\text{ts}})), \end{aligned} \tag{B.23}$$

where (a) follows from (3.3); and step (b) follows from continuity assumption in Assumption 1(b) along with the definition of PL(2) convergence in section 2.2. This proves part (c).

B.2.1 Formula for M

It is useful to derive expressions for the entries the covariance matrix $\mathbf{M}$ in (B.21). We use these to calculate the values in Section 3.7.

For the term m_{11},

$$m_{11} = \mathbb{E}\, S_{\text{ts}}^2 (P_0^0)^2 = \mathbb{E}\, S_{\text{ts}}^2 \mathbb{E}(P_0^0)^2 = \mathbb{E}\, S_{\text{ts}}^2 \cdot k_{11}, \tag{B.24}$$

where we have used the fact that $P_0^0 \perp\!\!\!\perp (S_{\text{ts}}, S_{\text{tr}})$. Next, $m_{12} = \mathbb{E}\, S_{\text{ts}}^2\ P_0^0 \widehat{P}_0$, where,

$$\begin{aligned} \widehat{P}_0 &= g_1^-(P_0^0 + P_0^+, Z_1^0 + Q_1^-, \overline{\gamma}_0^+, \overline{\gamma}_1^-, S_{\text{tr}}^-) \\ &= \frac{\mathbf{w}\gamma_0^+ P_0^+ + S_{\text{tr}}\mathbf{w}\gamma_1^- Q_1^-}{\mathbf{w}\gamma_0^+ + S_{\text{tr}}^2 \mathbf{w}\gamma_1^-} + P_0^0, \end{aligned} \tag{B.25}$$

and (P_0^0, P_0^+, Q_0^-) are independent of $(S_{\text{tr}}, S_{\text{ts}})$. Hence,

$$\begin{aligned} m_{12} &= \mathbb{E}\, S_{\text{ts}}^2 \cdot \mathbb{E}(P_0^0)^2 + \mathbb{E}\frac{S_{\text{ts}}^2 \overline{\gamma}_0^+}{\overline{\gamma}_0^+ + S_{\text{tr}}^2 \overline{\gamma}_1^-} \mathbb{E}[P_0^0 P_0^+] \\ &= m_{11} + \mathbb{E}\left(\frac{S_{\text{ts}}^2 \mathbf{w}\gamma_0^+}{\mathbf{w}\gamma_0^+ + S_{\text{tr}}^2 \mathbf{w}\gamma_1^-} \right) \cdot k_{12}, \end{aligned} \tag{B.26}$$

. Note that $\mathbb{E}[P_0^0 Q_1^-] = 0$ and $\mathbf{K}_0^+$ is the covariance matrix of (P_0^0, P_0^+) from line 21.

Finally, for m_{22} we have,

$$\begin{aligned} m_{22} &= \mathbb{E}\, S_{\text{ts}}^2 \widehat{P}_0 \widehat{P}_0 \\ &= \mathbb{E}\left(\frac{S_{\text{ts}}\overline{\gamma}_0^+}{\overline{\gamma}_0^+ + S_{\text{tr}}^2\overline{\gamma}_1^-}\right)^2 \mathbb{E}(P_0^+)^2 + \mathbb{E}\left(\frac{S_{\text{ts}}S_{\text{tr}}\overline{\gamma}_1^-}{\overline{\gamma}_0^+ + S_{\text{tr}}^2\overline{\gamma}_1^-}\right)^2 \mathbb{E}(Q_1^-)^2 \\ &+ \mathbb{E}\, S_{\text{ts}}^2 \mathbb{E}(P_0^0)^2 + 2\mathbb{E}\frac{\overline{\gamma}_0^+ S_{\text{ts}}^2}{\overline{\gamma}_0^+ + \overline{\gamma}_1^- S_{\text{tr}}^2} \cdot \mathbb{E}\, P_0^0 P_0^+ \\ &= k_{22}\mathbb{E}\left(\frac{S_{\text{ts}}\overline{\gamma}_0^+}{\overline{\gamma}_0^+ + S_{\text{tr}}^2\overline{\gamma}_1^-}\right)^2 + \tau_1^-\mathbb{E}\left(\frac{S_{\text{ts}}S_{\text{tr}}\overline{\gamma}_1^-}{\overline{\gamma}_0^+ + S_{\text{tr}}^2\overline{\gamma}_1^-}\right)^2 - m_{11} + 2m_{12}. \end{aligned} \tag{B.27}$$

B.3 Proofs of Special Cases (Section 3.7)

B.3.1 Proof of Corollary 1

This follows directly from the following observation:

$$\begin{aligned} \mathcal{E}_{\text{ts}}^{\mathsf{SLR}} &= \mathbb{E}(Z_{\text{ts}} + D - \widehat{Z}_{\text{ts}})^2 = \mathbb{E}(Z_{\text{ts}} - \widehat{Z}_{\text{ts}})^2 + \mathbb{E}\, D^2 \\ &= m_{11} + m_{22} - 2m_{12} + \sigma_d^2. \end{aligned}$$

Substituting equation (B.27) proves the claim.

B.3.2 Proof of Corollary 2

We are interested in identifying the following constants appearing in Corollary 1:

$$\mathbf{K}_0^+, \tau_1^-, \overline{\gamma}_0^+, \overline{\gamma}_1^-.$$

These quantities are obtained as fixed points of the State Evolution Equations in Algo. 3. We explain below how to obtain expressions for these constants. Since these are fixed points we ignore the subscript k corresponding to the iteration number in Algo. 3.

In the case of problem (3.48), the maps $\text{prox}_{f_{\text{in}}}$ and $\text{prox}_{f_{\text{out}}}$, i.e., g_0^+ and g_3^- respectively,

can be expressed as closed-form formulae. This leads to simplification of the SE equations as explained below.

We start by looking at the *forward pass* (finding quantities with superscript '+') of Algorithm 3 for different layers and then the *backward pass* (finding quantities with superscript '-') to get the parameters $\{\mathbf{K}_\ell^+, \tau_\ell^-, \mathbf{w}\alpha_\ell^\pm, \mathbf{w}\gamma_\ell^\pm\}$ for $\ell = 0, 1, 2$.

To begin with, notice that $f_{\text{in}}(w) = \frac{\lambda}{2}w^2$, and therefore the denoiser $g_0^+(\cdot)$ in (3.36) is simply,

$$g_0^+(r_0^-, \gamma_0^-) = \tfrac{\gamma_0^-}{\gamma_0^- + \lambda/\beta} r_0^-, \quad \text{and} \quad \tfrac{\partial g_0^+}{\partial r_0^-} = \tfrac{\gamma_0^-}{\gamma_0^- + \lambda/\beta}$$

Using the random variable R_0^- and substituting in the expression of the denoiser to get $\widehat{Z}_0$, we can now calculate $\overline{\alpha}_0^+$ using lines 18 and 20,

$$\overline{\alpha}_0^+ = \tfrac{\overline{\gamma}_0^-}{\overline{\gamma}_0^- + \lambda/\beta}, \qquad \overline{\gamma}_0^+ = \lambda/\beta. \tag{B.28}$$

Similarly, we have $f_{\text{out}}(p_2) = \frac{1}{2}(p_2 - y)^2$, whereby the output denoiser $g_3^-(\cdot)$ in the last layer for ridge regression is given by,

$$g_3^-(r_2^+, \gamma_2^+, y) = \frac{\gamma_2^+ r_2^+ + y}{\gamma_2^+ + 1}. \tag{B.29}$$

By substituting this denoiser in line 27 of the algorithm we get $\widehat{P}_2^-$ and thus, following the lines 32-35 of the algorithm we have

$$\overline{\alpha}_2^- = \tfrac{\overline{\gamma}_2^+}{\overline{\gamma}_2^+ + 1}, \qquad \text{whereby} \quad \overline{\gamma}_2^- = 1. \tag{B.30}$$

Having identified these constants $\overline{\alpha}_0^+, \overline{\gamma}_0^+, \overline{\alpha}_2^-, \overline{\gamma}_2^-$, we will now sequentially identify the

quantities

$$(\overline{\alpha}_0^+, \overline{\gamma}_0^+) \to \mathbf{K}_0^+ \to (\overline{\alpha}_1^+, \overline{\gamma}_1^+) \to \mathbf{K}_1^+ \to (\overline{\alpha}_2^+, \overline{\gamma}_2^+) \to \mathbf{K}_2^+,$$

in the forward pass, and then the quantities

$$\tau_0^- \leftarrow (\overline{\alpha}_0^-, \overline{\gamma}_0^-) \leftarrow \tau_1^- \leftarrow (\overline{\alpha}_1^-, \overline{\gamma}_1^-) \leftarrow \tau_2^- \leftarrow (\overline{\alpha}_2^-, \overline{\gamma}_2^-),$$

in the backward pass. Note that we also have

$$\overline{\alpha}_\ell^+ + \overline{\alpha}_\ell^- = 1. \tag{B.31}$$

Forward Pass: Observe that $\mathbf{K}_0^+ = \mathrm{Cov}(Z_0, Q_0^+)$. Now, from line 19, on simplification we get $Q_0^+ = -W_0^0$ whereby,

$$\mathbf{K}_0^+ = \mathrm{var}(W^0) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{B.32}$$

Notice that from line 21, the pair (P_0^0, P_0^+) is jointly Gaussian with covariance matrix $\mathbf{K}_0^+$. But the above equation means that $P_0^+ = -P_0^0$, whereby $R_0^+ = 0$ from line 15.

Now, the linear denoiser $g_1^+(\cdot)$ is defined as in (3.39a). Note that since we are considering i.i.d Gaussian features for this problem, the random variable S_{tr} in this layer is a constant σ_{tr}. Therefore, similar to layer $\ell = 0$ by evaluating lines 15-21 of the algorithm we get $Q_1^+ = -Z_1^0$, whereby

$$\overline{\alpha}_1^+ = \frac{\sigma_{\mathrm{tr}}^2 \overline{\gamma}_1^-}{\mathbf{w}\gamma_0^+ + \sigma_{\mathrm{tr}}^2 \overline{\gamma}_1^-},\ \overline{\gamma}_1^+ = \frac{\mathbf{w}\gamma_0^+}{\sigma_{\mathrm{tr}}^2} = \frac{\lambda}{\sigma_{\mathrm{tr}}^2 \beta},\ \mathbf{K}_1^+ = \sigma_{\mathrm{tr}}^2 \mathbf{K}_0^+. \tag{B.33}$$

Observe that this means

$$P_1^+ = -P_1^0. \tag{B.34}$$

Backward Pass: Since $Y = \phi_{\text{out}}(P_2^0, D) = P_2^0 + D$, line 33 of algorithm on simplification yields $P_2^- = D$, whereby we can get τ_2^-,

$$\tau_2^- = \mathbb{E}(P_2^-)^2 = \mathbb{E}[D^2] = \sigma_d^2. \tag{B.35}$$

To calculate the terms $(\overline{\alpha}_1^-, \overline{\gamma}_1^-)$, we use the decoiser g_2^- defined in (3.39a) for line 30 of Algorithm 3 to get $\widehat{P}_1$,

$$\widehat{P}_1 = \frac{\overline{\gamma}_1^+ R_1^+ + S_{\text{mp}}^- \overline{\gamma}_2^- R_2^-}{\overline{\gamma}_1^+ + (S_{\text{mp}}^-)^2 \overline{\gamma}_2^-} = \frac{S_{\text{mp}}^-(S_{\text{mp}}^+ P_1^0 + Q_2^-)}{\overline{\gamma}_1^+ + (S_{\text{mp}}^-)^2}, \tag{B.36}$$

where we have used $\overline{\gamma}_2^- = 1$, $R_1^+ = P_1^0 + P_1^+ = 0$ due to (B.34), and $R_2^- = Z_2^0 + Q_2^- = S_{\text{mp}}^+ P_1^0 + Q_2^-$ from lines 15, 29 and 3 respectively. We then calculate $\overline{\alpha}_1^-$ and $\overline{\gamma}_1^-$ as

$$\overline{\alpha}_1^- = \mathbb{E}\frac{\partial g_2^-}{\partial P_1^+} = \mathbb{E}\frac{\overline{\gamma}_1^+}{\overline{\gamma}_1^+ + (S_{\text{mp}}^-)^2}.$$

This gives,

$$\overline{\alpha}_1^- = \begin{cases} \frac{\lambda}{\sigma_{\text{tr}}^2 \beta} G & \beta < 1 \\ (1 - \frac{1}{\beta}) + \frac{1}{\beta}\frac{\lambda}{\sigma_{\text{tr}}^2 \beta} G & \beta \geq 1 \end{cases}. \tag{B.37}$$

Here, in the overparameterized case ($\beta > 1$), the denoiser g_2^- outputs R_1^+ with probability $1 - \frac{1}{\beta}$ and $\frac{\lambda}{\sigma_{\text{tr}}^2 \beta} G$ with probability $\frac{1}{\beta}$.

Next, from line 34 we get,

$$\overline{\gamma}_1^- = (\tfrac{1}{\overline{\alpha}_1^-} - 1)\overline{\gamma}_1^+ = \begin{cases} \frac{1}{G} - \frac{\lambda}{\sigma_{\text{tr}}^2 \beta} & \beta < 1 \\ \frac{\lambda}{\sigma_{\text{tr}}^2 \beta}(\frac{1}{G} - \frac{\lambda}{\sigma_{\text{tr}}^2 \beta}) & \\ \frac{\beta - 1}{G} + \frac{\lambda}{\sigma_{\text{tr}}^2 \beta} & \beta > 1 \end{cases} \tag{B.38}$$

Now from line 33 and equation (B.31) we get,

$$\begin{aligned}\overline{\alpha}_1^+ P_1^- &= \widehat{P}_1 - P_1^0 - \overline{\alpha}_1^- P_1^+ \overset{\text{(a)}}{=} \widehat{P}_1 - \overline{\alpha}_1^+ P_1^0 \\ &\overset{\text{(b)}}{=} \underbrace{\left(\frac{S_{\text{mp}}^- S_{\text{mp}}^+}{\frac{\lambda}{\sigma_{\text{tr}}^2\beta} + (S_{\text{mp}}^-)^2} - \overline{\alpha}_1^+ \right)}_{A} P_1^0 + \underbrace{\frac{S_{\text{mp}}^-}{\frac{\lambda}{\sigma_{\text{tr}}^2\beta} + (S_{\text{mp}}^-)^2}}_{B} Q_2^- \end{aligned} \tag{B.39}$$

where (a) follows from (B.34) and (B.31), and (b) follows from (B.36). From this one can obtain $\tau_1^- = \mathbb{E}(P_1^-)^2$ which can be calculated using the knowledge that P_1^0, Q_2^- are independent Gaussian with covariances $\mathbb{E}(P_1^0)^2 = \sigma_{\text{tr}}^2 \text{Var}(W^0)$, $\mathbb{E}(Q_2^-)^2 = \sigma_d^2$. Further, P_1^0, Q_2^- are independent of $(S_{\text{mp}}^+, S_{\text{mp}}^-)$.

Observe that by (B.39) we have

$$\tau_1^- = \frac{1}{(\overline{\alpha}_1^+)^2} \left(\mathbb{E}(A^2)\sigma_{\text{tr}}^2 \text{Var}(W^0) + \mathbb{E}(B^2)\sigma_d^2 \right). \tag{B.40}$$

with some simplification we get

$$\mathbb{E}(A^2) = (\frac{\lambda}{\sigma_{\text{tr}}^2\beta})^2 G' - (\frac{\lambda}{\sigma_{\text{tr}}^2\beta} G)^2, \tag{B.41a}$$

$$\mathbb{E}(B^2) = G - \frac{\lambda}{\sigma_{\text{tr}}^2\beta} G', \tag{B.41b}$$

where $G = G_{\text{mp}}(-\frac{\lambda}{\sigma_{\text{tr}}^2\beta})$, with G_{mp} given in Section 2.2.1, and G' is the derivative of G_{mp} calculated at $-\frac{\lambda}{\sigma_{\text{tr}}^2\beta}$.

Now consider the **under-parametrized** case ($\beta < 1$):

Let $u = -\frac{\lambda}{\sigma_{\text{tr}}^2\beta}$ and $z = G_{\text{mp}}(u)$. In this case we have

$$\overline{\alpha}_1^+ = 1 - \frac{\lambda}{\sigma_{\text{tr}}^2\beta} G = 1 + uz. \tag{B.42}$$

Note that,

$$G_{\text{mp}}^{-1}(z) = u \quad \overset{\text{(a)}}{\Rightarrow} R_{\text{mp}}(z) + \frac{1}{z} = u$$
$$\overset{\text{(b)}}{\Rightarrow} \frac{1}{1-\beta z} + \frac{1}{z} = u, \tag{B.43a}$$

where $R_{\text{mp}}(.)$ is the R-transform defined in [122] and (a) follows from the relationship between the R- and Stieltjes-transform and (b) follows from the fact that for Marchenko-Pastur distribution we have $R_{\text{mp}}(z) = \frac{1}{1-z\beta}$. Therefore,

$$G_{\text{mp}}(\frac{1}{1-\beta z} + \frac{1}{z}) = z$$
$$\Rightarrow G'_{\text{mp}}(\frac{1}{1-\beta z} + \frac{1}{z}) = G' = \frac{1}{\frac{\beta}{(1-\beta z)^2} - \frac{1}{z^2}}. \tag{B.44}$$

For the **over-parametrized** case ($\beta > 1$) we have:

$$\overline{\alpha}_1^+ = \tfrac{1}{\beta}(1 + \tfrac{\lambda}{\sigma_{\text{tr}}^2 \beta} G) = \frac{1-uz}{\beta}. \tag{B.45}$$

In this case, as mentioned in Section 2.2.1 and following the results from [122], the measure μ_β scales with β and thus $R_{\text{mp}}(z) = \frac{\beta}{1-z}$. Therefore, similar to (B.43a), z satisfies

$$\frac{\beta}{1-z} + \frac{1}{z} = u \quad \Rightarrow G' = \frac{1}{\frac{\beta}{(1-z)^2} - \frac{1}{z^2}}. \tag{B.46}$$

Now τ_1^- can be calculated as follows:

$$\tau_1^- = \eta^2 \Big(u^2 z^2 \sigma_{\text{tr}}^2 var(W^0)(\kappa - 1) + \sigma_d^2 z(uz\kappa + 1) \Big) \tag{B.47}$$

where

$$\eta = \begin{cases} \frac{1}{(1+uz)} & \beta < 1 \\ \frac{\beta}{(1-uz)} & \beta \geq 1 \end{cases}, \quad \kappa = \begin{cases} \frac{(1-\beta z)^2}{\beta z^2 - (1-\beta z)^2} & \beta < 1 \\ \frac{(1-z)^2}{\beta z^2 - (1-z)^2} & \beta \geq 1 \end{cases}, \tag{B.48}$$

and z is the solution to the fixed points

$$\begin{cases} \frac{1}{1-\beta z} + \frac{1}{z} = u & \beta < 1 \\ \frac{\beta}{1-z} + \frac{1}{z} = u & \beta \geq 1 \end{cases}. \tag{B.49}$$

B.3.3 Proof of Corollary 3

We calculate the parameters $\overline{\gamma}_0^+, \overline{\gamma}_1^-$, k_{22} and τ_1^- when $\lambda \to 0^+$. Before starting off, we note that

$$G_0 := \lim_{z \to 0^+} G_{\text{mp}}(-z) = \begin{cases} \frac{\beta}{1-\beta} & \beta < 1 \\ \frac{\beta}{\beta - 1} & \beta > 1 \end{cases}, \tag{B.50}$$

as described in Section 2.2.1. Following the derivations in Corollary 2, we have

$$\overline{\gamma}_0^+ = \lambda/\beta, \quad k_{22} = \text{Var}(W^0) \tag{B.51}$$

Now for $\lambda \to 0^+$, we have

$$1 - \overline{\alpha}_1^- = \begin{cases} 1 & \beta < 1 \\ \frac{1}{\beta} & \beta \geq 1 \end{cases}, \quad \overline{\gamma}_1^- = \begin{cases} \frac{1}{G_0} = \frac{1-\beta}{\beta} & \beta < 1 \\ \frac{\lambda}{(\beta-1)\sigma_{\text{tr}}^2 \beta} & \beta > 1 \end{cases}, \tag{B.52}$$

Using this in simplifying (B.39) for $\lambda \to 0^+$, we get

$$\tau_1^- = \mathbb{E}(P_1^-)^2 = \begin{cases} \sigma_d^2 G_0 & \beta < 1 \\ \beta \sigma_d^2 G_0 + \sigma_{\text{tr}}^2 \text{Var}(W^0)(\beta - 1) & \beta \geq 1 \end{cases}$$

where during the evaluation of $\mathbb{E}\left(\frac{S_{\text{mp}}^-}{\overline{\gamma}_1^+ + (S_{\text{mp}}^-)^2}\right)^2$, for the case of $\beta > 1$, we need to account for the point mass at 0 for S_{mp}^- with weight $1 - \frac{1}{\beta}$.

Next, notice that

$$a := \frac{\overline{\gamma}_0^+ \sigma_{\text{tr}}}{\overline{\gamma}_0^+ + \overline{\gamma}_1^- \sigma_{\text{tr}}^2} = \begin{cases} 0 & \beta < 1 \\ (1 - \frac{1}{\beta})\sigma_{\text{tr}} & \beta \geq 1 \end{cases},$$

and,

$$b := \frac{\overline{\gamma}_1^- \sigma_{\text{tr}}^2}{\overline{\gamma}_0^+ + \overline{\gamma}_1^- \sigma_{\text{tr}}^2} = \begin{cases} 1 & \beta < 1 \\ \frac{1}{\beta} & \beta \geq 1 \end{cases},$$

Thus applying Corollary 1, we get

$$\begin{aligned} \mathcal{E}_{\text{ts}}^{\mathsf{RR}} &= a^2 k_{22} + b^2 \tau_1^- + \sigma_d^2 \\ &= \begin{cases} \frac{1}{1-\beta}\sigma_d^2 & \beta < 1 \\ \frac{\beta}{\beta - 1}\sigma_d^2 + (1 - \frac{1}{\beta})\sigma_{\text{tr}}^2 \text{Var}(W^0) & \beta \geq 1 \end{cases}. \end{aligned}$$

This proves the claim.

Appendix C

Proofs for Chapter 4

C.1 Proof of Theorem 4

First consider the case when $n = 1$ with scalar inputs and outputs. Let $\theta_c = (w_c, f_c, b_c, c_c)$ be the parameters of a contractive RNN with $f_c = c_c = 1$, $b_c = 0$ and $w_c \in (0, 1)$. Hence, the contractive RNN is given by

$$h_{c,t} = \phi(w_c h_{c,t-1} + x_t), \quad y_t = h_{c,t}, \tag{C.1}$$

and $\phi(z) = \max\{0, z\}$ is the ReLU activation. Suppose θ_u are the parameters of an equivalent URNN. If θ has less than $2n = 2$ states, it must have $n = 1$ state. Let the equivalent URNN be

$$h_{u,t} = \phi(w_u h_{u,t-1} + f_u x_t + b_u), \quad y_t = c_u h_{u,t}, \tag{C.2}$$

for some parameters $\theta_u = (w_u, f_u, b_u, c_u)$. Since w_u is orthogonal, either $w_u = 1$ or $w_u = -1$. Also, either $f_u > 0$ or $f_u < 0$. First, consider the case when $w_u = 1$ and $f_u > 0$. Then, there exists a large enough input x_t such that for all time steps k, both systems are operating in

the active phase of ReLU. Therefore, we have two equivalent linear systems,

$$\text{contractive RNN:} \quad h_{c,t} = w_c h_{c,t-1} + x_t, \quad y_t = h_{c,t} \tag{C.3}$$

$$\text{URNN:} \quad h_{u,t} = h_{u,t-1} + f_u x_t + b_u, \quad y_t = c_u h_{u,t}. \tag{C.4}$$

In order to have identical input-output mapping for these linear systems for all x, it is required that $w_c = 1$, which is a contradiction. The other cases $w_c = -1$ and $f_u < 0$ can be treated similarly. Therefore, at least $n = 2$ states are needed for the URNN to match the contractive RNN with $n = 1$ state.

For the case of general n, consider the contractive RNN,

$$\mathbf{h}_t = \phi(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{F}\mathbf{x}_t + \mathbf{b}), \quad \mathbf{y}_t = \mathbf{C}\mathbf{h}_t, \tag{C.5}$$

where $\mathbf{W} = \text{Diag}(w_c, w_c, ..., w_c)$, $\mathbf{F} = \text{Diag}(f_c, f_c, ..., f_c)$, $\mathbf{b} = b_c \mathbf{1}_{n\times 1}$, and $\mathbf{C} = \text{Diag}(c_c, c_c, ..., c_c)$. This system is separable in that if $\mathbf{y} = G(\mathbf{x})$ then $y_i = G(x_i, \theta_c)$ for each input i. A URNN system will need 2 states for each scalar system requiring a total of $2n$ states.

C.2 Proof of Theorem 5

We use the same scalar contractive RNN (C.1), but with a sigmoid activation $\phi(z) = 1/(1 + e^{-z})$. Let $\boldsymbol{\theta} = (\mathbf{W}_u, \mathbf{f}_u, \mathbf{c}_u, \mathbf{b}_u)$ be the parameters of any URNN with scalar input and outputs. Suppose the URNN is controllable and observable at an input value x^*. Let h_c^* and $\mathbf{h}_u^*$ be, respectively, the fixed points of the hidden states for the contractive RNN and URNN:

$$\text{contractive RNN:} \quad h_c^* = \phi(w_c h_c + x^*), \tag{C.6}$$

$$\text{URNN:} \quad \mathbf{h}_u^* = \phi(\mathbf{W}_u \mathbf{h}_u + \mathbf{f}_u x^* + \mathbf{b}_u). \tag{C.7}$$

We take the linearizations [124] of each system around its fixed point and apply a small perturbation Δx around x^*. Therefore, we have two linear systems with identical input-output mapping given by,

$$\text{contractive RNN:} \quad \Delta h_{c,t} = d_c(w_c \Delta h_{t-1} + \Delta x_t), \quad y_t = \Delta h_{c,t} + h_c^*, \tag{C.8}$$

$$\text{URNN:} \quad \Delta \mathbf{h}_{u,t} = \mathbf{D}_u(\mathbf{W}_u \Delta \mathbf{h}_{u,t-1} + \mathbf{f}_u^\mathsf{T} \Delta x_t), \quad y_t = \mathbf{c}_u^\mathsf{T} \Delta \mathbf{h}_u + \mathbf{c}_u^\mathsf{T} \mathbf{h}_u^*, \tag{C.9}$$

where

$$d_c = \phi'(z_c^* = w_c h_c^* + x^*), \quad \mathbf{D}_u = \phi'(\mathbf{W}_u \mathbf{h}_u^* + \mathbf{f}_u x^* + \mathbf{b}_u),$$

are the derivatives of the activations at the fixed points. Since both systems are controllable and observable, their dimensions must be the same and the eigenvalues of the transition matrix must match. In particular, the URNN must be scalar, so $\mathbf{W}_u = w_u$ for some scalar w_u. For orthogonality, either $w_u = 1$ or $w_u = -1$. We look at the $w_u = 1$ case; the $w_u = -1$ case is similar. Since the eigenvalues of the transition matrix must match we have,

$$d_c w_c = d_u \Rightarrow \phi'(w_c h_c^* + x^*) w_c = \phi'(h_u^* + f_u x^* + b_u). \tag{C.10}$$

where h_u^* and h_c^* are the solutions to the fixed point equations:

$$h_c^* = \phi(w_c h_c + x^*), \quad h_u^* = \phi(h_u^* + f_b x^* + b_u). \tag{C.11}$$

Also, since two systems have the same output,

$$h_c^* = c_u h_u^*. \tag{C.12}$$

Now, (C.10) must hold at any input x^* where the URNN is controllable and observable. If the URNN is controllable and observable at some x^*, it is is controllable and observable in a neighborhood of x^*. Hence, (C.10) and (C.12) holds in some neighborhood of x^*. To write

this mathematically, define the functions,

$$g_c(x^*) := \begin{bmatrix} w_c\phi'(w_c h_c^* + x^*) \\ h_c^* \end{bmatrix}, \quad g_u(x^*) := \begin{bmatrix} \phi'(h_u^* + f_u x^* + b_u) \\ c_u h_u^* \end{bmatrix}, \tag{C.13}$$

where, for a given x^*, h_u^* and h_c^* are the solutions to the fixed point equations (C.11). We must have that $g_c(x^*) = g_u^*(x^*)$ for all x^* in some neighborhood. Taking derivatives of (C.13) and using the fact that $\phi(z)$ being a sigmoid, one can show that this matching can only occur when,

$$w_c = 1, \quad b_u = 0, \quad c_u = 1.$$

This is a contradiction since we have assumed that the RNN system is contractive which requires $|w_c| = 1$.

Appendix D

Proofs for Chapter 5

D.1 Proof of Proposition 1

Suppose we are given a convolutional model (5.5) with impulse response coefficients $\mathbf{L}_t$, $t = 0, \ldots, T-1$. It is well-known from linear systems theory [66] that linear time-invariant systems are input-output equivalent if and only if they have the same impulse response coefficients. So, we simply need to find matrices $(\mathbf{W}, \mathbf{F}, \mathbf{C})$ satisfying (5.9). First consider the single input single output (SISO) case where $n_x = n_y = 1$. Take any set of real non-zero scalars λ_i, $i = 0, \ldots, T-1$, that are distinct and set

$$\mathbf{W} = \text{diag}(\lambda_0, \ldots, \lambda_{T-1}), \quad \mathbf{F} = \mathbf{1}_T, \tag{D.1}$$

so there are $n = T$ hidden states. Then, for any t,

$$(\mathbf{C}\mathbf{W}^t\mathbf{F}) = \sum_{k=0}^{T-1} \mathbf{C}_k \lambda_k^t. \tag{D.2}$$

Equivalently, the impulse response coefficients in (5.9) are given by,

$$[\mathbf{L}_0, \cdots, \mathbf{L}_{T-1}] = \mathbf{C}\mathbf{V}, \tag{D.3}$$

where $\mathbf{V}$ is the Vandermode matrix $V_{jt} = \lambda_j^t$. Since the values λ_j are distinct, $\mathbf{V}$ is invertible and we can find a vector $\mathbf{C}$ matching arbitrary impulse response coefficients. Thus, when $n_x = n_y = 1$, we can find a linear RNN with at most $n = T$ hidden states that match the first T impulse response coefficients. To extend to the case of arbitrary n_x and n_y, we simply create $n_x n_y$ systems, one for each input-output component pair. Since each system will have T hidden states, the total number of states would be $n = T n_x n_y$.

D.2 Proof of Theorem 6

Given $\boldsymbol{y}_t = \sum_{j=0}^{t} \sqrt{\rho_j} \boldsymbol{\theta}_j \boldsymbol{x}_{t-j}$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{T-1})$, we consider a perturbation in $\boldsymbol{\theta}$, namely $\boldsymbol{\Delta}_\theta$. Therefore,

$$\widetilde{\boldsymbol{y}}_t = \sum_{j=0}^{t} \sqrt{\rho_j} \boldsymbol{\Delta}_{\theta j} \boldsymbol{x}_{t-j} \tag{D.4}$$

and the NTK for this model is given by

$$K_{t,s}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{\Delta}_\theta \in \mathcal{T}_\theta} \widetilde{\boldsymbol{y}}_t(\boldsymbol{\Delta}_\theta) \widetilde{\boldsymbol{y}}'_s(\boldsymbol{\Delta}_\theta)^\mathsf{T}. \tag{D.5}$$

where $\mathcal{T}_\theta$ is the standard basis for the parameter space. The following lemma shows this sum can be calculated as an expectation over a Gaussian random variable.

Lemma 3. *Let $\mathcal{V}$ be a finite dimensional Hilbert space and $\mathcal{W} = \mathbb{R}^m$ with the standard inner product and let $T, T' : \mathcal{V} \to \mathcal{W}$ be linear transformations. Let $\{\mathbf{v}_i\}_{i=1}^n$ be an ordered orthonormal basis for $\mathcal{V}$. Then we have*

$$\sum_{i=1}^{n} T(\mathbf{v}_i)(T'(\mathbf{v}_i))^\mathsf{T} = \mathbb{E}_{\boldsymbol{\alpha} \sim \mathcal{N}(0, \mathbf{I}_n)} \left[T\big(\sum_{i=1}^{n} \alpha_i \mathbf{v}_i\big) \Big(T'\big(\sum_{i=1}^{n} \alpha_i \mathbf{v}_i\big) \Big)^\mathsf{T} \right]. \tag{D.6}$$

Proof.

$$\mathbb{E}_{\boldsymbol{\alpha}\sim\mathcal{N}(0,\mathbf{I}_n)}\left[T\big(\sum_{i=1}^{n}\alpha_i\mathbf{v}_i\big)\Big(T'\big(\sum_{j=1}^{n}\alpha_j\mathbf{v}_j\big)\Big)^{\mathsf{T}}\right]=\mathbb{E}_{\boldsymbol{\alpha}\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\sum_{i,j=1}^{n}\alpha_i\alpha_j T\big(\mathbf{v}_i\big)T'\big(\mathbf{v}_j\big)^{\mathsf{T}}\right]$$
$$=\sum_{i=1}^{n}T(\mathbf{v}_i)(T'(\mathbf{v}_i))^{\mathsf{T}}. \tag{D.7}$$

□

Since $\widetilde{\boldsymbol{y}}_t(\boldsymbol{\Delta}_\theta)$ is a linear operator, by applying Lemma 3 we have,

$$\begin{aligned}K_{t,s}(\mathbf{x},\mathbf{x}')&=\mathbb{E}_{\boldsymbol{\Delta}_\theta\sim\mathcal{N}(0,1),\text{i.i.d.}}\ [\widetilde{\boldsymbol{y}}_t(\boldsymbol{\Delta}_\theta)\widetilde{\boldsymbol{y}}'_s(\boldsymbol{\Delta}_\theta)^{\mathsf{T}}]\\&=\mathbb{E}_{\boldsymbol{\Delta}_\theta\sim\mathcal{N}(0,1),\text{i.i.d.}}\ \left[(\sum_{j=0}^{t}\sqrt{\rho_j}\boldsymbol{\Delta}_{\theta j}\boldsymbol{x}_{t-j})(\sum_{k=0}^{t}\sqrt{\rho_k}\boldsymbol{\Delta}_{\theta k}\boldsymbol{x}'_{s-k})^{\mathsf{T}}\right]\\&=\mathbb{E}_{\boldsymbol{\Delta}_\theta\sim\mathcal{N}(0,1),\text{i.i.d.}}\ \left[(\sum_{j=0}^{t}\rho_j\ \boldsymbol{\Delta}_{\theta j}\boldsymbol{x}_{t-j}{\boldsymbol{x}'_{s-j}}^{\mathsf{T}}\boldsymbol{\Delta}_{\theta j}^{\mathsf{T}})\right]\end{aligned}$$

Therefore,

$$\begin{aligned}\Big(K_{t,s}(\mathbf{x},\mathbf{x}')\Big)_{m,m'}&=\mathbb{E}_{\boldsymbol{\Delta}_\theta\sim\mathcal{N}(0,1),\text{i.i.d.}}\ \left[\sum_{j=0}^{t}\rho_j\ \sum_{k,k'}(\Delta_{\theta j})_{m,k}x_{t-j,k}x'_{s-j,k'}(\Delta_{\theta j})_{k',m'}\right]\\&=(\sum_{j=0}^{t}\rho_j\ \boldsymbol{x}_{t-j}^{\mathsf{T}}\boldsymbol{x}'_{s-j})\delta_{m,m'}\end{aligned}$$

Thus, $K_{t,s}(\mathbf{x},\mathbf{x}')=(\sum_{j=0}^{t}\rho_j\ \boldsymbol{x}_{t-j}^{\mathsf{T}}\boldsymbol{x}'_{s-j})\mathbf{I}_{n_y}$ and we can write the full kernel as

$$K(\mathbf{x},\mathbf{x}')=\mathcal{T}(\mathbf{x})^{\mathsf{T}}D(\boldsymbol{\rho})\mathcal{T}(\mathbf{x}')\otimes\mathbf{I}_{n_y}, \tag{D.8}$$

where $\mathcal{T}(\mathbf{x})$ and $D(\boldsymbol{\rho})$ are defined in (5.18) and (5.19) respectively.

D.3 Proof of Theorem 7

Part (a) is a special case of a more general lemma, Lemma 1 which we present in Chapter 2 section 2.5. Let

$$\mathbf{q}_{t+1} = \frac{1}{\sqrt{n}}\mathbf{W}\mathbf{q}_t, \quad \mathbf{q}_0 = \mathbf{F}, \tag{D.9}$$

so that $\mathbf{q}_t$ represents the impulse response from $\boldsymbol{x}_t$ to $\mathbf{h}_t$. That is,

$$\mathbf{h}_t = \sum_{j=0}^{t} \mathbf{q}_{t-j}\boldsymbol{x}_j, \tag{D.10}$$

which is the convolution of $\mathbf{q}_t$ and $\mathbf{h}_t$. The system (D.9) is a special case of (2.43) with $L = 1$, no input $\mathbf{u}_t$ and

$$\mathbf{A} = \mathbf{W}, \quad G(\mathbf{q}) = \mathbf{q}.$$

Since there is only $L = 1$ transform, we have dropped the dependence on the index ℓ. Lemma 1 hows that $(\mathbf{q}_0, \ldots, \mathbf{q}_t)$ converges $PL(2)$ to a Gaussian vector $(Q_0, \ldots, Q_t)$ with zero mean. Note that each q_t is an $n \times n_x$ matrix so each Q_t is a $1 \times n_x$ vector. We claim that the Q_i's are independent. We prove this with induction. Suppose $(Q_0, \ldots, Q_t)$ are independent. We need to show $(Q_0, \ldots, Q_{t+1})$ are independent by using the evolution equations (2.47). Specifically, from (2.47b), $Z_i = Q_i$ for all i. Also, since each Q_i is zero mean, $\mu_i = 0$ and $\widetilde{Z}_i = Z_i = Q_i$. Since the $\widetilde{Z}_i$ are independent, the linear predictor coefficients in (2.47d) are zero: $F_{ti} = 0$. Therefore, $\widetilde{R}_t = R_t \sim \mathcal{N}(0, \nu_W P_t)$ is independent of $(R_0, \ldots, R_{t-1})$. From (2.47h), $Q_{t+1} = R_t$. So, we have that $(Q_0, \ldots, Q_{t-1})$ is an independent Gaussian vector. Finally, to compute the variance of the Q_{t+1}, observe

$$\begin{aligned}\mathrm{cov}(Q_{t+1}) &\overset{(a)}{=} \mathrm{cov}(R_t) \overset{(b)}{=} \nu_W P_t \\ &\overset{(c)}{=} \nu_W \mathrm{cov}(Z_t) \overset{(d)}{=} \nu_W \mathrm{cov}(Q_t),\end{aligned} \tag{D.11}$$

where (a) follows from (2.47h); (b) follows from (2.47f) and the fact that $F_{ti} = 0$ for all i; (c) follows from (2.47e); and (d) follows from the fact that $Z_t = Q_t$. Also, since $\mathbf{q}_0 = \mathbf{F}$, it follows that $Q_0 \sim \mathcal{N}(0, \nu_F \mathbf{I})$. We conclude that $\text{cov}(Q_t) = \nu_F \nu_W^t \mathbf{I}_{n_x}$. This proves part (a).

For part (b), we consider perturbations $\mathbf{\Delta}_W$, $\mathbf{\Delta}_F$, and $\mathbf{\Delta}_C$ of the parameters $\mathbf{W}$, $\mathbf{F}$, and $\mathbf{C}$. We have that,

$$\widetilde{\mathbf{h}}_t = \frac{1}{\sqrt{n}} \mathbf{W} \widetilde{\mathbf{h}}_{t-1} + \frac{1}{\sqrt{n}} \mathbf{\Delta}_W \mathbf{h}_t + \mathbf{\Delta}_F \boldsymbol{x}_t, \qquad \widetilde{\boldsymbol{y}}_t = \frac{1}{\sqrt{n}} \mathbf{C} \widetilde{\mathbf{h}}_t + \mathbf{\Delta}_C \mathbf{h}_t \tag{D.12}$$

Combining this equation with (5.1), we see that the mapping from $\boldsymbol{x}_t$ to $[\mathbf{h}_t \ \widetilde{\mathbf{h}}_t]$ is a linear time-invariant system. Let $\mathbf{q}_t \in \mathbb{R}^{n \times 2n_x}$ be its impulse response. The impulse response coefficients satisfy the recursive equations,

$$\mathbf{q}_{t+1} = \left[\frac{1}{\sqrt{n}} \mathbf{W} \mathbf{q}_{t,1}, \frac{1}{\sqrt{n}} (\mathbf{W} \mathbf{q}_{t,2} + \mathbf{\Delta}_W \mathbf{q}_{t,1}) \right], \quad \mathbf{q}_0 = [\mathbf{F}, \mathbf{\Delta}_F] \, .$$

We can analyze these coefficients in the LSL using Lemma 1. Specifically, let $L = 2$ and set

$$\mathbf{A}_1 = \mathbf{W}, \quad \mathbf{A}_2 = \mathbf{\Delta}_W .$$

Also, let

$$\mathbf{z}_{t1} = \overline{G}_1(\mathbf{q}_t) := \mathbf{q}_t, \tag{D.13a}$$

$$\mathbf{z}_{t2} = \overline{G}_2(\mathbf{q}_t) := [0 \ \ \mathbf{q}_{t1}]. \tag{D.13b}$$

Then, we have the updates,

$$\mathbf{q}_{t+1} = \frac{1}{\sqrt{n}} \mathbf{W} \mathbf{z}_{t1} + \frac{1}{\sqrt{n}} \mathbf{\Delta}_W \mathbf{z}_{t2}, \quad \mathbf{q}_0 = [\mathbf{F}, \mathbf{\Delta}_F] \, .$$

It follows from Lemma 1 that $(\mathbf{q}_0, \ldots, \mathbf{q}_{T-1})$ converges $PL(2)$ to zero mean Gaussian random variables $(Q_0, \ldots, Q_{T-1})$. Note that $Q_t = [Q_{t1}, Q_{t2}]$ where each Q_{t1} and Q_{t2} are random

vectors $\in \mathbb{R}^{1\times n_x}$. Similar to the proof of the previous theorem, we use induction to show that $(Q_0, \ldots, Q_t)$ are independent. Suppose that the claim is true for t. Then, Z_{i1} and Z_{i2} are functions of Q_i. So, for $\ell = 1, 2$, $Z_{t\ell}$ is independent of $Z_{i\ell}$ for $i < t$. Thus, the prediction coefficients $F_{ti\ell} = 0$ and, as before, $R_{t\ell} \sim \mathcal{N}(0, P_{t\ell})$ independent of $R_{i\ell}$, $i < t$. Thus, $Q_{t+1} = R_{t1} + R_{t2}$ is independent of $(Q_0, \ldots, Q_t)$.

We conclude by computing the $\mathrm{cov}(Q_t)$. We claim that, for all t, the variance of Q_t is of the form,

$$\mathrm{cov}(Q_t) = \begin{bmatrix} \tau_{t1}\mathbf{I}_{n_x} & 0 \\ 0 & \tau_{t2}\mathbf{I}_{n_x} \end{bmatrix} \tag{D.14}$$

for scalar τ_{t1}, τ_{t2}. Since $\mathbf{q}_0 = [\mathbf{F}, \boldsymbol{\Delta}_F]$, we have

$$\tau_{t1} = \nu_F, \quad \tau_{t2} = 1.$$

Now suppose that (D.14) is true for some t. From (2.47b),

$$Z_{t1} = Q_t, \quad Z_{t1} = (0, Q_{t1}),$$

from which we obtain that

$$\mathrm{cov}(Z_{t1}) = \begin{bmatrix} \tau_{t1}\mathbf{I}_{n_x} & 0 \\ 0 & \tau_{t2}\mathbf{I}_{n_x} \end{bmatrix}, \quad \mathrm{cov}(Z_{t2}) = \begin{bmatrix} 0 & 0 \\ 0 & \tau_{t1}\mathbf{I}_{n_x} \end{bmatrix}. \tag{D.15}$$

Therefore, we have

$$\begin{aligned} \mathrm{cov}(Q_{t+1}) &\overset{(a)}{=} \mathrm{cov}(R_{t,1}) + \mathrm{cov}(R_{t,2}) \\ &\overset{(b)}{=} \nu_W P_{t1} + P_{t2} \\ &\overset{(c)}{=} \nu_W \mathrm{cov}(Z_{t1}) + \mathrm{cov}(Z_{t2}) \overset{(d)}{=} \begin{bmatrix} \nu_W\tau_{t1}\mathbf{I}_{n_x} & 0 \\ 0 & (\tau_{t1} + \nu_W\tau_{t2})\mathbf{I}_{n_x} \end{bmatrix}, \end{aligned} \tag{D.16}$$

where (a) follows from (2.47h); (b) follows from (2.47f) and the fact that $F_{ti\ell} = 0$ for all i; (c) follows from (2.47e); and (d) follows from (D.15). It follows that

$$\tau_{t+1,1} = \nu_W \tau_{t1}, \quad \tau_{t+1,2} = \nu_W \tau_{t2} + \tau_{t1}.$$

These recursions have the solution,

$$\tau_{t1} = \nu_W^t \nu_F, \quad \tau_{t2} = t\nu_F \nu_W^{t-1} + \nu_W^t. \tag{D.17}$$

Since $[\mathbf{h}_t, \widetilde{\mathbf{h}}_t] = \sum_{j=0}^{t} \mathbf{q}_j \begin{bmatrix} \boldsymbol{x}_{t-j} \\ \boldsymbol{x}_{t-j} \end{bmatrix}$ and we know each $\mathbf{q}_t$ converges $PL(2)$ to random Q_t with covariances calculated in (D.14) and (D.17), we have

$$[\mathbf{h}_t, \widetilde{\mathbf{h}}_t] \overset{PL(2)}{=} [H_t, \widetilde{H}_t] = \sum_{j=0}^{t} Q_j \begin{bmatrix} \boldsymbol{x}_{t-j} \\ \boldsymbol{x}_{t-j} \end{bmatrix}, \tag{D.18}$$

where $H_t, \widetilde{H}_t$ are scalar random variables. For each t, s, we can now calculate the auto-correlation function for H as follows

$$\begin{aligned} \mathbb{E}[H_t H_s] &= \mathbb{E}[\sum_{j=0}^{t} \sum_{k=0}^{t} \boldsymbol{x}_{t-j}^{\mathsf{T}} Q_{j,1}^{\mathsf{T}} Q_{k,1} \boldsymbol{x}_{s-k}] \\ &= \sum_{j=0}^{t} \boldsymbol{x}_{t-j}^{\mathsf{T}} \mathbb{E}[Q_{j,1}^{\mathsf{T}} Q_{j,1}] \boldsymbol{x}_{s-j} \\ &= \sum_{j=0}^{t} \nu_W^j \nu_F \ \boldsymbol{x}_{t-j}^{\mathsf{T}} \boldsymbol{x}_{s-j}. \end{aligned} \tag{D.19}$$

Similarly for $\widetilde{H}$ we have

$$\mathbb{E}[\widetilde{H}_t \widetilde{H}_s] = \sum_{j=0}^{t} \left(j\nu_F \nu_W^{j-1} + \nu_W^j\right) \boldsymbol{x}_{t-j}^{\mathsf{T}} \boldsymbol{x}_{s-j}. \tag{D.20}$$

Thus, the impulse response of the system $\mathbf{L}_j = \mathbf{C}\mathbf{q}_{j,1}$ converge empirically to $\mathcal{N}(0, \Lambda)$ where,

$$\Lambda = \nu_C \lim_{n\to\infty} \frac{1}{n} q_{j,1}^{\mathsf{T}} q_{j,1} = \nu_C \mathbb{E}[Q_{j,1}^{\mathsf{T}} Q_{j,1}] = \nu_C \nu_F \nu_W^j \mathbf{I}_{n_x}. \tag{D.21}$$

This proves part (a). Note that $\mathbb{E}[\widetilde{H}_t \widetilde{H}_s']$ and $\mathbb{E}[H_t H_s']$ can be calculated similarly by substituting $\boldsymbol{x}_{s-j}$ with $\boldsymbol{x}'_{s-j}$ in (D.19) and (D.20).

Next, we calculate the NTK in this case

$$\begin{aligned} K_{t,s}(\mathbf{x}, \mathbf{x}') &= \sum_{\boldsymbol{\Delta}_\theta \in \mathcal{T}_\theta} \widetilde{\boldsymbol{y}}_t(\boldsymbol{\Delta}_\theta) \widetilde{\boldsymbol{y}}_s'(\boldsymbol{\Delta}_\theta)^{\mathsf{T}} \\ &\overset{(a)}{=} \mathbb{E}_{\boldsymbol{\Delta}_\theta \sim \mathcal{N}(0,1), \text{i.i.d.}} [\widetilde{\boldsymbol{y}}_t(\boldsymbol{\Delta}_\theta) \widetilde{\boldsymbol{y}}_s'(\boldsymbol{\Delta}_\theta)^{\mathsf{T}}], \end{aligned} \tag{D.22}$$

where (a) follows from Lemma 3. Combining with (D.12) we have

$$\begin{aligned} K_{t,s}(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{\mathbf{C}, \boldsymbol{\Delta}_C \sim \mathcal{N}(0,1), \text{i.i.d.}} \left[(\mathbf{C}\widetilde{\mathbf{h}}_t + \boldsymbol{\Delta}_C \mathbf{h}_t)(\mathbf{C}\widetilde{\mathbf{h}}_s' + \boldsymbol{\Delta}_C \mathbf{h}_s')^{\mathsf{T}} \right] \\ &= \left(\nu_C \mathbb{E}[\widetilde{H}_t \widetilde{H}_s'] + \mathbb{E}[H_t H_s'] \right) \mathbf{I}_{n_y}. \end{aligned} \tag{D.23}$$

Therefore,

$$K(\mathbf{x}, \mathbf{x}') = \mathcal{T}(\mathbf{x})^{\mathsf{T}} D(\boldsymbol{\rho}) \mathcal{T}(\mathbf{x}') \otimes \mathbf{I}_{n_y}, \tag{D.24}$$

where $\mathcal{T}(\mathbf{x})$ and $D(\boldsymbol{\rho})$ are given in (5.18) and (5.19) and

$$\rho_i = \nu_C (i \nu_F \nu_W^{i-1} + \nu_W^i) + \nu_W^i \nu_F. \tag{D.25}$$

This proves part (b).

D.4 Proof of Theorem 8

Bounding the Initial Impulse Response From Theorem 7, each coefficient of $\mathbf{L}^0_{\text{RNN},j}$ has mean zero and variance $\nu_C \nu_F \nu_W^j$. There are $n_x n_y$ such components. This proves (5.24).

Convolutional Equivalent Linear Model The key for the remainder of the proof is to use Theorems 6 and 7 to construct a scaled convolutional model that has the same NTK and intial conditions as the RNN. Then, we analyze the convolutional model to obtain the desired bound. To this end, let $\boldsymbol{\rho} = [\rho_0, \dots, \rho_{T-1}]$ be the scaling factors given in Theorem 7. For each initial condition $\boldsymbol{\theta}^0_{\text{RNN}} = (\mathbf{W}^0, \mathbf{F}^0, \mathbf{C}^0)$ of the RNN, suppose that we initialize the scaled convolutional model with

$$\boldsymbol{\theta}^0_{\text{conv},j} = \frac{1}{\sqrt{\rho_j} n^{(j+1)/2}} \mathbf{C}^0 (\mathbf{W}^0)^j \mathbf{F}^0.$$

The initial impulse response of the scaled convolutional model will then be

$$\mathbf{L}^0_{\text{conv},j} = \sqrt{\rho_j} \boldsymbol{\theta}_{\text{conv},j} = \frac{1}{n^{(j+1)/2}} \mathbf{C}^0 (\mathbf{W}^0)^j \mathbf{F}^0 = \mathbf{L}^0_{\text{RNN},j}. \tag{D.26}$$

Hence, the scaled convolutional model and the RNN have the same initial impulse response coefficients. We then train the scaled convolutional model on the training data using gradient descent with the same learning rate η used in the training of the RNN. Let $\mathbf{L}^\ell_{\text{conv},j}$ denote the impulse response of the scaled convolutional model after ℓ steps of gradient descent.

Gradient Descent Analysis of the Convolutional Model Next, we look at how the impulse response of the scaled convolutional model evolves over the gradient descent steps. It is convenient to do this analysis using some matrix notation. For each parameter, $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{T-1}]$, the convolutional filter parameters are $\mathbf{L}_j = \sqrt{\rho_j} \boldsymbol{\theta}_j$. Thus, we can write

$$\underline{\mathbf{L}} = \mathbf{D}^{1/2} \boldsymbol{\theta},$$

where $\mathbf{D}$ is a block diagonal operator with values ρ_j. Also, let $\underline{\widehat{\mathbf{y}}} = [\widehat{\mathbf{y}}_1, \dots, \widehat{\mathbf{y}}_N]$ be the set of predictions on the N training samples. Since the convolutional model is linear, we can write $\underline{\widehat{\mathbf{y}}} = \mathbf{A}\underline{\mathbf{L}}$ for some linear operator $\mathbf{A}$. The operator $\mathbf{A}$ would be a block Toeplitz with the

input data $\underline{\mathbf{x}} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$. Also, if we let $\underline{\mathbf{y}} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$ be the N training samples, the least squares cost is

$$\|\underline{\mathbf{y}} - \mathbf{A}\mathbf{D}^{1/2}\boldsymbol{\theta}\|_F^2.$$

Minimizing this loss function will result in GD steps,

$$\boldsymbol{\theta}^{\ell+1} = \boldsymbol{\theta}^{\ell} + \eta\mathbf{D}^{1/2}\mathbf{A}^{\mathsf{T}}(\underline{\mathbf{y}} - \mathbf{A}\mathbf{D}^{1/2}\boldsymbol{\theta}^{\ell}).$$

Now let $\mathbf{u}^{\ell} = \mathbf{D}^{-1/2}(\boldsymbol{\theta}^{\ell} - \boldsymbol{\theta}^{0})$ and $\mathbf{b} := \underline{\mathbf{y}} - \mathbf{A}\mathbf{D}^{1/2}\boldsymbol{\theta}^{0}$. Then,

$$\mathbf{u}^{\ell+1} = \mathbf{u}^{\ell} + \eta\mathbf{A}^{\mathsf{T}}(\mathbf{b} - \mathbf{A}\mathbf{D})\mathbf{u}^{\ell} = (\mathbf{I} - \eta\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{D})\mathbf{u}^{\ell} + \eta\mathbf{A}^{\mathsf{T}}\mathbf{b} \tag{D.27}$$

For $0 < \nu_W < 1$, we have that ρ_j satisfies the bound (5.22). Since $\mathbf{D}$ is a block diagonal matrix with entries ρ_j, $\|\mathbf{D}\| \leq \rho_{\max}$. Now select

$$B_1 := \frac{1}{\rho_{\max}\|\mathbf{A}\|^2}, \quad B_2 := \|\mathbf{A}^{\mathsf{T}}\mathbf{b}\|. \tag{D.28}$$

If we take $\eta < B_1$ then

$$\eta\mathbf{D}^{1/2}\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{D}^{1/2} \leq \eta\rho_{\max}\|\mathbf{A}\|^2 \leq \mathbf{I} \Rightarrow \|\mathbf{I} - \eta\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{D}\| \leq 1.$$

Hence, (D.27) shows that

$$\|\mathbf{u}^{\ell+1}\|_F \leq \|\mathbf{u}^{\ell}\|_F + \eta B_2 \Rightarrow \|\mathbf{u}^{\ell}\|_F \leq \eta\ell B_2, \tag{D.29}$$

where we have used the fact that $\mathbf{u}^0 = \mathbf{0}$. Now, since $\mathbf{u}^{\ell} = \mathbf{D}^{-1/2}(\boldsymbol{\theta}^{\ell} - \boldsymbol{\theta}^{0})$, the j-th component of $\boldsymbol{\theta}^{\ell}$ is

$$\boldsymbol{\theta}_j^{\ell} = \boldsymbol{\theta}_j^{0} + \sqrt{\rho_j}\mathbf{u}_j^{\ell}.$$

Hence, $\mathbf{L}_{\text{conv},j}^{\ell} = \sqrt{\rho_j}\boldsymbol{\theta}_j^{\ell} = \mathbf{L}_{\text{conv},j}^{0} + \rho_j\mathbf{u}_j^{\ell}$.

Applying (D.29) we obtain the bound on the convolutional model

$$\|\mathbf{L}_{\text{conv},j}^{\ell} - \mathbf{L}_{\text{conv},j}^{0}\|_F \leq \rho_j \eta \ell B_2. \tag{D.30}$$

Bounding the RNN Impulse Response From Theorems 6 and 7, the scaled convolutional model and linear RNN have the same NTK. Due to (D.26), they have the same input-output mapping at the initial conditions. Since the scaled convolutional model is linear in its parameters it follows that it is linear NTK model for the RNN. Therefore, using the NTK results such as Proposition 2, we have that for all input sequences $\mathbf{x}$ and GD time steps ℓ,

$$\lim_{n\to\infty} \left\| f_{\text{RNN}}(\mathbf{x}, \boldsymbol{\theta}_{\text{RNN}}^{\ell}) - f_{\text{conv}}(\mathbf{x}, \boldsymbol{\theta}_{\text{conv}}^{\ell}) \right\| = 0, \tag{D.31}$$

where the convergence is in probability. Thus, if we fix an input $\mathbf{x}$ and iteration ℓ and define

$$\mathbf{y}_{\text{RNN}} = f_{\text{RNN}}(\mathbf{x}, \boldsymbol{\theta}_{\text{RNN}}^{\ell}), \quad \mathbf{y}_{\text{conv}} = f_{\text{conv}}(\mathbf{x}, \boldsymbol{\theta}_{\text{conv}}^{\ell}),$$

the limit (D.31) can be re-written as

$$\lim_{n\to\infty} \|\boldsymbol{y}_{\text{RNN},j} - \boldsymbol{y}_{\text{conv},j}\| = 0, \tag{D.32}$$

for all j. Again, the convergence is in probability. Now consider the case where the input sequence $\mathbf{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1})$ is a sequence with $\boldsymbol{x}_j = \mathbf{0}$ for all $j > 0$ That is, it is only non-zero at the initial time step. Then for all time steps $\boldsymbol{y}_{\text{RNN},j} = \mathbf{L}_{\text{RNN},j}^{\ell} \boldsymbol{x}_0$ and $\boldsymbol{y}_{\text{conv},j} = \mathbf{L}_{\text{conv},j}^{\ell} \boldsymbol{x}_0$. Since this is true for all $\boldsymbol{x}_0$, (D.32) shows that for all time steps $j = 0, \ldots, T-1$,

$$\lim_{n\to\infty} \left\| \mathbf{L}_{\text{RNN},j}(\mathbf{x}, \boldsymbol{\theta}_{\text{RNN}}^{\ell}) - \mathbf{L}_{\text{conv},j}(\mathbf{x}, \boldsymbol{\theta}_{\text{conv}}^{\ell}) \right\|_F = 0 \tag{D.33}$$

where the convergence is in probability. Combining (D.33) with (D.30) proves (5.25).

D.5 Proof of Theorem 9

From the Proposition 2 and results from [3] we know that for step ℓ of the GD

$$\lim_{n\to\infty} \sup_{\ell\geq 0} \left\| F_n(\mathbf{x}, \boldsymbol{\theta}^\ell) - F_n^{\text{lin}}(\mathbf{x}, \boldsymbol{\alpha}^\ell) \right\| = 0 \tag{D.34}$$

where the subscript denotes the state dimension n and $\boldsymbol{\alpha}$ denotes the parameters of the linear NTK model.

$$\widehat{\mathbf{y}} = F^{\text{lin}}(\mathbf{x}, \boldsymbol{\alpha}) := F(\mathbf{x}, \boldsymbol{\theta}^0) + \sum_{i=1}^{N} K(\mathbf{x}^i, \mathbf{x})\boldsymbol{\alpha}_i, \tag{D.35}$$

where $K(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{Tn_y \times Tn_y}$ is the so-called NTK.

Consider the following perturbed system for $F(\mathbf{x}, \boldsymbol{\theta})$:

$$\boldsymbol{\delta}\mathbf{h}_t = \mathbf{W}_t\ \boldsymbol{\delta}\mathbf{h}_{t-1} + \mathbf{B}_t\ \boldsymbol{\delta\theta} + \mathbf{F}_t\ \boldsymbol{\delta x}_t, \quad \boldsymbol{\delta y}_t = \mathbf{C}_t\ \boldsymbol{\delta}\mathbf{h}_t. \tag{D.36}$$

Also, for any training data $(\mathbf{x}^i, \mathbf{y}^i)$, we have the following system

$$\boldsymbol{\delta}\mathbf{h}_t^i = \mathbf{W}_t^i\ \boldsymbol{\delta}\mathbf{h}_{t-1}^i + \mathbf{B}_t^i\ \boldsymbol{\delta\theta} + \mathbf{F}_t^i\ \boldsymbol{\delta x}_t^i, \quad \boldsymbol{\delta y}_t^i = \mathbf{C}_t^i\ \boldsymbol{\delta}\mathbf{h}_t^i. \tag{D.37}$$

Considering that the assumptions in (5.30) hold true, we have

$$\boldsymbol{y}_t(\boldsymbol{\delta}\mathbf{x}) = \sum_{i=1}^{N}\sum_{t'=0}^{T-1} [K(\mathbf{x}^i, \mathbf{x})]_{t,t'} (\boldsymbol{\alpha}_i^\ell)_{t'} \tag{D.38a}$$

$$= \sum_{i=1}^{N}\sum_{t'=0}^{T-1} \mathbb{E}_{\boldsymbol{\delta\theta}\sim\mathcal{N}(0,\mathbf{I})} \left[\boldsymbol{\delta y}_t(\boldsymbol{\delta\theta})\ \boldsymbol{\delta y}_{t'}(\boldsymbol{\delta\theta}, \boldsymbol{\delta x}_{t'})^\mathsf{T}\right]_{t,t'} (\boldsymbol{\alpha}_i^\ell)_{t'} \tag{D.38b}$$

$$= \sum_{i=1}^{N}\sum_{t'=0}^{T-1} \mathbb{E}_{\boldsymbol{\delta\theta}\sim\mathcal{N}(0,\mathbf{I})} \left[\mathbf{C}_t^i(\mathbf{W}_t^i\boldsymbol{\delta}\mathbf{h}_{t-1}^i + \mathbf{B}_t^i\boldsymbol{\delta\theta})\ (\mathbf{W}_{t'}\boldsymbol{\delta}\mathbf{h}_{t'-1} + \mathbf{B}_{t'}\boldsymbol{\delta\theta} + \mathbf{F}_{t'}\boldsymbol{\delta x}_{t'})^\mathsf{T}\mathbf{C}_{t'}^\mathsf{T}\right] (\boldsymbol{\alpha}_i^\ell)_{t'}. \tag{D.38c}$$

Therefore,

$$\frac{\partial \boldsymbol{y}_t}{\partial \boldsymbol{x}_s} = \sum_{i=1}^{N} \mathbb{E}_{\boldsymbol{\delta\theta}\sim\mathcal{N}(0,\mathbf{I})} \left[\mathbf{C}_t^i(\mathbf{W}_t^i \boldsymbol{\delta}\mathbf{h}_{t-1}^i + \mathbf{B}_t^i \boldsymbol{\delta\theta}) \ \left(\sum_{t'=s}^{T-1} \frac{\partial \mathbf{h}_{t'}}{\partial \boldsymbol{x}_s}^{\mathsf{T}} \mathbf{C}_{t'}^{\mathsf{T}}(\boldsymbol{\alpha}_i^\ell)_{t'} \right) \right]$$
$$= \sum_{i=1}^{N} \mathbf{C}_t^i \mathbf{W}_t^i \ \mathbb{E}_{\boldsymbol{\delta\theta}\sim\mathcal{N}(0,\mathbf{I})}[\boldsymbol{\delta}\mathbf{h}_{t-1}^i] \ \left(\mathbf{F}_s^{\mathsf{T}} \mathbf{C}_s^{\mathsf{T}}(\boldsymbol{\alpha}_i^\ell)_s + \sum_{t'=s+1}^{T-1} \mathbf{F}_s^{\mathsf{T}} (\prod_{j=s+1}^{t'} \mathbf{W}_j^{\mathsf{T}}) \mathbf{C}_{t'}^{\mathsf{T}}(\boldsymbol{\alpha}_i^\ell)_{t'} \right). \quad \text{(D.39)}$$

We know that there exist constants $C_0 > 0$ such that for all $i \in [N]$,

$$\left\| \mathbb{E}_{\boldsymbol{\delta\theta}\sim\mathcal{N}(0,\mathbf{I})}(\boldsymbol{\delta}\mathbf{h}_t^i) \right\|_2 \leq C_0 \rho^t. \quad \text{(D.40)}$$

If we show that for all $i \in [N]$, there exist constants $C_1 > 0$ such that

$$\left\| (\boldsymbol{\alpha}_i^\ell)_t \right\|_2 \leq C_1 \eta \ell, \quad \text{(D.41)}$$

we get,

$$\left\| \frac{\partial \boldsymbol{y}_t}{\partial \boldsymbol{x}_s} \right\| \leq \left(C_2 \rho^t + C_3 \rho^{|t-s|} \right) \eta \ell$$
$$\leq C_4 \rho^{|t-s|} \eta \ell, \quad \text{(D.42)}$$

which proves (5.31). To show (D.41), assume we are given training data $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^N$, let $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^N]^{\mathsf{T}}$ and $\mathbf{Y} = [\mathbf{y}^1, \ldots, \mathbf{y}^N]^{\mathsf{T}} \in \mathbb{R}^{NTn_y}$. Also let $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}^1, \ldots, \widehat{\mathbf{y}}^N]^{\mathsf{T}} \in \mathbb{R}^{NTn_y}$ be the predictions as defined in (D.35). We want to minimize the square loss

$$\tfrac{1}{2} \left\| \mathbf{Y} - \widehat{\mathbf{Y}} \right\|_F^2, \quad \text{(D.43)}$$

and we have

$$\widehat{\mathbf{Y}} := F^{\text{lin}}(\mathbf{X}, \boldsymbol{\alpha}) = F(\mathbf{X}, \boldsymbol{\theta}^0) + \mathbf{K}\boldsymbol{\alpha}. \quad \text{(D.44)}$$

Here, $\mathbf{K}$ is the empirical NTK such that

$$\mathbf{K} = \widetilde{K}(\mathbf{X}, \mathbf{X}) \otimes \mathbf{I}_{n_y} \ \in \mathbb{R}^{NTn_y \times NTn_y} \tag{D.45}$$

$$[\widetilde{K}(\mathbf{X}, \mathbf{X})]_{t,s} = K_{t,s}(\mathbf{X}, \mathbf{X}) \ \in \mathbb{R}^{N \times N} \qquad t, s = 0, \ldots, T-1. \tag{D.46}$$

At step ℓ of gradient descent we have:

$$\boldsymbol{\alpha}^{\ell+1} = \boldsymbol{\alpha}^{\ell} + \eta \mathbf{K}^{\mathsf{T}}(\mathbf{Y} - \mathbf{K}\boldsymbol{\alpha}^{\ell} - f(\mathbf{X}, \boldsymbol{\theta}^0)). \tag{D.47}$$

Let $\mathbf{u}^{\ell} = \boldsymbol{\alpha}^{\ell} - \boldsymbol{\alpha}^0$ and $\mathbf{b} = \mathbf{Y} - \mathbf{K}\boldsymbol{\alpha}^0 - f(\mathbf{X}, \boldsymbol{\theta}^0)$. Therefore,

$$\begin{aligned} \mathbf{u}^{\ell+1} &= \mathbf{u}^{\ell} + \eta \mathbf{K}^{\mathsf{T}}(\mathbf{b} - \mathbf{K}\mathbf{u}^{\ell}) \\ &= (\mathbf{I} - \eta \mathbf{K}^{\mathsf{T}}\mathbf{K})\mathbf{u}^{\ell} + \eta \mathbf{K}^{\mathsf{T}}\mathbf{b}. \end{aligned} \tag{D.48}$$

Let $C_5 = \left\|\mathbf{K}^{\mathsf{T}}\mathbf{b}\right\|_2$. Then,

$$\begin{aligned} \|\mathbf{u}^{\ell+1}\|_2 &\leq \|\mathbf{u}^{\ell}\|_2 + \eta C_5 \\ &\leq C_5' \eta \ell. \end{aligned} \tag{D.49}$$

Therefore, there exists a C_1 such that $\left\|(\boldsymbol{\alpha}_i^{\ell})_t\right\|_2 \leq C_1 \eta \ell$.

Bibliography

[1] Madhu Advani and Surya Ganguli. An equivalence between high dimensional bayes optimal inference and m-estimation. In *Advances in Neural Information Processing Systems*, pages 3378–3386, 2016.

[2] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[3] Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. *arXiv preprint arXiv:2006.10246*, 2020.

[4] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems*, pages 6155–6166, 2019.

[5] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

[6] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

[7] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

[8] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv*, pages arXiv–1409, 2014.

[10] Zhi Dong Bai and Yong Quan Yin. Limiting behavior of the norm of products of random matrices and two problems of geman-hwang. *Probability theory and related fields*, 73(4):555–569, 1986.

[11] Jean Barbier, Florent Krzakala, Nicolas Macris, Léo Miolane, and Lenka Zdeborová. Optimal errors and phase transitions in high-dimensional generalized linear models. *Proc. National Academy of Sciences*, 116(12):5451–5460, 2019.

[12] Jean Barbier, Florent Krzakala, Nicolas Macris, Léo Miolane, and Lenka Zdeborová. Optimal errors and phase transitions in high-dimensional generalized linear models. *Proc. National Academy of Sciences*, 116(12):5451–5460, March 2019.

[13] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *arXiv preprint arXiv:1906.11300*, 2019.

[14] M. Bayati, M. Lelarge, and A. Montanari. Universality in polytope phase transitions and iterative algorithms. In *Proc. IEEE ISIT*, pages 1643 –1647, July 2012.

[15] M. Bayati and A. Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Trans. Inform. Theory*, 57(2):764–785, February 2011.

[16] Mohsen Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57(2):764–785, 2011.

[17] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. National Academy of Sciences*, 116(32):15849–15854, 2019.

[18] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.

[19] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018.

[20] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188. IEEE, 1993.

[21] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[22] Ari S Benjamin, Hugo L Fernandes, Tucker Tomlinson, Pavan Ramkumar, Chris VerSteeg, Raeed H Chowdhury, Lee E Miller, and Konrad P Kording. Modern machine learning as a benchmark for fitting neural responses. *Frontiers in computational neuroscience*, 12:56, 2018.

[23] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*, 2019.

[24] Siegfried Bös and Manfred Opper. Dynamics of training. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 141–147. MIT Press, 1997.

[25] Burak Cakmak, Ole Winther, and Bernard H Fleury. S-AMP: Approximate message passing for general matrix ensembles. In *Proc. IEEE ITW*, 2014.

[26] Emmanuel Candes and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse problems*, 23(3):969, 2007.

[27] Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9):589–592, 2008.

[28] Minmin Chen, Jeffrey Pennington, and Samuel S Schoenholz. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. *arXiv preprint arXiv:1806.05394*, 2018.

[29] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2933–2943, 2019.

[30] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[31] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[32] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[33] Amit Daniely. Sgd learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2017.

[34] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pages 2253–2261, 2016.

[35] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

[36] Zeyu Deng, Abla Kammoun, and Christos Thrampoulidis. A model of double descent for high-dimensional binary linear classification. *arXiv preprint arXiv:1911.05822*, 2019.

[37] D. L. Donoho, A. Maleki, and A. Montanari. Message passing algorithms for compressed sensing. In *Proc. Inform. Theory Workshop*, pages 1–5, 2010.

[38] D. L. Donoho, A. Maleki, and A. Montanari. Message passing algorithms for compressed sensing II: Analysis and validation. In *Proc. Inform. Theory Workshop*, pages 1–5, January 2010.

[39] David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proc. National Academy of Sciences*, 106(45):18914–18919, 2009.

[40] Xialiang Dou and Tengyuan Liang. Training neural networks as learning data-adaptive kernels: Provable representation and approximation benefits. *Journal of the American Statistical Association*, pages 1–14, 2020.

[41] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

[42] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[43] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[44] Weinan E, Chao Ma, and Lei Wu. A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics. *Science China Mathematics*, Jan 2020.

[45] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[46] Melikasadat Emami, Mojtaba Sahraee Ardakan, Sundeep Rangan, and Alyson K Fletcher. Input-output equivalence of unitary and contractive rnns. In *Advances in Neural Information Processing Systems*, pages 15342–15352, 2019.

[47] Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson Fletcher. Generalization error of generalized linear models in high dimensions. In *International Conference on Machine Learning*, pages 2892–2901. PMLR, 2020.

[48] Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K Fletcher. Implicit bias of linear rnns. In *International Conference on Machine Learning*, pages 2982–2992. PMLR, 2021.

[49] Alyson Fletcher, Mojtaba Sahraee-Ardakan, Sundeep Rangan, and Philip Schniter. Expectation consistent approximate inference: Generalizations and convergence. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pages 190–194. IEEE, 2016.

[50] Alyson K Fletcher, Sundeep Rangan, and P. Schniter. Inference in deep networks in high dimensions. *arXiv:1706.06549*, 2017.

[51] Alyson K Fletcher, Sundeep Rangan, and P. Schniter. Inference in deep networks in high dimensions. *Proc. IEEE Int. Symp. Information Theory*, 2018.

[52] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993.

[53] Marylou Gabrié, Andre Manoel, Clément Luneau, Jean Barbier, Nicolas Macris, Florent Krzakala, and Lenka Zdeborová. Entropy and mutual information in models of deep neural networks. In *Proc. NIPS*, 2018.

[54] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

[55] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.

[56] Cédric Gerbelot, Alia Abbara, and Florent Krzakala. Asymptotic errors for convex penalized linear regression beyond gaussian matrices. *arXiv preprint arXiv:2002.04372*, 2020.

[57] Clark R Givens, Rae Michael Shortt, et al. A class of wasserstein metrics for probability distributions. *The Michigan Mathematical Journal*, 31(2):231–240, 1984.

[58] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

[59] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

[60] Hengtao He, Chao-Kai Wen, and Shi Jin. Generalized expectation consistent signal recovery for nonlinear measurements. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2333–2337. IEEE, 2017.

[61] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29, 2012.

[62] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[63] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[64] Li Jing, Caglar Gulcehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljacic, and Yoshua Bengio. Gated orthogonal recurrent units: On learning to forget. *Neural Computation*, 31(4):765–783, 2019.

[65] Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1733–1741. JMLR. org, 2017.

[66] Thomas Kailath. *Linear systems*, volume 156. Prentice-Hall Englewood Cliffs, NJ, 1980.

[67] Tohru Katayama. *Subspace methods for system identification*. Springer Science & Business Media, 2006.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Florent Krzakala, Marc Mézard, François Sausset, YF Sun, and Lenka Zdeborová. Statistical-physics-based reconstruction in compressed sensing. *Physical Review X*, 2(2):021005, 2012.

[70] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[71] Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[72] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8570–8581, 2019.

[73] Ljung Lennart. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, pages 1–14, 1999.

[74] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[75] L. Ljung and T. Glad. *Modeling of Dynamic Systems*. Prentice-Hall information and system sciences series. PTR Prentice Hall, 1994.

[76] Lennart Ljung. System identification. *Wiley encyclopedia of electrical and electronics engineering*, pages 1–19, 1999.

[77] Chao Ma, Lei Wu, et al. A comparative analysis of the optimization and generalization property of two-layer neural network and random feature models under gradient descent dynamics. *arXiv preprint arXiv:1904.04326*, 2019.

[78] Junjie Ma and Li Ping. Orthogonal amp. *IEEE Access*, 5:2020–2033, 2017.

[79] Andre Manoel, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Multi-layer generalized linear estimation. pages 2098–2102, 2017.

[80] Andre Manoel, Florent Krzakala, Gaël Varoquaux, Bertrand Thirion, and Lenka Zdeborová. Approximate message-passing for convex optimization with non-separable penalties. *arXiv preprint arXiv:1809.06304*, 2018.

[81] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[82] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

[83] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[84] Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2401–2409. JMLR. org, 2017.

[85] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proc. Conf. Uncertainty in Artificial Intelligence*, pages 352–359, 2002.

[86] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proc. UAI*, pages 362–369, 2001.

[87] Andrea Montanari. Graphical model concepts in compressed sensing. In Yonina C. Eldar and Gitta Kutyniok, editors, *Compressed Sensing: Theory and Applications*, pages 394–438. Cambridge Univ. Press, June 2012.

[88] Andrea Montanari, Feng Ruan, Youngtak Sohn, and Jun Yan. The generalization error of max-margin linear classifiers: High-dimensional asymptotics in the overparametrized regime. *arXiv preprint arXiv:1911.01544*, 2019.

[89] Vidya Muthukumar, Kailas Vodrahalli, and Anant Sahai. Harmless interpolation of noisy data in regression. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2299–2303. IEEE, 2019.

[90] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

[91] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[92] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer New York, 1996.

[93] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[94] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[95] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[96] Manfred Opper and Ole Winther. Expectation consistent approximate inference. *Journal of Machine Learning Research*, 6(Dec):2177–2204, 2005.

[97] P. Pandit, M. Sahraee, S. Rangan, and A. K. Fletcher. Asymptotics of MAP inference in deep networks. In *Proc. IEEE Int. Symp. Information Theory*, pages 842–846, 2019.

[98] Parthe Pandit, Mojtaba Sahraee-Ardakan, Sundeep Rangan, Philip Schniter, and Alyson K Fletcher. Inference with deep generative priors in high dimensions. *arXiv preprint arXiv:1911.03409*, 2019.

[99] Parthe Pandit, Mojtaba Sahraee-Ardakan, Sundeep Rangan, Philip Schniter, and Alyson K Fletcher. Inference with deep generative priors in high dimensions. *IEEE Journal on Selected Areas in Information Theory*, 2020.

[100] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[101] Jeffrey Pennington, Samuel S Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. *arXiv preprint arXiv:1802.09979*, 2018.

[102] Rik Pintelon and Johan Schoukens. *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

[103] Sundeep Rangan. Estimation with random linear mixing, belief propagation and compressed sensing. In *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2010.

[104] Sundeep Rangan, Vivek Goyal, and Alyson K Fletcher. Asymptotic analysis of map estimation via the replica method and compressed sensing. *Advances in Neural Information Processing Systems*, 22, 2009.

[105] Sundeep Rangan, Philip Schniter, and Alyson K Fletcher. Vector approximate message passing. *IEEE Trans. Information Theory*, 65(10):6664–6684, 2019.

[106] Sundeep Rangan, Philip Schniter, and Alyson K Fletcher. Vector approximate message passing. *IEEE Transactions on Information Theory*, 65(10):6664–6684, 2019.

[107] Galen Reeves. Additivity of information in multilayer networks via additive gaussian noise transforms. In *Proc. 55th Annual Allerton Conf. Communication, Control, and Computing (Allerton)*, pages 1064–1070. IEEE, 2017.

[108] Grant M Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

[109] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[110] Fariborz Salehi, Ehsan Abbasi, and Babak Hassibi. The impact of regularization on high-dimensional logistic regression. *arXiv preprint arXiv:1906.03761*, 2019.

[111] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[112] M. Seeger. Bayesian inference and optimal design for the sparse linear model. *J. Machine Learning Research*, 9:759–813, September 2008.

[113] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[114] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

[115] Torsten Söderström and Petre Stoica. *System identification*. Prentice-Hall International, 1989.

[116] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[117] Gilbert W Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17(3):403–409, 1980.

[118] Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[119] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[120] Keigo Takeuchi. Rigorous dynamics of expectation-propagation-based signal recovery from unitarily invariant measurements. pages 501–505, 2017.

[121] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[122] Antonia M Tulino, Sergio Verdú, et al. Random matrix theory and wireless communications. *Foundations and Trends® in Communications and Information Theory*, 1(1):1–182, 2004.

[123] Mats Viberg. Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, 31(12):1835–1851, 1995.

[124] Mathukumalli Vidyasagar. *Nonlinear systems analysis*, volume 42. Siam, 2002.

[125] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[126] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3570–3578. JMLR. org, 2017.

[127] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. In *Advances in Neural Information Processing Systems*, pages 9709–9721, 2019.

[128] Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 4880–4888, 2016.

[129] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[130] Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, pages 9951–9960, 2019.

[131] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[132] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888*, 2018.