UNDERSTANDING DEEP NEURAL NETWORKS AND OTHER NONPARAMETRIC METHODS IN MACHINE LEARNING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Yixi Xu

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Xiao Wang, Chair

Department of Statistics

Dr. Chuanhai Liu

Department of Statistics

Dr. Jun Xie

Department of Statistics

Dr. Lingsong Zhang

Department of Statistics

Approved by:

Dr. Jun Xie

Graduate Chair of the Department of Statistics

To my family.

ACKNOWLEDGMENTS

First of all, I would like to express my most sincere gratitude to my advisor Dr. Xiao Wang, who constantly inspires me to explore the foundation of machine learning and always provides insightful suggestions on my research. I would also like to thank Dr. Chuanhai Liu, Dr. Jun Xie, and Dr. Lingsong Zhang for their guidance as my advisory committee member. In addition, special thanks goes to Dr. Hao Zhang and Dr. Jun Xie for their priceless support for my study, research, and teaching in the past years.

I would like to express my gratitude to my collaborators: Dr. Jean Honorio, Dr. Chenyu Huang, Dr. Mary E. Johnson, Dr. Qiang Liu, Dr. Zhouwang Yang, Dr. Aijun Zhang, Dr. Xin Zhang, Ming Wen, and Yunling Zheng, who make me a better statistician. Besides, I would like to thank Ellen Gundlach, who I had worked with for four years as a teaching assistant.

I would like to thank my friends from the Department of Statistics at Purdue University, who always support and encourage me in their own ways.

Last but not the least, I am grateful to my parents for their unconditional love and support throughout my life.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CIFAR10 dataset	Canadian Institute For Advanced Research dataset
CPWL	continuous piecewise linear
DNN	deep neural network
MNIST database	modified National Institute of Standards and Technology database
ReLU	rectified linear units
SGD	stochastic gradient descent
WN-DNN	weight normalized deep neural network

ABSTRACT

Xu, Yixi Ph.D., Purdue University, August 2019. Understanding Deep Neural Networks and Other Nonparametric Methods in Machine Learning. Major Professor: Xiao Wang Professor.

It is a central problem in both statistics and computer science to understand the theoretical foundation of machine learning, especially deep learning. During the past decade, deep learning has achieved remarkable successes in solving many complex artificial intelligence tasks. The aim of this dissertation is to understand deep neural networks (DNNs) and other nonparametric methods in machine learning. In particular, three machine learning models have been studied: weight normalized DNNs, sparse DNNs, and the compositional nonparametric model.

The first chapter presents a general framework for norm-based capacity control for $L_{p,q}$ weight normalized DNNs. We establish the upper bound on the Rademacher complexities of this family. Especially, with an $L_{1,\infty}$ normalization, we discuss properties of a width-independent capacity control, which only depends on the depth by a square root term. Furthermore, if the activation functions are anti-symmetric, the bound on the Rademacher complexity is independent of both the width and the depth up to a log factor. In addition, we study the weight normalized deep neural networks with rectified linear units (ReLU) in terms of functional characterization and approximation properties. In particular, for an $L_{1,\infty}$ weight normalized network with ReLU, the approximation error can be controlled by the L_1 norm of the output layer.

In the second chapter, we study $L_{1,\infty}$-weight normalization for deep neural networks with bias neurons to achieve the sparse architecture. We theoretically establish the generalization error bounds for both regression and classification under the $L_{1,\infty}$-weight normalization. It is shown that the upper bounds are independent of the

network width and $\sqrt{k}$-dependence on the network depth k. These results provide theoretical justifications on the usage of such weight normalization to reduce the generalization error. We also develop an easily implemented gradient projection descent algorithm to practically obtain a sparse neural network. We perform various experiments to validate our theory and demonstrate the effectiveness of the resulting approach.

In the third chapter, we propose a compositional nonparametric method in which a model is expressed as a labeled binary tree of $2k+1$ nodes, where each node is either a summation, a multiplication, or the application of one of the q basis functions to one of the m_1 covariates. We show that in order to recover a labeled binary tree from a given dataset, the sufficient number of samples is $O(k \log(m_1 q) + \log(k!))$, and the necessary number of samples is $\Omega(k \log(m_1 q) - \log(k!))$. We further propose a greedy algorithm for regression in order to validate our theoretical findings through synthetic experiments.

1. INTRODUCTION

1.1 The General Prediction Problem

In this section, we define the general prediction problem. Assume that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are n independent random variables on $\mathcal{X} \subseteq \mathbb{R}^{m_1}$, $\boldsymbol{y}_1^*, \ldots, \boldsymbol{y}_n^*$ are on $\mathcal{Y}^* \subseteq \mathbb{R}^{m_2}$, $y_1, \ldots, y_n$ are on $\mathcal{Y} \subseteq \mathbb{R}$, and the noise $\varepsilon_1, \ldots, \varepsilon_n$ are independent while satisfying that $\mathbb{E}(\varepsilon_i) = 0$. The general prediction problem is defined as

$$\begin{aligned} \boldsymbol{y}_i^* &= f(\boldsymbol{x}_i) + \varepsilon_i \\ y_i &= t(\boldsymbol{y}_i^*), \end{aligned} \tag{1.1}$$

where $t : \mathcal{Y}^* \to \mathcal{Y}$ is a fixed function related to the prediction problem, and $f : \mathcal{X} \to \mathcal{Y}^*$ is an unknown function. We provide two examples in order to illustrate how to adapt Equation (1.1) to different settings. For regression, we have $m_2 = 1$, $\mathcal{Y}^* = \mathcal{Y}$, and t is the identity mapping. While for classification, we could define m_2 as the number of classes, $\mathcal{Y} = \{1, 2, \cdots, m_2\}$, and $t = \text{argmax}$.

1.2 Selected Frameworks for Model Estimation

1.2.1 The Linear Model

Consider the regression case, where t is an identity function, and $m_2 = 1$ in Equation (1.1). The linear model in addition assumes that

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\beta}^* + \beta_0^*,$$

where $\boldsymbol{\beta}^* \in \mathbb{R}^{m_1}$ and $\beta_0^* \in \mathbb{R}$. Furthermore, assume that the noise $\{\varepsilon_i : i = 1, \cdots, n\}$ are independent and identically distributed normal variables. The maximum likelihood estimate of $\boldsymbol{\beta}^*$ and β_0^* is obtained by minimizing the least squares error

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta} - \beta_0)^2.$$

However, the ordinary least squares estimate suffers, when the explanatory variables are highly correlated with each other, or the input dimension is high. To tackle this problem, we could use the ridge regression by minimizing a penalized loss

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta} - \beta_0)^2 + \lambda \left\| \boldsymbol{\beta} \right\|_2^2,$$

where λ is the tuning parameter. If $\lambda = 0$, it is equivalent to ordinary least squares. If $\lambda = \infty$, it forces $\boldsymbol{\beta} = \mathbf{0}$.

Lasso regression is another popular method to handle overfitting by minimizing a penalized loss

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta} - \beta_0)^2 + \lambda \left\| \boldsymbol{\beta} \right\|_1,$$

where λ is the tuning parameter. If $\lambda = 0$, it is equivalent to ordinary least squares. If $\lambda = \infty$, it forces $\boldsymbol{\beta} = \mathbf{0}$. Furthermore, lasso regression could be used for variable selection, as it exhibits sparsity.

As a summary, linear regression is straightforward to interpret and easy to implement. However, real data might fail to support the assumption of linear models in practice. In this case, linear model will severely underfit the data.

1.2.2 Spline-based Models

Consider the regression case, where t is an identity function, and $m_2 = 1$ in Equation (1.1). For simplicity, assume that the input dimension $m_1 = 1$, though spline-based models could be extended to high dimensions.

Regression Splines. Assume that

$$f(x) = \alpha_0 + \sum_{i=1}^{I} \alpha_i \phi_i(x),$$

where ϕ_i is a spline basis, for $i = 1, \cdots, I$. Besides, assume that the noise $\{\varepsilon_i : i = 1, \cdots, n\}$ are independent and identically distributed normal variables. Popular choices of splines include cubic splines and B-splines. The model is estimated by minimizing the least squares criterion:

$$\min_{\alpha_0, \cdots, \alpha_I} \sum_{i=1}^{n} (y_i - \alpha_0 - \sum_{i=1}^{I} \alpha_i \phi_i(x))^2.$$

Note that the knots of the splines $\{\phi_1, \cdots, \phi_I\}$ are usually pre-determined in practice. One important extension of regression splines is the adaptive free-knots splines [1–3], where the knots are estimated simultaneously along with the coefficients $\alpha_0, \cdots, \alpha_I$.

Smoothing Splines. Assume that the noise $\{\varepsilon_i : i = 1, \cdots, n\}$ are independent and identically distributed normal variables.

The model is estimated by minimizing the penalized likelihood score:

$$\min_{f} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_{\mathcal{X}} \left(f''(x) \right)^2 dx,$$

where the penalty term is introduced for lack of smoothness, and λ is the smoothing parameter. A smoothing spline estimator [4] has knots at each data point. As λ increases, the model becomes smoother. If $\lambda = 0$, the fitted model interpolates the data. If $\lambda = \infty$, the model is equivalent to linear regression.

When the input dimension is small, spline-based models have not only enjoyed sufficient theoretical guidance but also demonstrated their effectiveness via adequate simulation studies and real-world experiments. However, it turns out to be extremely expensive to include all high order interactions between the explanatory variables, when dealing with high dimensional data.

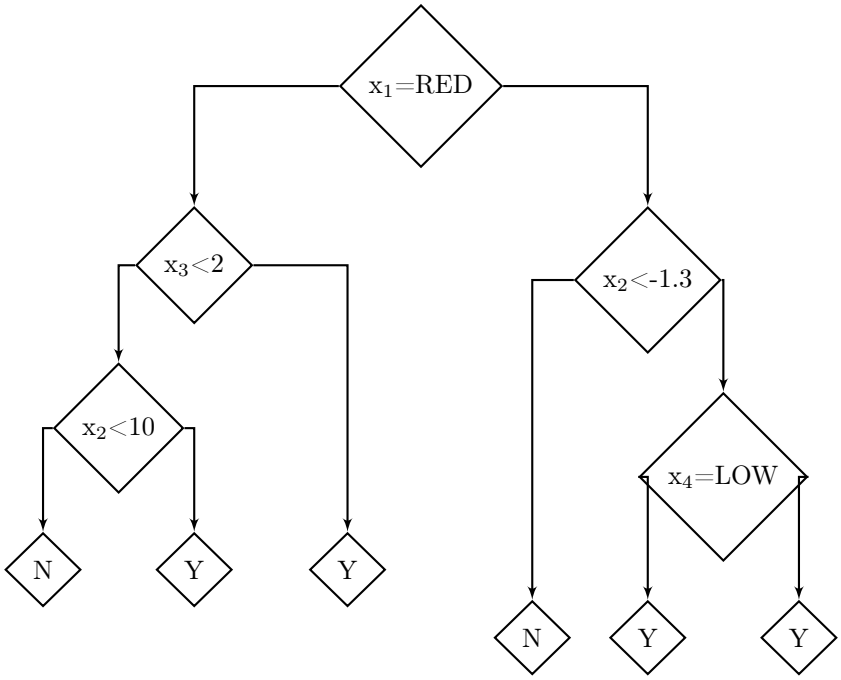


Fig. 1.1.: A decision tree for classification.

1.2.3 Tree-based Methods

Decision trees could be adapted to both regression and classification setting, known as classification and regression Tree. More specifically, the model could be represented as a binary tree, as shown in Figure 1.1. The binary decision tree is constructed in a greedy manner. In each step, the specific split and input variable are chosen to minimize the local loss function. A deep decision tree could easily overfit the data. Common stopping criteria include the maximum depth of the decision tree and the minimum number of samples required to be at a leaf node. Decision tree is easy to interpret, however the algorithm is known to be unstable. Even a small perturbation of input data can cause large changes in the tree.

Random forests are an ensemble learning method for classification, regression and other tasks. Random forests grow many decision tress and output the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees [5, 6]. In addition, the training set of each individual tree is sampled

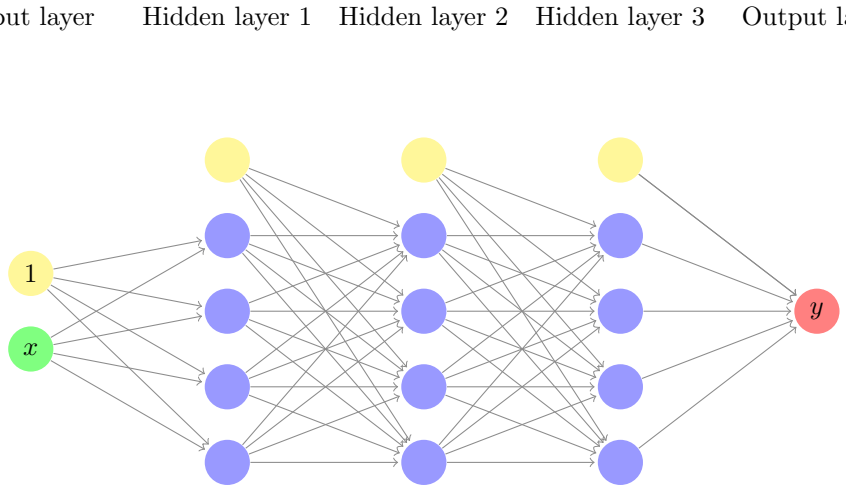


Fig. 1.2.: A 4-layer fully connected neural network.

with replacement from the original data. Thus, random forests significantly decrease the variance of the model by increasing the number of individual trees. Although random forests fail to provide straightforward interpretation as decision trees, they provide a way to measure the importance of any explanatory variable. The importance is evaluated by the damage to the model, if perturbations are added to the given variable.

Another way to improve the performance of decision trees is to combine the weak learners - decision trees into a single stronger learner in an iterative fashion [7–9]. The algorithm optimizes the loss function over function space by iteratively choosing a function (weak leaner) that points in the negative gradient direction [10].

1.2.4 Neural Networks

Neural networks and deep learning currently provide the state-of-the-art solutions to many complex problems especially in image recognition, speech recognition, and natural language processing. A neural network is a machine learning framework inspired by biological neural networks that constitute animal brains. Figure 1.2 gives a 4-layer fully connected neural network, and other popular neural networks include

convolutional neural networks [11] and residual neural networks [12]. Surprisingly, deep neural networks (DNNs) could achieve human or superhuman level performance on tremendous complicated tasks even without any domain knowledge. One interesting observation is that DNNs could disentangle highly curved manifolds in the input space into flattened manifolds in the hidden space [13]. Except the breakthroughs in empirical studies, neural networks are hard to interpret as a black box model, and their theoretical foundations have been less explored in the literature. Especially, overfitting is a notorious problem in deep learning, as it is common to have far more parameters of the model than the training sample size.

1.3 Overfitting

It is easy to perfectly fit the training data, given a large enough model, such as neural networks. However, this could not guarantee a good fitting on the unobserved data. Figure 1.3 gives such an example of overfitting. In this example, data are generated from the orange line with random noise and the model (the blue curve) is fitted on the green dots. As shown in Figure 1.3, even the model fits the training samples perfectly, the model fails when applied to the testing data (red dots).

We will then quantify overfitting by the generalization error. Let $L(f(\cdot), \cdot) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be the loss function. Define the expected and empirical risks, respectively, as

$$\mathbb{E}_L(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[L(f(\boldsymbol{x}), y)], \quad \widehat{\mathbb{E}}_L(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i),$$

where $\mathcal{D}$ is the underlying distribution of $(\boldsymbol{x}, y)$. In practice, the empirical risk is corresponding to the training error, and the expected risk is corresponding to the testing error. Generally, a learning algorithm is said to *overfit* if it is more accurate in fitting known data but less accurate in predicting new data. Mathematically, the difference between the expected risk and the empirical risk, called *generalization error*, is a measure of how accurately an algorithm is able to predict outcome values for

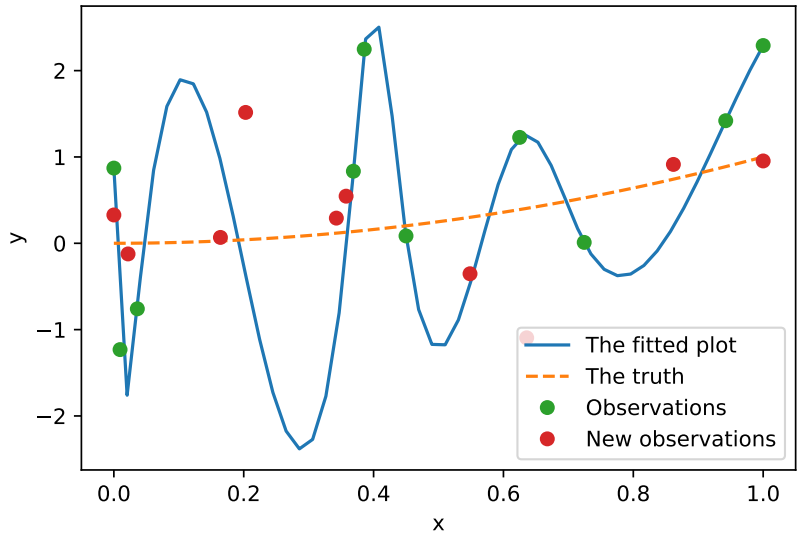


Fig. 1.3.: An overfitting example.

previously unseen data. Generalization error can be minimized by avoiding overfitting in the learning algorithm. Let

$$\mathcal{E}_L(f) = \Big| \mathbb{E}_L(f) - \widehat{\mathbb{E}}_L(f) \Big|. \tag{1.2}$$

Our goal is to control the generalization error $\mathcal{E}_L(f)$ and make it less sensitive to the network architecture when adopting a DNN model. This generalization error bound can be studied using *Rademacher complexity* by some standard techniques in [14], which will be introduced later.

Rademacher complexity is commonly used to measure the complexity of a hypothesis class with respect to a probability distribution $\mathcal{D}$ or a sample, and to analyze generalization bounds [15]. The *empirical Rademacher complexity* of the hypothesis class $\mathcal{F}$ with respect to a data set $S = \{z_1 \dots z_n\}$ is defined as:

$$\widehat{\mathfrak{R}}_S(\mathcal{F}) = \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(z_i) \right) \right],$$

where $\epsilon = \{\epsilon_1 \dots \epsilon_n\}$ are n independent Rademacher random variables. The *Rademacher complexity* of the hypothesis class $\mathcal{F}$ with respect to n samples is defined as:

$$\mathfrak{R}_n(\mathcal{F}) = \mathbb{E}_{S\sim\mathcal{D}^n}\left[\widehat{\mathfrak{R}}_S(\mathcal{F})\right].$$

The following theorem connects the Rademcher complexity to the generalization error bounds under some mild conditions.

Theorem 1.3.1 *Let z be a random variable of support $\mathcal{Z}$ and distribution $\mathcal{D}$. Let $S = \{z_1 \dots z_n\}$ be a data set of n i.i.d. samples drawn from $\mathcal{D}$. Let $\mathcal{F}$ be a hypothesis class satisfying $\mathcal{F} \subseteq \{f \mid f : \mathcal{Z} \to [0, A_0]\}$. Fix $\delta \in (0, 1)$. With probability at least $1 - \delta$ over the choice of S, the following holds for all $h \in \mathcal{F}$:*

$$\left|\mathbb{E}_{\mathcal{D}}[h] - \widehat{\mathbb{E}}_S[h]\right| \leq 2\mathfrak{R}_n(\mathcal{F}) + A_0\sqrt{\frac{\log(2/\delta)}{2n}}$$

Proof According to [14][Theorem 3.1], fix $\delta \in (0, 1)$. With probability at least $1 - \frac{\delta}{2}$ over the choice of S, the following holds for all $h \in \mathcal{F}$:

$$\mathbb{E}_{\mathcal{D}}[h/A_0] - \widehat{\mathbb{E}}_S[h/A_0] \leq 2\mathfrak{R}_n(\mathcal{F}/A_0) + \sqrt{\frac{\log(2/\delta)}{2n}}$$

With probability at least $1 - \frac{\delta}{2}$ over the choice of S, the following holds for all $h \in \mathcal{F}$:

$$\mathbb{E}_{\mathcal{D}}[-h/A_0] - \widehat{\mathbb{E}}_S[-h/A_0] \leq 2\mathfrak{R}_n(-\mathcal{F}/A_0) + \sqrt{\frac{\log(2/\delta)}{2n}}$$

By the definition of Rademacher complexity, $\mathfrak{R}_n(-\mathcal{F}/A_0) = \mathfrak{R}_n(\mathcal{F}/A_0) = \mathfrak{R}_n(\mathcal{F})/A_0$. Thus we complete the proof. ■

2. WEIGHT NORMALIZED DEEP NEURAL NETWORKS

2.1 Introduction

During the past decade, deep neural networks have demonstrated an amazing performance in solving many complex artificial intelligence tasks such as object recognition and identification, text understanding and translation, question answering, and more [16]. The capacity of *unregularized* fully connected DNNs, as a function of the network size and depth, is fairly well understood [17–19]. By bounding the $L_{2,\infty}$ norm of the incoming weights of each unit, [20] is able to accelerate the convergence of stochastic gradient descent optimization across applications in supervised image recognition, generative modeling, and deep reinforcement learning. However, theoretical investigations on such networks are less explored in the literature, and a few exceptions are [18, 21–25]. There is a central question waiting for an answer: Can we bound the capacity of fully connected DNNs **with bias neurons** by weight normalization alone, which has the least dependence on the architecture?

We study a general class of the weight normalized deep neural networks, including all layer-wise $L_{p,q}$ weight normalizations. These networks have a bias neuron per hidden layer, while prior studies [18, 21–25] only include the bias neuron in the input layer, which differs from the practical application. More discussions of the issues on bias neurons are presented in Section 2.2. We establish the upper bound on the Rademacher complexities of this family. In addition, we study the theoretical properties of WN-DNNs with rectified linear units in terms of the approximation error.

We first examine how the architecture of $L_{p,q}$ WN-DNNs influences their generalization properties. Specifically, for the $L_{1,\infty}$ WN-DNNs, we obtain a complexity bound that is independent of the width and has a square root dependence on the

depth k. Furthermore, the $O(\sqrt{k})$ term could be reduced to $O(\log k)$, if the activation functions are anti-symmetric. We will demonstrate later that it is nontrivial to extend the existing results to the DNNs with bias neurons.

Norm-constrained fully connected DNNs with no bias neuron were investigated in prior studies [18, 21–25]. Especially, spectral norm-constrained fully connected DNNs with no bias neurons were studied in [22, 24, 25]. Assume that the spectral norm of the weight matrix in each layer equals to 1, and the width of each hidden layer is d. Then the corresponding generalization bound is $O(\frac{\sqrt{k^3d^2}}{\sqrt{n}})$ [22, 24] and $O(\frac{\sqrt{kd^2}}{\sqrt{n}})$ [25], where n is the sample size, and k the depth. On the one hand, our result has a lower dependence on both the width and the depth of the neural network. Especially, we derive a generalization bound, which is independent of the width, and relies on the depth k by at most $O(\sqrt{k})$. On the other hand, it is easy to create a matrix, of which the spectral norm is greater than the $L_{1,\infty}$ norm and vice versa. Thus it is difficult to compare our results with the works of [22, 24, 25].

The $L_{p,q}$ norm-constrained fully connected DNNs with no bias neuron were studied in [18, 21, 23, 26]. Although we do not restrict ourselves to the $L_{1,\infty}$ WN-DNNs, we consider this simple case in order to compare with existing results in the literature. Assume that the $L_{1,\infty}$ norm of the weight matrix in each layer equals to 1, and all the activation functions are 1-Lipschitz continuous. As shown in Table 2.1, we extend previous works to WN-DNNs with bias neurons by separating the bias neuron from the hidden layer in each inductive step. In addition, their original results are included in Table 2.1. Note that if the activation functions are ReLU, one could achieve exactly the same generalization bounds by treating each bias neuron as a hidden neuron. However, this trick fails for other activation functions, such as tanh. Because 1 is not even in the range of the activation function tanh. In comparison, our generalization bound is $O(\frac{\log(k)}{\sqrt{n}})$ if the activation functions are tanh, since tanh is anti-symmetric.

For WN-DNNs with ReLU, we provide the exact characterization of its corresponding function class and examine the approximation properties. It is shown that

Table 2.1.: Rademacher complexity bounds of $L_{1,\infty}$ WN-DNNs with/without bias neurons.

Activation Function	Anti-symmetric		ReLU		Other	
With Bias Neurons	Y	N	Y	N	Y	N
[18]	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$
[21]	$O(\frac{k}{\sqrt{n}})$	$O(\frac{1}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$
[23]	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$	$O(\frac{2^k}{\sqrt{n}})$
[26]	$O(\frac{\sqrt{k^3}}{\sqrt{n}})$	$O(\frac{\sqrt{k}}{\sqrt{n}})$	$O(\frac{\sqrt{k}}{\sqrt{n}})$	$O(\frac{\sqrt{k}}{\sqrt{n}})$	$O(\frac{\sqrt{k^3}}{\sqrt{n}})$	$O(\frac{\sqrt{k}}{\sqrt{n}})$
Our results	$O(\frac{\log k}{\sqrt{n}})$		$O(\frac{\sqrt{k}}{\sqrt{n}})$		$O(\frac{\sqrt{k}}{\sqrt{n}})$	

the $L_{1,\infty}$ WN-DNN with ReLU is able to approximate any Lipschitz continuous function arbitrarily well by increasing the norm of its output layer and growing its size. Early work on neural network approximation theory includes the universal approximation theorem [27–29]. This indicates that a fully connected network with a single hidden layer can approximate any continuous functions. More recent work expands the result of shallow networks to deep networks with an increased interest in the expressive power of deep networks especially for some families of "hard" functions [30–35]. For instance, [34] shows that for any positive integer l, there exist neural networks with $\Theta(l^3)$ layers and $\Theta(1)$ nodes per layer which can not be approximated by networks with $\Theta(l)$ layers unless they possess $\Omega(2^l)$ nodes. These results on the other hand request for an artificial neural network of which the generalization bound grows slowly with the depth, and even avoid explicit dependence on the depth.

The contributions of this chapter are summarized as follows.

1. We extend the $L_{2,\infty}$ weight normalization [20] to more general $L_{p,q}$ WN-DNNs, and relate these classes to those represented by unregularized DNNs.

2. We include a bias node not only for the input layer but also for every hidden layer. As discussed in Claim 1, it is nontrivial to extend prior research to study this case.

3. We study the Rademacher complexities of WN-DNNs. Especially, with any $L_{1,\infty}$ normalization, we have a capacity control that is independent of the width and depends on the depth by $O(\sqrt{k})$. Furthermore, if the activation functions are anti-symmetric, the capacity control is independent of the both the width and the depth up to a log factor.

4. When the activation functions are ReLU, we characterize the function class and analyze the approximation property of the $L_{p,q}$ WN-DNNs.

The chapter is organized as follows. In Section 2, we define the $L_{p,q}$ WN-DNNs, and analyze the corresponding function class. Section 3 discusses the Rademacher complexities for this class. In Section 4, we study WN-DNNs with ReLU in terms of functional characterization and approximation.

2.2 Notation and Motivation

In this section, we introduce some notation, which will be used in the remainder of the chapter. In addition, we motivates the introduction of WN-DNNs with a concrete example.

2.2.1 Notation

The Lp, q norm of a $s_1 \times s_2$ matrix A is defined as

$$\|A\|_{p,q} = \left(\sum_{j=1}^{s_2} \left(\sum_{i=1}^{s_1} |a_{ij}|^p \right)^{q/p} \right)^{1/q},$$

where $1 \leq p < \infty$ and $1 \leq q \leq \infty$. Define p^* by $\frac{1}{p} + \frac{1}{p^*} = 1$. When $q = \infty$, $\|A\|_{p,\infty} = \sup_j \left(\sum_{i=1}^{s_1} |a_{ij}|^p \right)^{1/p}$. When $p = q = 2$, the $L_{p,q}$ is the Frobenius norm.

A function $\sigma : \mathbb{R} \to \mathbb{R}$ is *positive homogeneous* if $\sigma(cu) = c\sigma(u)$ for any $c > 0$ and $u \in \mathbb{R}$, and is *anti-symmetric* if $\sigma(-u) = -\sigma(u)$ for any $u \in \mathbb{R}$. Assume that two vectors $\boldsymbol{o}^1$ and $\boldsymbol{o}^2$ have the same length. Then $\boldsymbol{o}^1 \leq \boldsymbol{o}^2$ if $o_j^1 \leq o_j^2$, $\forall j$. For any activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and $\mathbf{u} \in \mathbb{R}^J$, define $\sigma \circ \mathbf{u} = (\sigma(u_1), \cdots, \sigma(u_J))$. For an affine transformation $T(\mathbf{u}) = \mathbf{V}^T(1, \mathbf{u}^T)^T$, a norm $\|\cdot\|_*$, and $p, q \geq 1$, define $\|T\|_* = \|\mathbf{V}\|_*$. Especially for the $L_{p,q}$ norm, define $\|T\|_{p,q} = \|\mathbf{V}\|_{p,q}$ and $\|T[,j]\|_p = \|\mathbf{V}[,j]\|_p$, where $\mathbf{V}[,j]$ is the jth column of the matrix $\mathbf{V}$.

Fully connected neural networks. A fully connected neural network on $\mathbb{R}^{d_0} \to \mathbb{R}^{d_{k+1}}$ with k hidden layers is defined by a set of $k+1$ affine transformations $T_1 : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}, T_2 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}, \cdots, T_{k+1} : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$ and a series of activation functions $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_k)$. The affine transformations are parameterized by $T_i(\mathbf{u}) = \boldsymbol{W}_i^T \mathbf{u} + \mathbf{B}_i$, where $\boldsymbol{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}, \mathbf{B}_i \in \mathbb{R}^{d_i}$ for $i = 1. \cdots, k+1$. The function represented by this neural network is

$$f = T_{k+1} \circ \sigma_k \circ T_k \circ \cdots \circ \sigma_1 \circ T_1 \circ \boldsymbol{x}.$$

Before introducing the $L_{p,q}$ WN-DNNs, we build an augmented layer for each hidden layer by appending the bias neuron 1 to the original layer. Then combine the weight matrix and the bias vector as a new matrix.

1. Define the first hidden layer

$$f_1(\boldsymbol{x}) = T_1 \circ \boldsymbol{x} \triangleq \langle \tilde{\mathbf{V}}_1, (1, \boldsymbol{x}^T)^T \rangle,$$

   where $\tilde{\mathbf{V}}_1 = (\mathbf{B}_1, \boldsymbol{W}_1^T)^T \in \mathbb{R}^{(d_0+1) \times d_1}$.

2. Sequentially for $i = 2, \cdots, k$, define the ith hidden layer as

$$f_i(\boldsymbol{x}) = T_i \circ \sigma_{i-1} \circ f_{i-1}(\boldsymbol{x}) \triangleq \langle \tilde{\mathbf{V}}_i, (1, \sigma_{i-1} \circ f_{i-1}^T(\boldsymbol{x}))^T \rangle, \tag{2.1a}$$

   where $\tilde{\mathbf{V}}_i = (\mathbf{B}_i, \boldsymbol{W}_i^T)^T \in \mathbb{R}^{(d_{i-1}+1) \times d_i}$.

3. The output layer is

$$f(\boldsymbol{x}) = T_{k+1} \circ \sigma_k \circ f_k(\boldsymbol{x}) \triangleq \langle \tilde{\mathbf{V}}_{k+1}, (1, \sigma_k \circ f_k^T(\boldsymbol{x}))^T \rangle, \tag{2.1b}$$

   where $\tilde{\mathbf{V}}_{k+1} = (\mathbf{B}_{k+1}, \boldsymbol{W}_{k+1}^T)^T \in \mathbb{R}^{(d_k+1) \times d_{k+1}}$.

2.2.2 Motivation

We motivate the introduction of WN-DNNs with a toy example. This example shows that the product of Frobenius norms of all layers fails to control the output of a neural network.

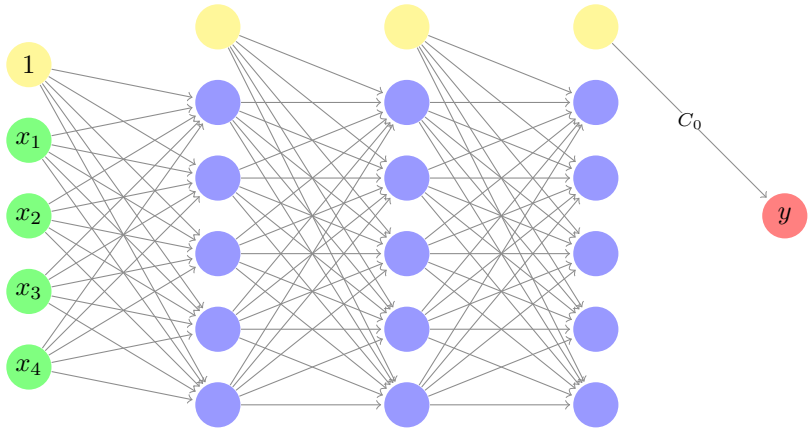


Fig. 2.1.: A motivating example: visualization of f_0.

As shown in Figure 2.1, define a 4-layer neural network $f_0 : \mathbb{R}^4 \to \mathbb{R} = T_4 \circ \sigma_3 \circ T_3 \circ \sigma_2 \circ T_2 \circ \sigma_1 \circ T_1$, where $T_1 : \mathbb{R}^4 \to \mathbb{R}^5$, $T_2 : \mathbb{R}^5 \to \mathbb{R}^5$,$T_3 : \mathbb{R}^5 \to \mathbb{R}^5$, and $T_4 : \mathbb{R}^5 \to \mathbb{R}$. We will show that the output of f_0 could approach infinity even if the product of Frobenius norms of all layers is constant 1. We first create a T_1, T_2 and T_3 such that $\|T_1\|_F = \frac{1}{C_0}$ and $\|T_2\|_F = \|T_3\|_F = 1$. As shown in Figure 2.1, the output only depends on the bias neuron in the last hidden layer. It is easy to verify that $\prod_{i=1}^{4} \|T_i\|_F = 1$ and $f_0 \equiv C_0$. The output is unbounded, as C_0 could be any positive number. The above example indicates the need to introduce weight normalization in addition to the constrained product of Frobenius norms. Furthermore, the result can be extended to arbitrary norms.

Claim 1 *Fix $k \in \mathbb{N}$, $\boldsymbol{d} = (d_0, d_1, \cdots, d_k, 1) \in \mathbb{N}_+^{k+1}$, a norm $\|\cdot\|_*$, and k activation functions $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_k)$. Define $\mathbb{N}_{\gamma_* \le \gamma}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$ as the collection of all neural networks*

$f = T_{k+1} \circ \sigma_k \circ T_k \circ \cdots \circ \sigma_1 \circ T_1 \circ \boldsymbol{x}$ such that the affine function $T_i(\mathbf{u})$ is defined on $\mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ for $i = 1, \cdots, k+1$ and

$$\gamma_* = \prod_{i=1}^{k+1} \|T_i\|_* \leq \gamma.$$

Then for any sample $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathbb{R}^{m_1}$,

$$\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}) = \infty.$$

Proof We first show that for any $\gamma_0 > 0$, any activation functions $\boldsymbol{\sigma}$, any norm $\|\cdot\|_*$, and any $C_0 \in \mathbb{R}$, there exists a function $f_T = T_{k+1} \circ \sigma_k \circ T_k \circ \cdots \circ \sigma_1 \circ T_1 \circ \boldsymbol{x}$, such that $f_T \equiv C_0$ and $\prod_{i=1}^{k+1} \|T_i\|_* \leq \gamma_0$. Assume that $\|(1, 0, \cdots, 0)\|_* = a_0$. Note that $a_0 > 0$ by the definition of the norm. Choose the first affine function T_1 such that $\|T_1\|_* = \frac{\gamma_0}{a_0 C_0}$. Then for $i = 2, \cdots, k$, generate affine functions T_i satisfying that $\|T_i\|_* = 1$. Finally, define the output layer as $T_{k+1} = C_0$. Note that $f_T \equiv C_0$, and

$$\prod_{i=1}^{k+1} \|T_i\|_* \leq \frac{\gamma_0}{a_0 C_0} * 1^{k-1} * a_0 C_0 = \gamma_0.$$

Based on the above, we have

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}) &= \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}} \left(\frac{1}{n} \sum_{i=1}^n \epsilon_i f(z_i) \right) \right] \\
&= \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}} \max \left(\frac{1}{n} \sum_{i=1}^n \epsilon_i f(z_i), \frac{1}{n} \sum_{i=1}^n \epsilon_i (-f(z_i)) \right) \right] \\
&= \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}} \left| \frac{1}{n} \sum_{i=1}^n \epsilon_i f(z_i) \right| \right] \\
&= \mathbb{E} \left[\mathbb{E} \left[\sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}} \left| \frac{1}{n} \sum_{i=1}^n \epsilon_i f(z_i) \right| \middle| \sum_{i=1}^n \epsilon_i \right] \right] \\
&\geq \mathbb{P}(\sum_{i=1}^n \epsilon_i \neq 0) \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{\gamma_* \leq \gamma}} \left| \frac{1}{n} \sum_{i=1}^n \epsilon_i f(z_i) \right| | \sum_{i=1}^n \epsilon_i \neq 0 \right]
\end{aligned}$$

$$\geq \frac{1}{2}\mathbb{E}_\epsilon \left[\sup_{f\in\mathcal{N}^{k,}_{\gamma_*\leq\gamma}\boldsymbol{d},\boldsymbol{\sigma}} \left| \frac{1}{n}\sum_{i=1}^{n}\epsilon_i f(z_i) \right| \middle| \sum_{i=1}^{n}\epsilon_i \neq 0 \right] \tag{2.2a}$$

$$\geq \frac{1}{2}\mathbb{E}_\epsilon \left[\sup_{C_0>0} \left(\frac{1}{n}\sum_{i=1}^{n}\epsilon_i sgn(\sum_{i=1}^{n}\epsilon_i)C_0 \right) | \sum_{i=1}^{n}\epsilon_i \neq 0 \right]$$

$$= \infty,$$

where the step in Equation (2.2a) follows from the observation that $\mathbb{P}(\sum_{i=1}^{n}\epsilon_i \neq 0) = 1$ if n is an odd number, and $\mathbb{P}(\sum_{i=1}^{n}\epsilon_i \neq 0) = 1 - \frac{1}{2}\mathbb{P}(\sum_{i=2}^{n}\epsilon_i = 1) - \frac{1}{2}\mathbb{P}(\sum_{i=2}^{n}\epsilon_i = -1) \geq \frac{1}{2}$ if n is an even number. ■

Prior studies [18, 21–26] included the bias neuron only in the input layer and considered layered networks parameterized by a sequence of weight matrices only, that is $\mathbf{B}_i = \mathbf{0}$ for all $i = 1, \cdots, k+1$. While fixing the architecture of neural networks, these works imply that $\prod_{i=1}^{k+1}\|\mathbf{W}_i\|_*$ is sufficient to control the Rademacher complexity of the function class represented by these DNNs. The $\|\cdot\|_*$ norm of $\mathbf{W}$ is the spectral norm of $\mathbf{W}^T$ in [22, 24, 25], the $L_{1,\infty}$ norm of $\mathbf{W}$ in [18, 23], the $L_{1,\infty}/L_{2,2}$ norm in [26], and the $L_{p,q}$ norm in [21], where $p \in [1, \infty)$ and $q \in [1, \infty]$. However, this kind of control fails once the bias neuron is added to each hidden layer, demonstrating the necessity to use WN-DNNs instead.

2.3 The $L_{p,q}$ WN-DNNs

In this section, we introduce the WN-DNN, which includes all layer-wise $L_{p,q}$ weight normalizations. We begin with the definition of the fully connected neural networks.

Definition 2.3.1 *A $k+1$-layer $L_{p,q}$ WN-DNN*

$$f(x) = T_{k+1} \circ \sigma_k \circ T_k \circ \cdots \circ \sigma_1 \circ T_1 \circ \boldsymbol{x}$$

by a normalization constant c is defined by $k+1$ affine transformations $T_1 : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}, T_2 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}, \cdots, T_{k+1} : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$, and k activation functions $\sigma_1, \cdots, \sigma_k$, such that $\|T_i\|_{p,q} \leq c$ for $i = 1, \cdots, k$.

Furthermore, define $\mathcal{N}_{p,q,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$ *as the collection of all functions that could be represented by an* $L_{p,q}$ *WN-DNN with the normalization constant* c *and the activation functions* $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_k)$ *such that*

(a) It has k *hidden layers;*

(b) The number of neurons in the i*th hidden layer is* d_i *for* $i = 1, 2, \cdots, k$*. The dimension of input is* d_0*, and output* d_{k+1}*;*

(c) $\|T_i\|_{p,q} \leq c$ *for* $i = 1, \cdots, k$*;*

(d) $\|T_{k+1}[, j]\|_p \leq o_j$ *for* $j = 1, \cdots, d_{k+1}$*.*

Proposition 2.3.1 *If all the activation functions are positive homogeneous, Part* (c) *of Definition 2.3.1 is equivalent to* $\|T_i\|_{p,q} = c$ *for* $i = 1, \cdots, k$*.*

Next, we provides some useful observations regarding $\mathcal{N}_{p,q,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$.

Theorem 2.3.1 *Let* $c, c_1, c_2 > 0$*,* $\boldsymbol{o}, \boldsymbol{o}^1, \boldsymbol{o}^2 \in \mathbb{R}_+^{d_{k+1}}$*,* $p \in [1, \infty)$*,* $q \in [1, \infty]$*,* $k \in \mathbb{N}$*,* $\boldsymbol{d} = (d_0, d_1 \cdots, d_{k+1}) \in \mathbb{N}_+^{k+2}$*,* $\boldsymbol{d}^1 = (d_0, d_1^1 \cdots, d_k^1, d_{k+1}) \in \mathbb{N}_+^{k+2}$*, and* $\boldsymbol{d}^2 = (d_0, d_1^2 \cdots, d_k^2, d_{k+1}) \in \mathbb{N}_+^{k+2}$*.*

(a) $\mathcal{N}_{p,q,c_1,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}} \subseteq \mathcal{N}_{p,q,c_2,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$ *if* $c_1 \leq c_2$*.* $\mathcal{N}_{p,q,c,\boldsymbol{O}^1}^{k,\boldsymbol{d},\boldsymbol{\sigma}} \subseteq \mathcal{N}_{p,q,c,\boldsymbol{O}^2}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$ *if* $\boldsymbol{o}^1 \leq \boldsymbol{o}^2$*. If* $g \in \mathcal{N}_{p,q,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$*, then* $\alpha g \in \mathcal{N}_{p,q,c,\alpha\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$*.*

(b) $\mathcal{N}_{p_1,q,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}} \subseteq \mathcal{N}_{p_2,q,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$ *if* $1 \leq p_1 \leq p_2 < \infty$*.* $\mathcal{N}_{p,q_1,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}} \subseteq \mathcal{N}_{p,q_2,c,\boldsymbol{O}}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$ *if* $1 \leq q_1 \leq q_2 \leq \infty$*.*

(c) $\mathcal{N}_{p,q,c,\boldsymbol{O}}^{k,\boldsymbol{d}^1,\boldsymbol{\sigma}} \subseteq \mathcal{N}_{p,q,c,\boldsymbol{O}}^{k,\boldsymbol{d}^2,\boldsymbol{\sigma}}$ *if* $d_i^2 \geq d_i^1$ *for* $i = 1, \cdots, k$*.*

(d) If $f_1 \in \mathcal{N}_{p,q,c,\boldsymbol{O}^1}^{k,\boldsymbol{d}^1,\boldsymbol{\sigma}}, f_2 \in \mathcal{N}_{p,q,c,\boldsymbol{O}^2}^{k,\boldsymbol{d}^2,\boldsymbol{\sigma}}$*, then* $f_0 = (f_1, f_2) \in \mathcal{N}_{p,q,c_0,\boldsymbol{O}^0}^{k,\boldsymbol{d}^0,\boldsymbol{\sigma}}$*, where* $c_0 = 2^{\frac{1}{q}} c, \boldsymbol{o}^0 = (\boldsymbol{o}^1, \boldsymbol{o}^2)$*,* $d_0^0 = d_0$*,* $d_i^0 = d_i^1 + d_i^2$ *for* $i = 1, \cdots, k+1$*. Especially, when* $q = \infty$*,* $c_0 = c$*.*

Proof Part (a) is straightforward from the definition.

For Part (b), note that $\|\cdot\|_{p_1} \geq \|\cdot\|_{p_2}$ when $p_1 \leq p_2$, hence $\{\boldsymbol{v} : \|\boldsymbol{v}\|_{p_1} \leq C\} \subseteq \{\boldsymbol{v} : \|\boldsymbol{v}\|_{p_2} \leq C\}$. Then the first line of Part (b) follows from this observation above and the conclusion of Part 1.

Regarding Part (c), for any $g \in \mathcal{N}^{k,\boldsymbol{d}^1}_{p,q,c,\boldsymbol{O}}$, we could add $d_i^2 - d_i^1$ neurons in each hidden layer with no connection to other neurons, thus not increasing the norm of each layer, and this neural network belongs to $\mathcal{N}^{k,\boldsymbol{d}^2}_{p,q,c,\boldsymbol{O}}$. Regarding Part (d), assume

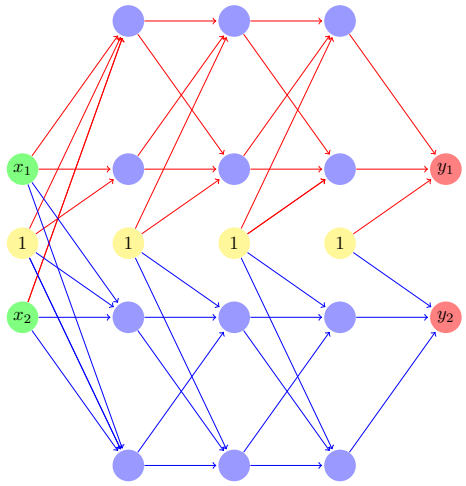


Fig. 2.2.: Concatenate two neural networks.

$f_j(\mathbf{u}) = T^j_{k+1} \circ \sigma_k \circ T^j_k \cdots \circ \sigma_1 \circ T^j_1 \circ \mathbf{u}$ for $j = 1, 2$, where $T^j_i(\mathbf{u}) = (\mathbf{W}^j_i)^T\mathbf{u} + \mathbf{B}^j_i$. By their definitions, $\left\|(\mathbf{B}^j_1, (\mathbf{W}^j_1)^T)^T\right\|_{p,q} \leq c$ for $i = 1, \cdots, k$ and $j = 1, 2$. As shown in Figure 2.2, the idea is to combine f_1 and f_2 with no connections between their hidden layers. We then provide the rigorous proof to construct $(f_1(\mathbf{u}), f_2(\mathbf{u})) = T_{k+1} \circ \sigma_k \circ T_k \cdots \circ \sigma_1 \circ T_1 \circ \mathbf{u}$, where $T_i(\mathbf{u}) = \mathbf{W}^T_i\mathbf{u} + \mathbf{B}_i$. Define

$$\mathbf{B}_1 = \begin{pmatrix} \mathbf{B}^1_1 \\ \mathbf{B}^2_1 \end{pmatrix},$$

and

$$\mathbf{W}_1 = \begin{pmatrix} \mathbf{W}^1_1 & \mathbf{W}^2_1 \end{pmatrix}.$$

For $i = 2, \cdots, k+1$, define

$$\mathbf{B}_i = \begin{pmatrix} \mathbf{B}_i^1 \\ \mathbf{B}_i^2 \end{pmatrix},$$

and

$$\mathbf{W}_i \in \mathbb{R}^{(d_{i-1}^1 + d_{i-1}^2) \times (d_i^1 + d_i^2)} = \begin{pmatrix} \mathbf{W}_i^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_i^2 \end{pmatrix}.$$

Note that $\|T_j\|_{p,q} = \left(\left\|T_j^1\right\|_{p,q}^q + \left\|T_j^2\right\|_{p,q}^q \right)^{\frac{1}{q}} \leq 2^{\frac{1}{q}} c$ for $j = 1, \cdots, k$, and $\|T_{k+1}[,l]\|_p \leq o_l$ for $l = 1, \cdots, d_{k+1}^1 + d_{k+1}^2$, where $\boldsymbol{o} = (\boldsymbol{o}^1, \boldsymbol{o}^2)$. ■

In Theorem 2.3.1, Part (a) shows the increased expressive power of neural networks by either a larger normalization constant or norm constraints of the output vectors. Part (b) discusses how the choice of $L_{p,q}$ normalization influence its representation capacity. Part (c) describes the gain in representation power by widening the neural networks. However, deepening the neural network does not essentially enlarge the function class of WN-DNNs. It is not guaranteed that $\sigma \circ \sigma \circ f(\boldsymbol{x}) = \sigma \circ f(\boldsymbol{x})$, where $f(\boldsymbol{x}) : \mathbb{R}^s \to \mathbb{R}^t$ is a function and $\boldsymbol{x} \in \mathbb{R}^s$. Thus it could be impossible to get the $\sigma \circ f(\boldsymbol{x})$ with one additional hidden layer. One exception is ReLU. As long as the normalization constant $c \geq 1$, the increase of the depth will lead to a larger function class of WN-DNNs. Part (d) indicates that the concatenation of two WN-DNNs is still a WN-DNN.

2.4 Estimating the Rademacher Complexities

In this section, we will provide bounds on the Rademacher complexities of $\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}$. We makes the following assumptions for all the theorems in this section.

- The number of hidden layers is $k \in \mathbb{N}$, the normalization constant is $c > 0$, and the constraint of the output layer is $o > 0$. The widths are $d_0 = m_1$, $d_i \in \mathbb{N}_+$ for $i = 1, \cdots, k$, and $d_{k+1} = 1$.

- The activation functions are $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_k)$. The ith activation function σ_i is ρ_i Lipschitz continous, and $\sigma_i(0) = 0$ for $i = 1, \cdots, k$.

It is shown that the complexity bound is independent of the width when $p = 1$ in the following theorem. Otherwise it depends on the width by $O\left(\prod_{i=1}^{k} d_i^{\frac{1}{p^*}}\right)$. This dependence could be reduced to $O\left(\prod_{i=1}^{k} d_i^{[\frac{1}{p^*}-\frac{1}{q}]_+}\right)$ if the activation functions are positive homogeneous, which will be addressed in Theorem 2.4.3.

Theorem 2.4.1 *For any set $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$, we have*

(a) for $p \in (1, 2]$,

$$\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq o\sqrt{\frac{(k+1)\log 16}{n}} \left(\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{\frac{1}{p^*}}\right) + \frac{1}{\sqrt{n}} o c^k \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} \left(\sqrt{(k+1)\log 16} \max_i \|\boldsymbol{x}_i\|_{p^*} + \min\{\sqrt{p^*-1} \max_i \|\boldsymbol{x}_i\|_{p^*}, m_1^{\frac{1}{p^*}} \sqrt{2\log(2m_1)} \max_i \|\boldsymbol{x}_i\|_\infty\}\right), \tag{2.3}$$

(b) for $p \in 1 \cup (2, \infty)$,

$$\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq o\sqrt{\frac{(k+1)\log 16}{n}} \left(\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{\frac{1}{p^*}}\right) + \frac{o c^k}{\sqrt{n}} \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} * \left[m_1^{\frac{1}{p^*}} \sqrt{2\log(2m_1)} \max_i \|\boldsymbol{x}_i\|_\infty + \sqrt{(k+1)\log 16} \max_i \|\boldsymbol{x}_i\|_{p^*}\right]. \tag{2.4}$$

Theorem 2.4.1 indicates that the generalization bound depends explicitly on the depth by $O(\sqrt{k+1})$. However, the dependence could increase up to $O((k+1)^{\frac{3}{2}})$, if $c\rho_\ell \geq 1$ for $l = 1, \cdots, k$. As a comparison, [26] investigated the Rademacher complexity of the $L_{1,\infty}$ and $L_{2,2}$ WN-DNNs with the rectified linear units (ReLU) and no bias neurons, which depends on the depth by $O(\sqrt{k+1})$. ReLU could map 1 exactly to 1, thus we could treat the bias neuron as a hidden neuron and obtain the similar result by applying the conclusion of Theorem 2 in [26]. But we fail to apply the same

trick for other activation functions, such as tanh. Because the bias neuron 1 is not even included in the range of tanh. Then there is a question: Is the inclusion of bias neurons lead to extra dependence of generalization bounds on the depth? We will address this question in the following theorem. Furthermore, we will provide a complexity bound for $L_{1,\infty}$ WN-DNNs that matches [26], not only for ReLU, but also for other activation functions.

Corollary 1 *Assume that $c\rho_i \geq 1$ for $i = 1, \cdots, k$. For any set $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$ and $q \geq 1$, we have*

$$\widehat{\mathfrak{R}}_S(\mathcal{N}_{1,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq \frac{1}{\sqrt{n}} o c^k \prod_{i=1}^{k} \rho_i \max_i \left\|(1, \boldsymbol{x}_i^T)\right\|_\infty \left(\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}\right).$$

In particular, when $c\rho_i = 1$ for $i = 1, \cdots, k$, the output layer constraint is $o = 1$, and $\|\boldsymbol{x}_j\|_\infty \leq 1$ for $j = 1, \cdots, n$, the above bound becomes

$$\frac{1}{\sqrt{n}}\left(\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}\right),$$

which could be further reduced if the activation functions are anti-symmetric as shown in Theorem 2.4.2 and Corollary 2. Furthermore, there will be no explicit dependence on either the depth or the width outside of the log factor for $L_{1,q}$ WN-DNNs, where $q \geq 1$.

Theorem 2.4.2 *Assume that all the activation functions are anti-symmetric. For any set $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$, we have*

(a) for $p \in (1, 2]$,

$$\begin{aligned}\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq\ & o\sqrt{\frac{\log 16}{n}} \left(\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{\frac{1}{p^*}}\right) + \\ & \frac{1}{\sqrt{n}} o c^k \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} \left(\sqrt{(k+1)\log 16} \max_i \|\boldsymbol{x}_i\|_{p^*} \right. \\ & \left. + \min\{\sqrt{p^*-1} \max_i \|\boldsymbol{x}_i\|_{p^*}, m_1^{\frac{1}{p^*}} \sqrt{2\log(2m_1))} \max_i \|\boldsymbol{x}_i\|_\infty\}\right),\end{aligned} \tag{2.5}$$

(b) for $p \in 1 \cup (2, \infty)$,

$$\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) \leq o\sqrt{\frac{\log 16}{n}} \left(\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{\frac{1}{p^*}} \right) + \frac{oc^k}{\sqrt{n}} \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} * \left[m_1^{\frac{1}{p^*}} \sqrt{2\log(2m_1)} \max_i \|\boldsymbol{x}_i\|_\infty + \sqrt{(k+1)\log 16} \max_i \|\boldsymbol{x}_i\|_{p^*} \right]. \tag{2.6}$$

Although there is no explicit dependence on the depth, the term

$$\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{\frac{1}{p^*}}$$

in Theorem 2.4.2 equals to $k+1$ when $p = 1$ and $c\rho_\ell = 1$ for $\ell = 1, \cdots, k$. In comparison, [21] derived an architecture independent upperbound of Rademacher complexity of $L_{1,\infty}$ WN-DNNs without bias neurons under the same assumption. Then there is a question: Is it possible to derive a architecture independent generalization bound for $L_{1,\infty}$ WN-DNNs with bias neurons? In the following theorem, We focus on $L_{1,\infty}$ WN-DNNs and provide an upperbound of the corresponding Rademacher complexity, which is independent of both the depth and the width outside of the log factor.

Corollary 2 *Assume that all the activation functions are anti-symmetric. For any set* $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$ *and* $q \geq 1$ *, we have*

$$\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{1,q,c,o}) \leq \frac{1}{\sqrt{n}} oc^k \prod_{i=1}^{k} \rho_i \max_i \left\|(1, \boldsymbol{x}_i^T)\right\|_\infty (\sqrt{\log(4k+6)} + \sqrt{2\log(2m_1)}).$$

When $p > 1$, all the above theorems suggest a $\prod_{i=1}^{k} d_i^{\frac{1}{p^*}}$ dependence of the generalization upper bound on the width. In the following theorem, this dependence is reduced to $\prod_{i=1}^{k} d_i^{[\frac{1}{p^*} - \frac{1}{q}]_+}$, assuming that the activation functions are positive homogeneous. Especially, the complexity bound is independent of the width when $q \leq p^*$.

Theorem 2.4.3 *Assume that all the activation functions are positive homogeneous. For any set* $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$ *, we have*

(a) for $p \in (1, 2]$,

$$\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) \leq o\sqrt{\frac{(k+1)\log 16}{n}} \left(\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{[\frac{1}{p^*}-\frac{1}{q}]_+} \right) + \frac{1}{\sqrt{n}} o c^k \prod_{i=1}^{k} \rho_i d_i^{[\frac{1}{p^*}-\frac{1}{q}]_+} \left(\sqrt{(k+1)\log 16} \max_i \|\boldsymbol{x}_i\|_{p^*} + \min\{\sqrt{p^*-1} \max_i \|\boldsymbol{x}_i\|_{p^*}, m_1^{\frac{1}{p^*}} \sqrt{2\log(2m_1)} \max_i \|\boldsymbol{x}_i\|_\infty\} \right), \tag{2.7}$$

(b) for $p \in 1 \cup (2, \infty)$,

$$\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) \leq o\sqrt{\frac{(k+1)\log 16}{n}} \left(\sum_{i=1}^{k+1} c^{k-i+1} \prod_{\ell=i}^{k} \rho_\ell d_\ell^{[\frac{1}{p^*}-\frac{1}{q}]_+} \right) + \frac{oc^k}{\sqrt{n}} \prod_{i=1}^{k} \rho_i d_i^{[\frac{1}{p^*}-\frac{1}{q}]_+} * \left[m_1^{\frac{1}{p^*}} \sqrt{2\log(2m_1)} \max_i \|\boldsymbol{x}_i\|_\infty + \sqrt{(k+1)\log 16} \max_i \|\boldsymbol{x}_i\|_{p^*} \right]. \tag{2.8}$$

Although the main results in this section focus on the Rademacher complexity, it is easy to derive the generalization error bounds. It is common to assume that the loss function is Lipschitz continuous on each element of the response variable. If $d_{k+1} = 1$ or the loss function could be decomposed over the elements of the response variable, generalization error bounds are straightforward from the bounds on the Rademacher complexity [15]. Otherwise, one may apply the result of [36] to derive the generalization bound.

2.5 Function Class Characterization and Approximation Properties of WN-DNNs with ReLU

In this section, we focus on WN-DNNs with ReLU, and write $\sigma = \sigma_{\text{ReLU}}$ for convenience. We characterize the function class and provide the error bounds for the approximation error of Lipschitz continuous functions.

2.5.1 Exact Characterization of WN-DNNs with ReLU

The exact characterization of functions representable by neural networks with ReLU has been studied in the literature [30, 37]. On the one hand, any function represented by a neural network with ReLU is a CPWL function. On the other hand, any CPWL function could be represented by a neural network with ReLU. We will then show a similar result for WN-DNNs.

We provide a technical lemma before introducing the main theorem, which might be of separate interest. For any max affine function $g : \mathbb{R}^s \to \mathbb{R}$, there exists a $L_{p,q}$ WN-DNN $f : \mathbb{R}^s \to \mathbb{R}$ such that $f = g$. Max-affine functions [38] are widely used as piecewise linear approximations for multivariate functions to fit multi-dimensional data, especially in convex optimization.

Lemma 1 *Assume that* $g(\mathbf{u}) = \max\{\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2, \cdots, \boldsymbol{a}_J^T\mathbf{u}+b_J\}$ *is defined on* $\mathbb{R}^s$*, and* $\|\boldsymbol{a}_i\|_1 + |b_i| \leq L$ *for* $i = 1, 2\cdots, J$*. Then* $g \in \mathcal{N}_{p,q,c,2L}^{k,\boldsymbol{d},\sigma_{\mathrm{ReLU}}}$*, where* $k = [\log_2 J]$*,* $d_i = 2^{k+2-i}$ *for* $i = 1, 2, \cdots, k$*, and* $c = 2^{\frac{3}{p}+\frac{k+3}{2q}-1}$*. Especially,* $g \in \mathcal{N}_{p,\infty,2^{3/p},2L}^{k,\boldsymbol{d},\sigma_{\mathrm{ReLU}}}$*, where* $k = [\log_2 J]$*,* $d_i = 2^{k+2-i}$ *for* $i = 1, 2, \cdots, k$*.*

In the following theorem, it will be shown that the function class of WN-DNNs with ReLU is exactly the class of CPWL functions.

Theorem 2.5.1 *Every* $\mathbb{R}^{m_1} \to \mathbb{R}$ $L_{p,q}$ *WN-DNN with ReLU represents a CPWL function. What's more, every CPWL function* $g(\boldsymbol{x}) : \mathbb{R}^{m_1} \to \mathbb{R}$ *could be represented by a* $L_{p,q}$ *WN-DNN for any* $p \in [1, \infty)$ *and* $q \in [1, \infty]$*.*

Proof It is clear from the definition that every $L_{p,q}$ normalized neural network with ReLU represents a CPWL function.

It follows from [Theorem 1]wang2005generalization that every CPWL function could be represented by the difference of two max affine functions. Equivalently, for any CPWL function $g(\boldsymbol{x}) : \mathbb{R}^{m_1} \to \mathbb{R}$, there exists $J_1, J_2 \in \mathbb{N}$, $\boldsymbol{a}_i, \boldsymbol{b}_l \in \mathbb{R}^{m_1}$ and $\boldsymbol{a}_{i0}, \boldsymbol{b}_{l0} \in \mathbb{R}$, where $i = 1, \cdots, J_1$ and $l = 1, \cdots, J_2$, such that

$$g(\boldsymbol{x}) = \max_{i=1,\cdots,J_1} \left(\boldsymbol{a}_i^T\boldsymbol{x} + a_{i0}\right) - \max_{l=1,\cdots,J_2} \left(\boldsymbol{b}_l^T\boldsymbol{x} + b_{l0}\right).$$

Finally the conclusion follows from Lemma 1 and Theorem 2.3.1 (d). ■

2.5.2 Approximation Properties

We will show that any wide one-hidden-layer neural network with ReLU could be exactly represented by a deep but narrow normalized neural network with ReLU. In addition, Lemma 2 indicates that

$$\mathcal{N}_{1,\infty,\cdot,o}^{1,(m_1,r,1)} \subseteq \mathcal{N}_{p,\infty,1,2o}^{k,(m_1,([r/k]+2m_1+3)\mathbf{1}_k,1)}$$

for any $r > 1$, $k, o > 0$, where $[x]$ is the smallest integer which is greater than or equal to x, and $\mathbf{1}_k = (1, \cdots, 1) \in \mathbb{R}^k$.

Lemma 2 *Assume that a function*

$$g(\boldsymbol{x}) : \mathbb{R}^{m_1} \to \mathbb{R} = \sum_{i=1}^{r} c_i \sigma(\boldsymbol{w}_i^T \boldsymbol{x} + b_i),$$

satisfies that $\sum_{i=1}^{r} |c_i| \leq o$ *and* $\left\|(b_i, \boldsymbol{w}_i^T)\right\|_1 = 1$. *Then for any integer* $k \in [1, r]$,

$$g \in \mathcal{N}_{p,q,wid_k^{1/q},2o}^{k,\boldsymbol{d}^k,\sigma_{\mathrm{ReLU}}},$$

where $wid_k = [r/k] + 2m_1 + 3$, $d_0^k = m_1$, $d_i^k = wid_k$ *for* $i = 1, \cdots, k$, *and* $d_{k+1}^k = 1$.

Based on Lemma 2, we establish that a WN-DNN with ReLU is able to approximate any Lipschitz continuous function arbitrarily well by loosing the constraint for the norm of the output layer and either widening or deepening the neural network at the same time. Especially, for $L_{p,\infty}$ WN-DNNs with ReLU, the approximation error could be only controlled by the norm of the output layer, while the $L_{p,\infty}$ norm of each hidden layer is fixed to be one.

Theorem 2.5.2 $f : \mathcal{X} \to \mathbb{R}$, *satisfying that* $\|f\|_\infty \leq L$, *and* $|f(x) - f(y)| \leq L \|x - y\|_\infty$. *Then for any* $p \in [1, \infty)$, $q \in [1, \infty]$, *and any integer*

$$k \in [1, C_r(m_1)(\log \frac{o}{L})^{-2(m_1+1)/(m_1+4)} \left(\frac{o}{L}\right)^{2(m_1+3)/(m_1+4)}],$$

if o is greater than a constant depending only on m_1, there exists a function $h \in \mathcal{N}^{k,\boldsymbol{d}^k,\sigma_{\mathrm{ReLU}}}_{p,q,wid_k^{1/q},2o}$, where

$$wid_k = [k^{-1}C_r(m_1)(\log \frac{o}{L})^{-\frac{2(m_1+1)}{m_1+4}} \left(\frac{o}{L}\right)^{\frac{2(m_1+3)}{m_1+4}}] + 2m_1 + 3,$$

$\boldsymbol{d}^k = (m_1, wid_k, \cdots, wid_k, 1)$, such that

$$\sup_{\|\boldsymbol{x}\|_\infty \le 1} |f(\boldsymbol{x}) - h(\boldsymbol{x})| \le C(m_1)L(\frac{o}{L})^{-\frac{2}{m_1+1}} \log \frac{o}{L},$$

where $C_r(m_1)$ and $C(m_1)$ denotes some constant that depends only on m_1.

2.6 Proofs

In this section, we provide the detailed proofs of the main results. Technical lemmas and their proofs are deferred to the appendix. For simplicity, we define

$$A^p_{m_1,S} = \begin{cases} \min\{\sqrt{p^*-1}\max_i \|\boldsymbol{x}_i\|_{p^*}, m_1^{\frac{1}{p^*}}\sqrt{2\log(2m_1)}\max_i \|\boldsymbol{x}_i\|_\infty\}\sqrt{n}, & p \in (1,2]. \\ \sqrt{2\log(2m_1)n}m_1^{\frac{1}{p^*}}\max_i \|\boldsymbol{x}_i\|_\infty, & p \in \{1\} \cup (2,\infty). \end{cases} \tag{2.9}$$

where $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$ and $\boldsymbol{x}_i \in \mathbb{R}^{m_1}$ for $i = 1.\cdots, m_1$.

2.6.1 Proof of Proposition 2.3.1

Proof We prove by induction on the depth $k+1$. It is trivial when $k = 0$. Let $k = 1$. Consider an arbitrary neural network

$$f \in \mathcal{N}^{1,\boldsymbol{d},\sigma_1}_{p,q,c,\boldsymbol{o}} = T_2 \circ \sigma_1 \circ T_1 \circ \boldsymbol{x},$$

where $T_i(\mathbf{u}) = \mathbf{W}_i^T\mathbf{u} + \mathbf{B}_i$ for $i = 1, 2$. If $\|T_1\|_{p,q} > 0$, define the new affine transformation T_1^* by $T_1^* = sT_1$, where $s = c/\|T_1\|_{p,q}$. Then $\|T_1^*\|_{p,q} = c$. Define the new output layer by $T_2^*(\mathbf{u}) = (\mathbf{W}_2^*)^T\mathbf{u} + \mathbf{B}_2^*$, where

$$\mathbf{W}_2^* = \mathbf{W}_2/s, \mathbf{B}_2^* = \mathbf{B}_2.$$

Since $s \geq 1$, $\|T_2^*[,j]\|_p \leq o_j$, $\forall j$. If $\|T_1\|_{p,q} = 0$, choose an affine transformation T_1^* satisfying that $\|T_1^*\|_{p,q} = c$, and define $T_2^*(\mathbf{u}) = \mathbf{0u} + \mathbf{B}_2$. It is easy to verify that $f = T_2^* \circ \sigma_1 \circ T_1^* \circ \boldsymbol{x}$ and $\|T_2^*[,j]\|_p \leq o_j$, $\forall j$.

Assume that the result holds when $k < K$. When $k = K$, consider a neural network $f(\boldsymbol{x}) \in \mathcal{N}_{p,q,c,\boldsymbol{O}}^{K,\boldsymbol{d},\boldsymbol{\sigma}} = T_{K+1} \circ \sigma_K \circ T_K \circ \cdots \sigma_1 \circ T_1 \circ \boldsymbol{x}$, where $\boldsymbol{d} = (d_0, d_1 \cdots, d_{K+1})$ and $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_K)$. By the inductive hypothesis, its Kth hidden layer

$$f_K(\boldsymbol{x}) \in \mathcal{N}_{p,q,c,\boldsymbol{O}^*}^{K-1,\boldsymbol{d}_K,\boldsymbol{\sigma}_K},$$

where $\boldsymbol{d}_K = (d_0, d_1 \cdots, d_K)$, $\boldsymbol{\sigma}_K = (\sigma_1, \cdots, \sigma_{K-1})$ and $o_j^* = \|T_K[,j]\|_p$. Thus, there exists a series of affine transformations $\{T_i^*\}_{i=1,\cdots,K}$, such that

$$f_K(\boldsymbol{x}) = T_K^* \circ \sigma_{K-1} \circ T_{K-1}^* \circ \cdots \circ \sigma_1 \circ T_1^* \circ \boldsymbol{x},$$

$\|T_i^*\|_{p,q} = c$ for $i = 1, \cdots, K-1$, and $\|T_K^*[,j]\| \leq o_j^*$ $\forall j$. It indicates that

$$f(\mathbf{x}) = T_{K+1} \circ \sigma_K \circ T_K^* \circ \sigma_{K-1} \circ T_{K-1}^* \circ \cdots \circ \sigma_1 \circ T_1^* \circ \boldsymbol{x}.$$

If $\|T_K^*\|_{p,q} > 0$, create the new affine transformation $T_K^{**} = sT_K^*$, where $s = c/\|T_K^*\|_{p,q}$, such that $\|T_K^{**}\|_{p,q} = c$. Define the output layer by $T_{K+1}^*(\mathbf{u}) = (\mathbf{W}_{K+1}^*)^T\mathbf{u} + \mathbf{B}_{K+1}^*$, where

$$\mathbf{W}_{K+1}^* = \mathbf{W}_{K+1}/s, \mathbf{B}_{K+1}^* = \mathbf{B}_{K+1}.$$

Since $\|\boldsymbol{o}^*\|_q \leq c$, $s \geq 1$ and $\left\|T_{K+1}^*[,j]\right\|_p \leq o_j$ $\forall j$. If $\|T_K^*\|_{p,q} = 0$, create an affine transformation T_K^{**} satisfying that $\|T_K^{**}\|_{p,q} = c$, and define the output layer $T_{K+1}^* = \mathbf{B}_{K+1}$. Such T_{K+1}^* satisfies that $\left\|T_{K+1}^*[,j]\right\|_p \leq o_j$ $\forall j$. ■

2.6.2 Proof of Theorem 2.4.1

Proof Fixing the sample S, $p, q \geq 1$, the activation functions $\boldsymbol{\sigma}$, and the architecture of the DNN, define a series of random variables $\{Z_0, Z_1, \cdots, Z_k\}$ as

$$Z_0 = \left\| \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i \right\|_{p^*},$$

and

$$Z_j = \sup_{f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}} \left\| \sum_{i=1}^{n} \epsilon_i \sigma_j \circ f_j(\boldsymbol{x}_i) \right\|_{p^*}$$

for $j = 1, \cdots, k$, where $\{\epsilon_1, \cdots, \epsilon_n\}$ are n independent Rademacher random variables, and f_j denotes the jth hidden layer of the neural network f. The proof has two main steps. In the first step, we prove by induction that for $j = 1, \cdots, k$ and any $t \in \mathbb{R}$

$$\mathbb{E}_\epsilon \exp(\mathrm{tZ_j}) \leq 4^{\mathrm{j}} \exp\left(\frac{\mathrm{t}^2 \mathrm{n s_j^2}}{2} + \mathrm{tc^j} \prod_{\mathrm{i}=1}^{\mathrm{j}} \rho_{\mathrm{i}} \mathrm{d_i^{\frac{1}{p^*}}} \mathrm{A^p_{m_1,S}} \right),$$

where $A^p_{m_1,S}$ is defined in Equation (2.9), and

$$s_j = \sum_{i=1}^{j} c^{j-i+1} \prod_{\ell=i}^{j} \rho_\ell d_\ell^{\frac{1}{p^*}} + \max_i \|\boldsymbol{x}_i\|_{p^*} c^j \prod_{l=1}^{j} \rho_\ell d_\ell^{\frac{1}{p^*}}.$$

Note that $s_{j+1} = c\rho_{j+1} d_{j+1}^{\frac{1}{p^*}} (s_j + 1)$.

The case when $j = 0$ is straightforward from Lemma 6.

When $j \geq 1$,

$$\begin{aligned}
\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_j}\right) &= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\left\|\tilde{\mathbf{V}}_j\right\|_{p,q} \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \circ (\tilde{\mathbf{V}}_{\mathrm{j}}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right\|_{\mathrm{p}^*} \right) \\
&\leq \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_p \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \mathrm{d_j^{\frac{1}{p^*}}} \left| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \left(\boldsymbol{v}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \right| \right) \\
&\leq 2\mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_p \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \mathrm{d_j^{\frac{1}{p^*}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \left(\boldsymbol{v}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \right) \qquad (2.10a) \\
&\leq 2\mathbb{E}_\epsilon \exp\left(\mathrm{t} \rho_{\mathrm{j}} \mathrm{d_j^{\frac{1}{p^*}}} \sup_{\substack{\|\boldsymbol{v}\|_p \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \boldsymbol{v}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \qquad (2.10b)
\end{aligned}$$

$$
\begin{aligned}
&\leq 2\mathbb{E}_{\epsilon}\exp\left(\mathrm{tc}\rho_{\mathrm{j}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}}))\right\|_{\mathrm{p}^*}\right)\\
&\leq 2\mathbb{E}_{\epsilon}\exp\left(\mathrm{tc}\rho_{\mathrm{j}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\left(|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|+\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}})\right\|_{\mathrm{p}^*}\right)\right)\\
&\leq 2\left[\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j tc}\rho_{\mathrm{j}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|\right)\right]^{\frac{1}{r_j}}*\\
&\left[\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j^* tc}\rho_{\mathrm{j}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}})\right\|_{\mathrm{p}^*}\right)\right]^{\frac{1}{r_j^*}} \qquad (2.10c)\\
&\leq 2\left[2\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j tc}\rho_{\mathrm{j}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\right)\right]^{\frac{1}{r_j}}\left[\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j^* tc}\rho_{\mathrm{j}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\mathrm{Z_{j-1}}\right)\right]^{\frac{1}{r_j^*}} \qquad (2.10d)\\
&\leq 4^j\exp\left(\frac{\mathrm{t^2c^2}\rho_{\mathrm{j}}^2\mathrm{d}_{\mathrm{j}}^{\frac{2}{\mathrm{p}^*}}\mathrm{n(s_{j-1}+1)^2}}{2}+\mathrm{tc^j}\prod_{\mathrm{i}=1}^{\mathrm{j}}\rho_{\mathrm{i}}\mathrm{d}_{\mathrm{i}}^{\frac{1}{\mathrm{p}^*}}\mathrm{A}^{\mathrm{p}}_{\mathrm{m_1,S}}\right).
\end{aligned}
$$

The step in Equation (2.10a) follows from the observation that

$$
\mathbb{E}_{\epsilon}\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\leq c\\ f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}}\left(\boldsymbol{v}^{\mathrm{T}}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}}))\right)|\right)\leq
$$

$$
\mathbb{E}_{\epsilon}\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\leq c\\ f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}}\left(\boldsymbol{v}^{\mathrm{T}}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}}))\right)\right)+\mathbb{E}_{\epsilon}\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\leq c\\ f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}}\mathrm{d}_{\mathrm{j}}^{\frac{1}{\mathrm{p}^*}}\sum_{\mathrm{i}=1}^{\mathrm{n}}(-\epsilon_{\mathrm{i}})\sigma_{\mathrm{j}}\left(\boldsymbol{v}^{\mathrm{T}}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}}))\right)\right).
$$

The step in Equation (2.10b) follows from Lemma 9. Note that Equation (2.10c) holds for any $r_j > 1$ and $r_j^* = \frac{r_j}{r_j-1}$ by Hölder's inequality $\mathbb{E}(|XY|) \leq \mathbb{E}(|X|^{r_j})^{\frac{1}{r_j}}\mathbb{E}(|Y|^{r_j^*})^{\frac{1}{r_j^*}}$. The step in Equation (2.10d) follows from $\mathbb{E}_{\epsilon}\exp(|\mathrm{X}|) \leq \mathbb{E}_{\epsilon}\exp(\mathrm{X}) + \mathbb{E}_{\epsilon}\exp(-\mathrm{X})$.

By Lemma 5, for any $t \in \mathbb{R}$, we have

$$
\mathbb{E}_{\epsilon}\exp(\mathrm{t}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}) \leq \exp(\frac{\mathrm{t^2 n}}{2}).
$$

Then we get the desired result by choosing the optimal $r_j = s_{j-1} + 1$ while following the inductive hypothesis.

The second step is by Jensen's inequality. For any $\lambda > 0$,

$$
\begin{aligned}
n\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) &= \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}} \left(\sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}_i) \right) \right] \\
&\leq \frac{1}{\lambda} \log \mathbb{E}_\epsilon \exp \left(\lambda \sup_{\mathrm{f} \in \mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}} \left(\sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \mathrm{f}(\boldsymbol{x}_{\mathrm{i}}) \right) \right) \\
&\leq \frac{1}{\lambda} \log \mathbb{E}_\epsilon \exp \left(\lambda \mathrm{o} \sup_{\mathrm{f} \in \mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} (1, \sigma_{\mathrm{k}} \circ \mathrm{f}_{\mathrm{k}}(\boldsymbol{x}_{\mathrm{i}})) \right\|_{\mathrm{p}^*} \right) \\
&\leq \frac{1}{\lambda} \left[(k+1) \log 4 + \frac{\lambda^2 o^2 n (s_k+1)^2}{2} + \lambda o c^k \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} A^p_{m_1,S} \right], \qquad (2.11a)
\end{aligned}
$$

where the step in Equation (2.11a) is derived using a similar technique as in the first main step. By choosing the optimal $\lambda = \frac{\sqrt{(k+1)\log 16}}{o(s_k+1)\sqrt{n}}$, we have

$$
\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) \leq o \sqrt{\frac{(k+1)\log 16}{n}} (s_k+1) + \frac{1}{n} o c^k \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} A^p_{m_1,S}.
$$

■

2.6.3 Proof of Corollary 1

Proof We provide a complexity bound for the case when $p = 1$. This bound has a lower dependence on the depth under some conditions. Fixing the sample S, $q \geq 1$, the activation functions $\boldsymbol{\sigma}$, and the architecture of the DNN, define a series of random variables $\{Z_0, Z_1, \cdots, Z_k\}$ as

$$
Z_0 = \left\| \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i \right\|_\infty,
$$

and

$$
Z_j = \sup_{f \in \mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}} \left\| \sum_{i=1}^{n} \epsilon_i \sigma_j \circ f_j(\boldsymbol{x}_i) \right\|_\infty,
$$

for $j = 1, \cdots, k$, where $\{\epsilon_1, \cdots, \epsilon_n\}$ are n independent Rademacher random variables, and f_j denotes the jth hidden layer of the neural network f. Similar to the case when

$p > 1$, the proof has two main steps. In the first step, we prove by induction that for $j = 1, \cdots, k$ and any $t \in \mathbb{R}$

$$\mathbb{E}_\epsilon \exp(\mathrm{tZ_j}) \leq \sum_{i=1}^{j} 2^{j-i+2} \exp\left(\frac{\mathrm{nt^2 c^{2(j-i+1)}} \prod_{\ell=\mathrm{i}}^{\mathrm{j}} \rho_\ell^2}{2} \right) + 2^j \exp\left(\frac{\mathrm{nt^2 c^{2j}} \prod_{\mathrm{l}=1}^{\mathrm{j}} \rho_\ell^2 \max_{\mathrm{i}} \|\boldsymbol{x}_{\mathrm{i}}\|_\infty^2}{2} + \mathrm{tc^j} \prod_{\mathrm{i}=1}^{\mathrm{j}} \rho_{\mathrm{i}} \mathrm{A}^1_{\mathrm{m_1,S}} \right)$$

When $j = 0$, by Lemma 6, for any $t \in \mathbb{R}$, we have

$$\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_0}\right) \leq \exp\left(\frac{\mathrm{t^2 n} \max \|\mathbf{x}_{\mathrm{i}}\|_\infty^2}{2} + \mathrm{tA}^1_{\mathrm{m_1,S}} \right).$$

For the case when $j \geq 1$,

$$\begin{aligned}
\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_j}\right) &= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\left\|\tilde{\mathbf{V}}_j\right\|_{1,q} \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \circ (\tilde{\mathbf{V}}_{\mathrm{j}}^{\mathrm{T}}(1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right\|_\infty \right) \\
&= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_1 \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \left| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \left(\boldsymbol{v}^{\mathrm{T}}(1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \right| \right) \\
&\leq 2\mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_1 \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \left(\boldsymbol{v}^{\mathrm{T}}(1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \right) \qquad (2.12a) \\
&\leq 2\mathbb{E}_\epsilon \exp\left(\mathrm{t}\rho_{\mathrm{j}} \sup_{\substack{\|\boldsymbol{v}\|_1 \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \boldsymbol{v}^{\mathrm{T}}(1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \qquad (2.12b) \\
&\leq 2\mathbb{E}_\epsilon \exp\left(\mathrm{tc}\rho_{\mathrm{j}} \sup_{\mathrm{f} \in \mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}} \boldsymbol{d},\boldsymbol{\sigma}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right\|_\infty \right)
\end{aligned}$$

$$
\begin{aligned}
&= 2\mathbb{E}_\epsilon \exp\left(\mathrm{tc}\rho_\mathrm{j}\max\left(|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}|,\ \sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})\right\|_\infty\right)\right)\\
&= 2\mathbb{E}_\epsilon \max\left(\exp(\mathrm{tc}\rho_\mathrm{j}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}|), \exp\left(\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})\right\|_\infty\right)\right)\\
&\le 2\mathbb{E}_\epsilon \exp\left(\mathrm{tc}\rho_\mathrm{j}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}|\right) + 2\mathbb{E}_\epsilon \exp\left(\mathrm{tc}\rho_\mathrm{j}\mathrm{Z}_{\mathrm{j}-1}\right)\\
&\le 2^2\exp\left(\frac{\mathrm{t}^2\mathrm{c}^2\rho_\mathrm{j}^2\mathrm{n}}{2}\right) + 2\left[\sum_{\mathrm{i}=1}^{\mathrm{j}-1}2^{\mathrm{j}-\mathrm{i}+1}\exp\left(\frac{\mathrm{n}(\mathrm{c}\rho_\mathrm{j}\mathrm{t})^2\mathrm{c}^{2(\mathrm{j}-\mathrm{i})}\prod_{\ell=\mathrm{i}}^{\mathrm{j}-1}\rho_\ell^2}{2}\right)+\right.\\
&\left.2^{j-1}\exp\left(\frac{\mathrm{n}(\mathrm{c}\rho_\mathrm{j}\mathrm{t})^2\mathrm{c}^{2(\mathrm{j}-1)}\prod_{\mathrm{l}=1}^{\mathrm{j}-1}\rho_\ell^2\max_\mathrm{i}\|\boldsymbol{x}_\mathrm{i}\|_\infty^2}{2} + (\mathrm{tc}\rho_\mathrm{j})\mathrm{c}^{\mathrm{j}-1}\prod_{\mathrm{i}=1}^{\mathrm{j}-1}\rho_\mathrm{i}\mathrm{A}^1_{\mathrm{m}_1,\mathrm{S}}\right)\right]
\end{aligned}
\tag{2.12c}
$$

The step in Equation (2.12a) follows from the observation that

$$
\mathbb{E}_\epsilon\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\le c\\ f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}\sigma_\mathrm{j}\left(\boldsymbol{v}^\mathrm{T}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i}))\right)|\right)\le
$$

$$
\mathbb{E}_\epsilon\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\le c\\ f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}\sigma_\mathrm{j}\left(\boldsymbol{v}^\mathrm{T}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i}))\right)\right)+\mathbb{E}_\epsilon\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\le c\\ f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}}\sum_{\mathrm{i}=1}^{\mathrm{n}}(-\epsilon_\mathrm{i})\sigma_\mathrm{j}\left(\boldsymbol{v}^\mathrm{T}(1,\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i}))\right)\right).
$$

The step in Equation (2.12b) follows from Lemma 9. The step in Equation (2.12c) follows from Lemma 5 and the inductive hypothesis.

The second step is by Jensen's inequality. For any $\lambda > 0$,

$$
\begin{aligned}
n\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) &= \mathbb{E}_\epsilon\left[\sup_{f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}\left(\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}_i)\right)\right]\\
&\le \frac{1}{\lambda}\log\mathbb{E}_\epsilon\exp\left(\lambda\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left(\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}\mathrm{f}(\boldsymbol{x}_\mathrm{i})\right)\right)\\
&\le \frac{1}{\lambda}\log\mathbb{E}_\epsilon\exp\left(\lambda\mathrm{o}\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_\mathrm{i}(1,\sigma\circ\mathrm{f}_\mathrm{j}(\boldsymbol{x}_\mathrm{i}))\right\|_\infty\right)
\end{aligned}
$$

$$
\begin{aligned}
&\leq \frac{1}{\lambda}\log \mathbb{E}_{\epsilon}\left[\exp\left(\lambda \mathrm{o}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|\right)+\exp\left(\lambda \mathrm{o}\mathrm{Z}_{\mathrm{j}}\right)\right] \\
&\leq \frac{1}{\lambda}\log\left[2\exp(\frac{\lambda^2\mathrm{o}^2\mathrm{n}}{2})+\sum_{\mathrm{i}=1}^{\mathrm{k}}2^{\mathrm{k}-\mathrm{i}+2}\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2(\mathrm{k}-\mathrm{i}+1)}\prod_{\ell=\mathrm{i}}^{\mathrm{k}}\rho_\ell^2}{2}\right)+\right. \\
&\left.2^k\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2\mathrm{k}}\prod_{\mathrm{l}=1}^{\mathrm{k}}\rho_\ell^2\max_{\mathrm{i}}\|\boldsymbol{x}_{\mathrm{i}}\|_\infty^2}{2}+{}^{\mathrm{k}}\prod_{\mathrm{i}=1}^{\mathrm{k}}\rho_{\mathrm{i}}\mathrm{A}^1_{\mathrm{m}_1,\mathrm{S}}\right)\right], \qquad (2.13a)
\end{aligned}
$$

where the step in Equation (2.13a) is derived using a similar technique as in the first main step. Especially, if $c_i\rho_i \geq 1$ for $i = 1, \cdots, k$,

$$
\begin{aligned}
(2.13a) &\leq \frac{1}{\lambda}\log\left[\sum_{i=1}^{k+1}2^{k-i+2}\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2\mathrm{k}}\prod_{\mathrm{l}=1}^{\mathrm{k}}\rho_\ell^2}{2}\right)+\right. \\
&\left.2^k\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2\mathrm{k}}\prod_{\mathrm{l}=1}^{\mathrm{k}}\rho_\ell^2\max_{\mathrm{i}}\|\boldsymbol{x}_{\mathrm{i}}\|_\infty^2}{2}+\lambda\mathrm{oc}^{\mathrm{k}}\prod_{\mathrm{i}=1}^{\mathrm{k}}\rho_{\mathrm{i}}\mathrm{A}^1_{\mathrm{m}_1,\mathrm{S}}\right)\right] \\
&\leq \frac{1}{\lambda}\log\left[2^{k+2}\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2\mathrm{k}}\prod_{\mathrm{l}=1}^{\mathrm{k}}\rho_\ell^2}{2}\right)+\right. \\
&\left.2^{k+2}\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2\mathrm{k}}\prod_{\mathrm{l}=1}^{\mathrm{k}}\rho_\ell^2\max_{\mathrm{i}}\|\boldsymbol{x}_{\mathrm{i}}\|_\infty^2}{2}+\lambda\mathrm{oc}^{\mathrm{k}}\prod_{\mathrm{i}=1}^{\mathrm{k}}\rho_{\mathrm{i}}\mathrm{A}^1_{\mathrm{m}_1,\mathrm{S}}\right)\right] \\
&\leq \frac{1}{\lambda}\log\left[2^{k+3}\exp\left(\frac{\mathrm{no}^2\lambda^2\mathrm{c}^{2\mathrm{k}}\prod_{\mathrm{l}=1}^{\mathrm{k}}\rho_\ell^2\max_{\mathrm{i}}\left\|(1,\boldsymbol{x}_{\mathrm{i}}^{\mathrm{T}})\right\|_\infty^2}{2}+\lambda\mathrm{oc}^{\mathrm{k}}\prod_{\mathrm{i}=1}^{\mathrm{k}}\rho_{\mathrm{i}}\mathrm{A}^1_{\mathrm{m}_1,\mathrm{S}}\right)\right] \\
&= \frac{(k+3)\log 2}{\lambda}+\frac{no^2\lambda c^{2k}\prod_{l=1}^{k}\rho_\ell^2\max_i\left\|(1,\boldsymbol{x}_i^T)\right\|_\infty^2}{2}+oc^k\prod_{i=1}^{k}\rho_iA^1_{m_1,S}.
\end{aligned}
$$

By choosing the optimal λ, we have

$$\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq \frac{1}{\sqrt{n}} o c^k \prod_{i=1}^{k} \rho_i \max_i \left\|(1, \boldsymbol{x}_i^T)\right\|_\infty (\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}).$$

■

2.6.4 Proof of Theorem 2.4.2

Proof We first show the case when $p > 1$. The proof has two main steps.

Fixing the sample S, $p, q \geq 1$, the activation functions $\boldsymbol{\sigma}$, and the architecture of the DNN, define a series of random variables $\{Z_0, Z_1, \cdots, Z_k\}$ as

$$Z_0 = \left\| \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i \right\|_{p^*},$$

and

$$Z_j = \sup_{f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{i=1}^{n} \epsilon_i \sigma_j \circ f_j(\boldsymbol{x}_i) \right\|_{p^*}$$

for $j = 1, \cdots, k$, where $\{\epsilon_1, \cdots, \epsilon_n\}$ are n independent Rademacher random variables, and f_j denotes the jth hidden layer of the neural network f.

In the first step, we prove by induction that for $j = 1, \cdots, k$ and any $t \in \mathbb{R}$

$$\mathbb{E}_\epsilon \exp(\mathrm{tZ_j}) \leq 4 \exp\left(\frac{\mathrm{t^2 n s_j^2}}{2} + \mathrm{tc^j} \prod_{\mathrm{i=1}}^{\mathrm{j}} \rho_\mathrm{i} \mathrm{d_i^{\frac{1}{p^*}}} \mathrm{A^p_{m_1,S}} \right),$$

where $A^p_{m_1,S}$ is defined in Equation (2.9), and

$$s_j = \sum_{i=1}^{j} c^{j-i+1} \prod_{\ell=i}^{j} \rho_\ell d_\ell^{\frac{1}{p^*}} + \max_i \|\boldsymbol{x}_i\|_{p^*} c^j \prod_{l=1}^{j} \rho_\ell d_\ell^{\frac{1}{p^*}}.$$

Note that $s_{j+1} = c\rho_{j+1} d_{j+1}^{\frac{1}{p^*}} (s_j + 1)$.

When $j = 0$, by Lemma 6, for any $t \in \mathbb{R}$, we have

$$\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_0}\right) \leq \exp\left(\frac{\mathrm{t^2 n} \max \|\mathbf{x_i}\|_{\mathrm{p^*}}^2}{2} + \mathrm{tA^p_{m_1,S}} \right).$$

For the case when $j \geq 1$,

$$\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_j}\right) = \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\left\|\tilde{\mathbf{V}}_j\right\|_{p,q} \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \sigma_\mathrm{j} \circ (\tilde{\mathbf{V}}_\mathrm{j}^{\mathrm{T}}(1, \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})) \right\|_{\mathrm{p}^*} \right)$$

$$\leq \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_p \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \left| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \sigma_\mathrm{j} \left(\boldsymbol{v}^{\mathrm{T}}(1, \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})) \right) \right| \right)$$

$$= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_p \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \sigma_\mathrm{j} \left(\boldsymbol{v}^{\mathrm{T}}(1, \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})) \right) \right) \tag{2.14a}$$

$$\leq \mathbb{E}_\epsilon \exp\left(\mathrm{t}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \sup_{\substack{\|\boldsymbol{v}\|_p \leq c \\ f \in \mathcal{N}^{k,}_{p,q,c,o} \boldsymbol{d},\boldsymbol{\sigma}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \boldsymbol{v}^{\mathrm{T}}(1, \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})) \right) \tag{2.14b}$$

$$\leq \mathbb{E}_\epsilon \exp\left(\mathrm{tc}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \sup_{\mathrm{f} \in \mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}} \boldsymbol{d},\boldsymbol{\sigma}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} (1, \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i})) \right\|_{\mathrm{p}^*} \right)$$

$$\leq \mathbb{E}_\epsilon \exp\left(\mathrm{tc}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \left(|\sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i}| + \sup_{\mathrm{f} \in \mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}} \boldsymbol{d},\boldsymbol{\sigma}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i}) \right\|_{\mathrm{p}^*} \right) \right)$$

$$\leq \left[\mathbb{E}_\epsilon \exp\left(\mathrm{r_j tc}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} |\sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i}| \right) \right]^{\frac{1}{r_j}} *$$

$$\left[\mathbb{E}_\epsilon \exp\left(\mathrm{r_j^* tc}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \sup_{\mathrm{f} \in \mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}} \boldsymbol{d},\boldsymbol{\sigma}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \sigma_{\mathrm{j}-1} \circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_\mathrm{i}) \right\|_{\mathrm{p}^*} \right) \right]^{\frac{1}{r_j^*}} \tag{2.14c}$$

$$\leq \left[2\mathbb{E}_\epsilon \exp\left(\mathrm{r_j tc}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_\mathrm{i} \right) \right]^{\frac{1}{r_j}} \left[\mathbb{E}_\epsilon \exp\left(\mathrm{r_j^* tc}\rho_\mathrm{j} \mathrm{d}_\mathrm{j}^{\frac{1}{\mathrm{p}^*}} \mathrm{Z_{j-1}} \right) \right]^{\frac{1}{r_j^*}}, \tag{2.14d}$$

$$\leq 2^{\frac{1}{r_j}} 4^{\frac{1}{r_j^*}} \exp\left(\frac{\mathrm{t^2 c^2} \rho_\mathrm{j}^2 \mathrm{d}_\mathrm{j}^{\frac{2}{\mathrm{p}^*}} \mathrm{n}(\mathrm{s_{j-1}} + 1)^2}{2} + \mathrm{tc^k} \prod_{\mathrm{i}=1}^{\mathrm{k}} \rho_\mathrm{i} \mathrm{d}_\mathrm{i}^{\frac{1}{\mathrm{p}^*}} \mathrm{A}^{\mathrm{p}}_{\mathrm{m_1,S}} \right) \tag{2.14e}$$

$$\leq 4\exp\left(\frac{\mathrm{t}^2\mathrm{c}^2\rho_{\mathrm{j}}^2\mathrm{d}_{\mathrm{j}}^{\frac{2}{\mathrm{p}^*}}\mathrm{n}(\mathrm{s}_{\mathrm{j}-1}+1)^2}{2}+\mathrm{tc}^{\mathrm{k}}\prod_{\mathrm{i}=1}^{\mathrm{k}}\rho_{\mathrm{i}}\mathrm{d}_{\mathrm{i}}^{\frac{1}{\mathrm{p}^*}}\mathrm{A}_{\mathrm{m}_1,\mathrm{S}}^{\mathrm{p}}\right).$$

The step in Equation (2.14a) follows from the assumption that σ_j is anti-symmetric.

The step in Equation (2.14b) follows from Lemma 9. Note that Equation (2.14c) holds for any $r_j > 1$ and $r_j^* = \frac{r_j - 1}{r_j}$ by Hölder's inequality $\mathbb{E}(|XY|) \leq \mathbb{E}(|X|^{r_j})^{\frac{1}{r_j}}\mathbb{E}(|Y|^{r_j^*})^{\frac{1}{r_j^*}}$, and we choose $r_j = s_{j-1}+1$. The step in Equation (2.14d) follows from the inequality $\mathbb{E}_\epsilon\exp(|\mathrm{X}|) \leq \mathbb{E}_\epsilon\exp(\mathrm{X}) + \mathbb{E}_\epsilon\exp(-\mathrm{X})$. By choosing $r_j = s_{j-1}+1$, Equation (2.14e) follows from Lemma 5 and the inductive hypothesis.

The second step is based on Jensen's inequality. For any $\lambda > 0$,

$$\begin{aligned}
n\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) &= \mathbb{E}_\epsilon\left[\sup_{f\in\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}}\left(\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}_i)\right)\right] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}_\epsilon\exp\left(\lambda\sup_{\mathrm{f}\in\mathcal{N}_{\mathrm{p,q,c,o}}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}}\left(\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\mathrm{f}(\boldsymbol{x}_{\mathrm{i}})\right)\right) \\
&\leq \frac{1}{\lambda}\log\mathbb{E}_\epsilon\exp\left(\lambda\mathrm{o}\sup_{\mathrm{f}\in\mathcal{N}_{\mathrm{p,q,c,o}}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}(1,\sigma_{\mathrm{k}}\circ\mathrm{f}_{\mathrm{k}}(\boldsymbol{x}_{\mathrm{i}}))\right\|_{\mathrm{p}^*}\right) \\
&\leq \frac{1}{\lambda}\left[\log 4 + \frac{\lambda^2 o^2 n(s_k+1)^2}{2} + \lambda o c^k\prod_{i=1}^{k}\rho_i d_i^{\frac{1}{p^*}}A_{m_1,S}^p\right], \qquad (2.15a)
\end{aligned}$$

where the step in Equation (2.15a) is derived using a similar technique as in the first main step. By choosing the optimal $\lambda = \frac{\sqrt{\log 16}}{o(s_k+1)\sqrt{n}}$, we have

$$\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq o\sqrt{\frac{\log 16}{n}}(s_k+1) + \frac{1}{n}oc^k\prod_{i=1}^{k}\rho_i d_i^{\frac{1}{p^*}}A_{m_1,S}^p.$$

■

2.6.5 Proof of Corollary 2

Proof Fixing the sample S, $q \geq 1$, the activation functions $\boldsymbol{\sigma}$, and the architecture of the DNN, define a series of random variables $\{Z_0, Z_1, \cdots, Z_k\}$ as

$$Z_0 = \left\|\sum_{i=1}^{n}\epsilon_i\boldsymbol{x}_i\right\|_\infty,$$

and

$$Z_j = \sup_{f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{i=1}^{n} \epsilon_i \sigma_j \circ f_j(\boldsymbol{x}_i) \right\|_\infty$$

for $j = 1, \cdots, k$, where $\{\epsilon_1, \cdots, \epsilon_n\}$ are n independent Rademacher random variables, and f_j denotes the jth hidden layer of the neural network f. Similar to the case when $p > 1$, the proof has two main steps. In the first step, we prove by induction that for $j = 1, \cdots, k$ and any $t \in \mathbb{R}$

$$\mathbb{E}_\epsilon \exp(\mathrm{tZ_j}) \leq 2 \sum_{\mathrm{i}=1}^{\mathrm{j}} \exp\left(\frac{\mathrm{nt^2 c^{2(j-i+1)}} \prod_{\ell=\mathrm{i}}^{\mathrm{j}} \rho_\ell^2}{2} \right) + \exp\left(\frac{\mathrm{nt^2 c^{2j}} \prod_{\mathrm{l}=1}^{\mathrm{j}} \rho_\ell^2 \max_{\mathrm{i}} \|\boldsymbol{x}_{\mathrm{i}}\|_\infty^2}{2} + \mathrm{tc^j} \prod_{\mathrm{i}=1}^{\mathrm{j}} \rho_{\mathrm{i}} \mathrm{A}^1_{\mathrm{m_1,S}} \right).$$

When $k = 0$, by Lemma 6, for any $t \in \mathbb{R}$,

$$\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_0}\right) \leq \exp\left(\frac{\mathrm{t^2 n} \max \|\mathbf{x}_{\mathrm{i}}\|_\infty^2}{2} + \mathrm{tA}^1_{\mathrm{m_1,S}} \right).$$

For the case when $k \geq 1$,

$$\begin{aligned}
\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_j}\right) &= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\left\|\tilde{\mathbf{V}}_j\right\|_{1,q} \leq c \\ f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \circ (\tilde{\mathbf{V}}_{\mathrm{j}}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right\|_\infty \right) \\
&= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_1 \leq c \\ f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}}} |\sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \left(\boldsymbol{v}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right)| \right) \\
&= \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\|\boldsymbol{v}\|_1 \leq c \\ f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \left(\boldsymbol{v}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \right) \qquad (2.16a) \\
&\leq \mathbb{E}_\epsilon \exp\left(\mathrm{t}\rho_{\mathrm{j}} \sup_{\substack{\|\boldsymbol{v}\|_1 \leq c \\ f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \boldsymbol{v}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right) \qquad (2.16b)
\end{aligned}$$

$$
\begin{aligned}
&\leq \mathbb{E}_{\epsilon}\mathrm{exp}\left(\mathrm{tc}\rho_{\mathrm{j}} \sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k}}_{\mathrm{p,q,c,o}}(\boldsymbol{d},\boldsymbol{\sigma})} \left\|\sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}}(1,\sigma_{\mathrm{j}-1}\circ \mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}}))\right\|_{\infty}\right)\\
&= \mathbb{E}_{\epsilon}\mathrm{exp}\left(\mathrm{tc}\rho_{\mathrm{j}}\max\left(|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|, \sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k}}_{\mathrm{p,q,c,o}}(\boldsymbol{d},\boldsymbol{\sigma})} \left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}})\right\|_{\infty}\right)\right)\\
&= \mathbb{E}_{\epsilon}\max\left(\mathrm{exp}(\mathrm{tc}\rho_{\mathrm{j}}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|), \mathrm{exp}\left(\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k}}_{\mathrm{p,q,c,o}}(\boldsymbol{d},\boldsymbol{\sigma})} \left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}-1}\circ\mathrm{f}_{\mathrm{j}-1}(\boldsymbol{x}_{\mathrm{i}})\right\|_{\infty}\right)\right)\\
&\leq \mathbb{E}_{\epsilon}\mathrm{exp}\left(\mathrm{tc}\rho_{\mathrm{j}}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|\right) + \mathbb{E}_{\epsilon}\mathrm{exp}\left(\mathrm{tc}\rho_{\mathrm{j}}\mathrm{Z}_{\mathrm{j}-1}\right)\\
&\leq 2\mathrm{exp}\left(\frac{\mathrm{t}^2\mathrm{c}^2\rho_{\mathrm{j}}^2\mathrm{n}}{2}\right) + \sum_{\mathrm{i}=1}^{\mathrm{k}-1} 2\mathrm{exp}\left(\frac{\mathrm{n}(\mathrm{c}\rho_{\mathrm{j}}\mathrm{t})^2\mathrm{c}^{2(\mathrm{j}-\mathrm{i})}\prod_{\ell=\mathrm{i}}^{\mathrm{j}-1}\rho_{\ell}^2}{2}\right) +\\
&\mathrm{exp}\left(\frac{\mathrm{n}(\mathrm{c}\rho_{\mathrm{j}}\mathrm{t})^2\mathrm{c}^{2(\mathrm{j}-1)}\prod_{\mathrm{l}=1}^{\mathrm{j}-1}\rho_{\ell}^2\max_{\mathrm{i}}\|\boldsymbol{x}_{\mathrm{i}}\|_{\infty}^2}{2} + (\mathrm{tc}\rho_{\mathrm{j}})\mathrm{c}^{\mathrm{j}-1}\prod_{\mathrm{i}=1}^{\mathrm{j}-1}\rho_{\mathrm{i}}\mathrm{A}^1_{\mathrm{m}_1,\mathrm{S}}\right). \quad (2.16c)
\end{aligned}
$$

The step in Equation (2.16a) follows from the assumption that the activation function is anti-symmetric. The step in Equation (2.16b) follows from Lemma 9. The step in Equation (2.16c) follows from Lemma 5 and the inductive hypothesis.

The second step is based on Jensen's inequality. For any $\lambda > 0$,

$$
\begin{aligned}
n\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) &= \mathbb{E}_{\epsilon}\left[\sup_{f\in\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}}\left(\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}_i)\right)\right]\\
&\leq \frac{1}{\lambda}\log\mathbb{E}_{\epsilon}\mathrm{exp}\left(\lambda \sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left(\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\mathrm{f}(\boldsymbol{x}_{\mathrm{i}})\right)\right)\\
&\leq \frac{1}{\lambda}\log\mathbb{E}_{\epsilon}\mathrm{exp}\left(\lambda\mathrm{o}\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}_{\mathrm{p,q,c,o}}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}(1,\sigma\circ\mathrm{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{i}}))\right\|_{\infty}\right)\\
&\leq \frac{1}{\lambda}\log\mathbb{E}_{\epsilon}\left[\mathrm{exp}\left(\lambda\mathrm{o}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|\right) + \mathrm{exp}\left(\lambda\mathrm{o}\mathrm{Z}_{\mathrm{j}}\right)\right]
\end{aligned}
$$

$$\leq \frac{1}{\lambda}\log\left[2\exp(\frac{\lambda^2 o^2 n}{2}) + 2\sum_{i=1}^{k}\exp\left(\frac{no^2\lambda^2 c^{2(k-i+1)}\prod_{\ell=i}^{k}\rho_\ell^2}{2}\right) + \exp\left(\frac{no^2\lambda^2 c^{2k}\prod_{l=1}^{k}\rho_\ell^2 \max_i \|\boldsymbol{x}_i\|_\infty^2}{2} + \lambda o c^k \prod_{i=1}^{k}\rho_i A^1_{m_1,S}\right)\right], \tag{2.17a}$$

where Equation (2.17a) is obtained by applying a similar technique as that in the first main step. Especially, if $c_i\rho_i \geq 1$ for $i = 1, \cdots, k$,

$$\begin{aligned}
(2.17a) &\leq \frac{1}{\lambda}\log\left[2\sum_{i=1}^{k+1}\exp\left(\frac{no^2\lambda^2 c^{2k}\prod_{l=1}^{k}\rho_\ell^2}{2}\right) + \exp\left(\frac{no^2\lambda^2 c^{2k}\prod_{l=1}^{k}\rho_\ell^2 \max_i \|\boldsymbol{x}_i\|_\infty^2}{2} + \lambda o c^k \prod_{i=1}^{k}\rho_i A^1_{m_1,S}\right)\right] \\
&\leq \frac{1}{\lambda}\log\left[2(k+1)\exp\left(\frac{no^2\lambda^2 c^{2k}\prod_{l=1}^{k}\rho_\ell^2}{2}\right) + \exp\left(\frac{no^2\lambda^2 c^{2k}\prod_{l=1}^{k}\rho_\ell^2 \max_i \|\boldsymbol{x}_i\|_\infty^2}{2} + \lambda o c^k \prod_{i=1}^{k}\rho_i A^1_{m_1,S}\right)\right] \\
&\leq \frac{1}{\lambda}\log\left[(2k+3)\exp\left(\frac{no^2\lambda^2 c^{2k}\prod_{l=1}^{k}\rho_\ell^2 \max_i \left\|(1,\boldsymbol{x}_i^{\mathrm{T}})\right\|_\infty^2}{2} + \lambda o c^k \prod_{i=1}^{k}\rho_i A^1_{m_1,S}\right)\right] \\
&= \frac{\log(2k+3)}{\lambda} + \frac{no^2\lambda c^{2k}\prod_{l=1}^{k}\rho_\ell^2 \max_i \left\|(1,\boldsymbol{x}_i^{T})\right\|_\infty^2}{2} + oc^k\prod_{i=1}^{k}\rho_i A^1_{m_1,S}.
\end{aligned}$$

By choosing the optimal λ, we have

$$\widehat{\mathfrak{R}}_S(\mathcal{N}^{k,\boldsymbol{d},\boldsymbol{\sigma}}_{p,q,c,o}) \leq \frac{1}{\sqrt{n}} oc^k \prod_{i=1}^{k}\rho_i \max_i \left\|(1,\boldsymbol{x}_i^T)\right\|_\infty (\sqrt{2\log(2k+3)} + \sqrt{2\log(2m_1)}).$$

■

2.6.6 Theorem 2.4.3

Proof We first show the case when $p > 1$. The proof has two main steps.

Fixing the sample S, $p, q \geq 1$, the activation functions $\boldsymbol{\sigma}$, and the architecture of the DNN, define a series of random variables $\{Z_0, Z_1, \cdots, Z_k\}$ as

$$Z_0 = \left\| \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i \right\|_{p^*},$$

and

$$Z_j = \sup_{f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{i=1}^{n} \epsilon_i \sigma_j \circ f_j(\boldsymbol{x}_i) \right\|_{p^*},$$

for $j = 1, \cdots, k$, where $\{\epsilon_1, \cdots, \epsilon_n\}$ are n independent Rademacher random variables, and f_j denotes the jth hidden layer of the neural network f.

In the first step, we prove by induction that for $j = 1, \cdots, k$ and any $t \in \mathbb{R}$

$$\mathbb{E}_\epsilon \exp(\mathrm{tZ_j}) \leq 4^{\mathrm{j}} \exp\left(\frac{\mathrm{t^2 n s_j^2}}{2} + \mathrm{tc^j} \prod_{\mathrm{i=1}}^{\mathrm{j}} \rho_{\mathrm{i}} \mathrm{d_i^{[\frac{1}{p^*} - \frac{1}{q}]_+}} \mathrm{A^p_{m_1,S}} \right),$$

where $A^p_{m_1,S}$ is defined in Equation (2.9), and

$$s_j = \sum_{i=1}^{j} c^{j-i+1} \prod_{\ell=i}^{j} \rho_\ell d_\ell^{[\frac{1}{p^*} - \frac{1}{q}]_+} + \max_i \|\boldsymbol{x}_i\|_{p^*} c^j \prod_{l=1}^{j} \rho_\ell d_\ell^{[\frac{1}{p^*} - \frac{1}{q}]_+}.$$

Note that $s_{j+1} = c\rho_{j+1} d_{j+1}^{[\frac{1}{p^*} - \frac{1}{q}]_+} (s_j + 1)$.

When $k = 0$, by Lemma 6, for any $t \in \mathbb{R}$,

$$\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_0}\right) \leq \exp\left(\frac{\mathrm{t^2 n} \max \|\mathbf{x_i}\|_{\mathrm{p^*}}^2}{2} + \mathrm{tA^p_{m_1,S}} \right).$$

For the case when $j \geq 1$,

$$\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_j}\right) = \mathbb{E}_\epsilon \exp\left(\mathrm{t} \sup_{\substack{\left\|\tilde{\mathbf{V}}_j\right\|_{p,q} \leq c \\ f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}}} \left\| \sum_{\mathrm{i=1}}^{\mathrm{n}} \epsilon_{\mathrm{i}} \sigma_{\mathrm{j}} \circ (\tilde{\mathbf{V}}_{\mathrm{j}}^{\mathrm{T}} (1, \sigma_{\mathrm{j-1}} \circ \mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})) \right\|_{\mathrm{p^*}} \right)$$

$$\leq \mathbb{E}_{\epsilon}\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\leq c\\ f\in\mathcal{N}^{k,}_{p,q,c,o}\boldsymbol{d},\boldsymbol{\sigma}}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}}\left(\boldsymbol{v}^{\mathrm{T}}(1,\sigma_{\mathrm{j-1}}\circ\mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}}))\right)|\right) \tag{2.18a}$$

$$\leq 2\mathbb{E}_{\epsilon}\exp\left(\mathrm{t}\sup_{\substack{\|\boldsymbol{v}\|_p\leq c\\ f\in\mathcal{N}^{k,}_{p,q,c,o}\boldsymbol{d},\boldsymbol{\sigma}}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j}}\left(\boldsymbol{v}^{\mathrm{T}}(1,\sigma_{\mathrm{j-1}}\circ\mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}}))\right)\right)$$

$$\leq 2\mathbb{E}_{\epsilon}\exp\left(\mathrm{t}\rho_{\mathrm{j}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\sup_{\substack{\|\boldsymbol{v}\|_p\leq c\\ f\in\mathcal{N}^{k,}_{p,q,c,o}\boldsymbol{d},\boldsymbol{\sigma}}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\boldsymbol{v}^{\mathrm{T}}(1,\sigma_{\mathrm{j-1}}\circ\mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}}))\right) \tag{2.18b}$$

$$\leq 2\mathbb{E}_{\epsilon}\exp\left(\mathrm{tc}\rho_{\mathrm{j}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}}\boldsymbol{d},\boldsymbol{\sigma}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}(1,\sigma_{\mathrm{j-1}}\circ\mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}}))\right\|_{\mathrm{p}^*}\right)$$

$$\leq 2\mathbb{E}_{\epsilon}\exp\left(\mathrm{tc}\rho_{\mathrm{j}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\left(|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|+\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}}\boldsymbol{d},\boldsymbol{\sigma}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j-1}}\circ\mathrm{f_{-1}}(\boldsymbol{x}_{\mathrm{i}})\right\|_{\mathrm{p}^*}\right)\right)$$

$$\leq 2\left[\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j tc}\rho_{\mathrm{j}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}|\right)\right]^{\frac{1}{r_j}}*$$

$$\left[\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j^* tc}\rho_{\mathrm{j}}\mathrm{d_j}^{\frac{1}{\mathrm{p}^*}}\sup_{\mathrm{f}\in\mathcal{N}^{\mathrm{k},}_{\mathrm{p,q,c,o}}\boldsymbol{d},\boldsymbol{\sigma}}\left\|\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\sigma_{\mathrm{j-1}}\circ\mathrm{f_{j-1}}(\boldsymbol{x}_{\mathrm{i}})\right\|_{\mathrm{p}^*}\right)\right]^{\frac{1}{r_j^*}} \tag{2.18c}$$

$$\leq 2\left[2\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j tc}\rho_{\mathrm{j}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\sum_{\mathrm{i}=1}^{\mathrm{n}}\epsilon_{\mathrm{i}}\right)\right]^{\frac{1}{r_j}}\left[\mathbb{E}_{\epsilon}\exp\left(\mathrm{r_j^* tc}\rho_{\mathrm{j}}\mathrm{d_j}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\mathrm{Z_{j-1}}\right)\right]^{\frac{1}{r_j^*}} \tag{2.18d}$$

$$\leq 4^j\exp\left(\frac{\mathrm{t^2c^2}\rho_{\mathrm{j}}^2\mathrm{d_j}^{2[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\mathrm{n(s_{j-1}+1)^2}}{2}+\mathrm{tc^j}\prod_{\mathrm{i}=1}^{\mathrm{j}}\rho_{\mathrm{i}}\mathrm{d_i}^{[\frac{1}{\mathrm{p}^*}-\frac{1}{\mathrm{q}}]_+}\mathrm{A}^{\mathrm{p}}_{\mathrm{m_1,S}}\right).$$

The step in Equation (2.18a) follows from Lemma 7. The step in Equation (2.18b) follows from Lemma 9. Note that Equation (2.18c) holds for any $r_j > 1$ and $r_j^* = \frac{r_j}{r_j-1}$ by Hölder's inequality $\mathbb{E}(|XY|) \leq \mathbb{E}(|X|^{r_j})^{\frac{1}{r_j}}\mathbb{E}(|Y|^{r_j^*})^{\frac{1}{r_j^*}}$. The step in Equation (2.18d)

follows from $\mathbb{E}_\epsilon \exp(|\mathrm{X}|) \leq \mathbb{E}_\epsilon \exp(\mathrm{X}) + \mathbb{E}_\epsilon \exp(-\mathrm{X})$. By Lemma 5, for any $t \in \mathbb{R}$, we have

$$\mathbb{E}_\epsilon \exp(\mathrm{t} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}}) \leq \exp(\frac{\mathrm{t}^2 \mathrm{n}}{2}).$$

Then we get the desired result by choosing the optimal $r_j = s_{j-1} + 1$, while following the inductive hypothesis.

The second step is based on Jensen's inequality. For any $\lambda > 0$,

$$\begin{aligned}
n\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) &= \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}} \left(\sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}_i) \right) \right] \\
&\leq \frac{1}{\lambda} \log \mathbb{E}_\epsilon \exp \left(\lambda \sup_{\mathrm{f} \in \mathcal{N}_{\mathrm{p,q,c,o}}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}} \left(\sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \mathrm{f}(\boldsymbol{x}_{\mathrm{i}}) \right) \right) \\
&\leq \frac{1}{\lambda} \log \mathbb{E}_\epsilon \exp \left(\lambda \mathrm{o} \sup_{\mathrm{f} \in \mathcal{N}_{\mathrm{p,q,c,o}}^{\mathrm{k},\boldsymbol{d},\boldsymbol{\sigma}}} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} (1, \sigma \circ \mathrm{f_k}(\boldsymbol{x}_{\mathrm{i}})) \right\|_{\mathrm{p}^*} \right) \\
&\leq \frac{1}{\lambda} \left[(k+1) \log 4 + \frac{\lambda^2 o^2 n (s_k+1)^2}{2} + \lambda o c^k \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} A_{m_1,S}^p \right] \qquad (2.19a)
\end{aligned}$$

where the step in Equation (2.19a) is derived using a similar technique as in the first main step. By choosing the optimal $\lambda = \frac{\sqrt{(k+1)\log 16}}{o(s_k+1)\sqrt{n}}$, we have

$$\widehat{\mathfrak{R}}_S(\mathcal{N}_{p,q,c,o}^{k,\boldsymbol{d},\boldsymbol{\sigma}}) \leq o \sqrt{\frac{(k+1)\log 16}{n}} (s_k+1) + \frac{1}{n} o c^k \prod_{i=1}^{k} \rho_i d_i^{\frac{1}{p^*}} A_{m_1,S}^p$$

■

2.6.7 Proof of Lemma 1

Proof It is sufficient to prove the case when $J = 2^k$. If $J < 2^k$, we could add $2^k - J$ duplicate $\boldsymbol{a}_1^T \mathbf{u} + b_1$'s into the max function. By Lemma 8, there exists a neural network such that $g = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \circ \mathbf{u}$, $T_1(\boldsymbol{v}) : \mathbb{R}^s \to \mathbb{R}^{2^{k+1}}$, $T_i : \mathbb{R}^{2^{k+3-i}} \to \mathbb{R}^{2^{k+2-i}}$ for $i = 2, \cdots, k$, $T_{k+1} : \mathbb{R}^4 \to \mathbb{R}$, $\|T_1\|_{p,q} \leq L 2^{1+\frac{k+1}{q}}$, $\|T_i\|_{p,q} = 2^{\frac{3}{p}+\frac{k+2-i}{q}-1}$ for $i = 2, \cdots, k$, and $\|T_{k+1}\|_p = 2^{\frac{2}{p}-1}$. Thus

$$\|T_{k+1}\|_p \prod_{i=1}^{k} \|T_i\|_{p,q} \leq L2^{[1+\frac{k+1}{q}+\sum_{i=2}^{k}(\frac{3}{p}+\frac{k+2-i}{q}-1)+\frac{2}{p}-1]} \leq 2L2^{\left(\frac{3}{p}+\frac{k+3}{2q}-1\right)k}.$$

Define the normalization constant $c = 2^{\frac{3}{p}+\frac{k+3}{2q}-1}$ and $o = 2L$. Normalize T_i by $cT_i/\|T_i\|_{p,q}$ for $i = 1, 2, \cdots, k$ and T_{k+1} by $T_{k+1}\prod_{i=1}^{k}\|T_i\|_{p,q}/c^k$. As ReLU is positive homogeneous, the normalized DNN still represents the same function as the original one. Thus we complete the proof. ■

2.6.8 Proof of Lemma 2

Proof For simplicty, we assume $\sigma = \sigma_{\text{ReLU}}$ in the remainder of the proof.

$\left\|(b_i, \boldsymbol{w}_i^T)\right\|_1 = 1$ implies $\left\|(b_i, 2\boldsymbol{w}_i^T)\right\|_1 \leq 2$, thus by Theorem 2.3.1 Part (a), it is sufficient to show that g could be represented by a neural network in $\mathcal{N}_{p,q,wid_k^{1/q},o}^{k,\boldsymbol{d}^k,\sigma_{\text{ReLU}}}$ if instead $\left\|(b_i, 2\boldsymbol{w}_i^T)\right\|_1 = 1$. In addition, by Theorem 2.3.1 Parts (a), (b) and (c), it is equivalent to show that when $\sum_{i=1}^{r}|c_i| = 1$, g could be represented by some neural network in $\mathcal{N}_{1,\infty,1,1}^{k,\boldsymbol{d},\sigma_{\text{ReLU}}}$ where $d_i \leq [r/k] + 2m_1 + 3$ for $i = 1, \cdots, k$.

Decompose the shallow neural network as

$$g(\boldsymbol{x}) = \left(\sum_{i=1}^{r_1} c_i^+\right) g_+(\boldsymbol{x}) - \left(\sum_{i=1}^{r_2} c_i^-\right) g_-(\boldsymbol{x}),$$

where

$$g_+(\boldsymbol{x}) = \sum_{i=1}^{r_1} c_i^+ \sigma\left((\boldsymbol{w}_i^+)^T\boldsymbol{x} + b_i^+\right) / \sum_{i=1}^{r_1} c_i^+, \quad g_-(\boldsymbol{x}) = \sum_{i=1}^{r_2} c_i^- \sigma\left((\boldsymbol{w}_i^-)^T\boldsymbol{x} + b_i^-\right) / \sum_{i=1}^{r_2} c_i^-,$$

for some $c_i^+, c_i^- > 0$. Note that $\left\|\alpha^T A^T\right\|_1 \leq 1$ if $\alpha \in \mathbb{R}^s$ satisfies that $\|\alpha\|_1 \leq 1$, and $A \in \mathbb{R}^{t\times s}$ satisfies that $\|A\|_{1,\infty} \leq 1$. Additionally

$$\sum_{i=1}^{r_1} c_i^+ + \sum_{i=1}^{r_2} c_i^- = \sum_{i=1}^{r} |c_i| = 1.$$

Thus it is sufficient to show that

$$(g_+(\boldsymbol{x}), g_-(\boldsymbol{x}))$$

could be represented by some neural network in $\mathcal{N}_{1,\infty,1,1}^{k,\boldsymbol{d},\sigma_{\mathrm{ReLU}}}$, where each hidden layer contains both $\sigma \circ \boldsymbol{x}$ and $\sigma \circ (-\boldsymbol{x})$, while satisfying that $d_i \leq [r_1/k] + [r_2/k] + 2m_1 + 2$ for $i = 1, \cdots, k$ and $d_{k+1} = 2$.

When $k = 1$, it is trivial.

When $k = 2$, we construct the first hidden layer consisting of $[r_1/2] + [r_2/2] + 2m_1$ hidden neurons:

$$\{(\boldsymbol{w}_i^+)^T\boldsymbol{x} + b_i^+ : i = 1, \cdots, [r_1/2]\}, \{(\boldsymbol{w}_i^-)^T\boldsymbol{x} + b_i^- : i = 1, \cdots, [r_2/2]\}, \boldsymbol{x}, -\boldsymbol{x}.$$

For the second hidden layer, there are $2 + r - ([r_1/2] + [r_2/2]) + 2m_1$ hidden neurons. The first neuron

$$\eta_1 = \sum_{i=1}^{[r_1/2]} c_i^+ \sigma\left((\boldsymbol{w}_i^+)^T\boldsymbol{x} + b_i^+\right) / \sum_{i=1}^{[r_1/2]} c_i^+,$$

the second neuron

$$\eta_2 = \sum_{i=1}^{[r_2/2]} c_i^- \sigma\left((\boldsymbol{w}_i^-)^T\boldsymbol{x} + b_i^-\right) / \sum_{i=1}^{[r_2/2]} c_i^-,$$

then follows $\sigma \circ \boldsymbol{x}$, $\sigma \circ (-\boldsymbol{x})$ and the left $r - ([r_1/2] + [r_2/2])$ hidden neurons

$$\{\eta_i^+ = (\boldsymbol{w}_i^+)^T\sigma \circ \boldsymbol{x} - (\boldsymbol{w}_i^+)^T\sigma \circ (-\boldsymbol{x}) + b_i^+ : i = [r_1/2] + 1, \cdots, r_1\},$$

$$\{\eta_i^- = (\boldsymbol{w}_i^-)^T\sigma \circ \boldsymbol{x} - (\boldsymbol{w}_i^-)^T\sigma \circ (-\boldsymbol{x}) + b_i^- : i = [r_2/2] + 1, \cdots, r_2\}.$$

The output layer only contains two hidden neurons (g_+, g_-), which could be computed respectively by

$$\frac{\sum_{i=1}^{[r_1/2]} c_i^+}{\sum_{i=1}^{r_1} c_i^+}\sigma(\eta_1) + \sum_{i=[r_1/2]+1}^{r_1} \frac{c_i^+}{\sum_{i=1}^{r_1} c_i^+}\sigma(\eta_i^+) \quad \text{and} \quad \frac{\sum_{i=1}^{[r_2/2]} c_i^-}{\sum_{i=1}^{r_2} c_i^-}\sigma(\eta_2) + \sum_{i=[r_2/2]+1}^{r_2} \frac{c_i^-}{\sum_{i=1}^{r_2} c_i^-}\sigma(\eta_i^-).$$

Thus, we find a neural network in $\mathcal{N}_{1,\infty,1,o}^{2,\boldsymbol{d},\sigma_{\mathrm{ReLU}}}$ representing (g_+, g_-), where $d_i \leq [r_1/2] + [r_2/2] + 2m_1 + 2$.

When $k = K$, define $r_1^* = (K-1)[r_1/K], r_2^* = (K-1)[r_2/K], r^* = r_1 + r_2$ and

$$\begin{aligned} g^*(\boldsymbol{x}) &= (g_+^*(\boldsymbol{x}), g_-^*(\boldsymbol{x})) \\ &= \left(\frac{1}{\sum_{i=1}^{r_1^*} c_i^+} \sum_{i=1}^{r_1^*} c_i^+ \sigma\left((\boldsymbol{w}_i^+)^T \boldsymbol{x} + b_i^+\right), \frac{1}{\sum_{i=1}^{r_2^*} c_i^-} \sum_{i=1}^{r_2^*} c_i^- \sigma\left((\boldsymbol{w}_i^-)^T \boldsymbol{x} + b_i^-\right) \right). \end{aligned}$$

By induction assumption, g^* could be represented $h^* \in \mathcal{N}_{1,\infty,1,1}^{K-1,\boldsymbol{d}^*,\sigma_{\mathrm{ReLU}}}$, where $d_i^* \leq [r_1^*/(K-1)] + [r_2^*/(K-1)] + 2m_1 + 2$. In order to construct a WN-DNN representing (g_+, g_-), we keep the first $K-1$ hidden layers of h^* and build the Kth hidden layer based on the output layer of h^*. Since the $(K-1)$th hidden layer contains both $\sigma \circ \boldsymbol{x}$ and $\sigma \circ (-\boldsymbol{x})$. Thus except the original two neurons, we could add

$$\{(\boldsymbol{w}_i^+)^T(\sigma \circ \boldsymbol{x} - \sigma \circ (-\boldsymbol{x})) + b_i^+ : i = r_1^* + 1, \cdots, r_1\},$$

$$\{(\boldsymbol{w}_i^-)^T(\sigma \circ \boldsymbol{x} - \sigma \circ (-\boldsymbol{x})) + b_i^- : i = r_2^* + 1, \cdots, r_2\}, \sigma \circ \boldsymbol{x}), \sigma \circ (-\boldsymbol{x})$$

to the Kth hidden layer. Note that $\left\|(b_i, 2\boldsymbol{w}_i^T)\right\|_1 = 1$, thus we does not increase the $L_{1,\infty}$ norm of the Kth transformation by adding these neurons.

We finally construct the output layer by

$$\begin{aligned} &\frac{\sum_{i=1}^{r_1^*} c_i^+}{\sum_{i=1}^{r_1} c_i^+} \sigma(g_+^*(\boldsymbol{x})) + \sum_{i=r_1^*+1}^{r_1} \frac{c_i^+}{\sum_{i=1}^{r_1} c_i^+} \sigma\left((\boldsymbol{w}_i^+)^T \boldsymbol{x} + b_i^+\right), \\ &\frac{\sum_{i=1}^{r_2^*} c_i^-}{\sum_{i=1}^{r_2} c_i^-} \sigma(g_-^*(\boldsymbol{x})) + \sum_{i=r_2^*+1}^{r_2} \frac{c_i^-}{\sum_{i=1}^{r_2} c_i^-} \sigma\left((\boldsymbol{w}_i^-)^T \boldsymbol{x} + b_i^-\right). \end{aligned}$$

Thus, we build a neural network in $\mathcal{N}_{1,\infty,1,1}^{K,\boldsymbol{d},\sigma_{\mathrm{ReLU}}}$ representing (g_+, g_-). The width of the ith hidden layer $d_i \leq [r_1/K] + [r_2/K] + 2m_1 + 3$. ■

2.6.9 Proof for Theorem 2.5.2

Proof Assume f is an arbitrary function defined on $\mathbb{R}^{m_1} \to \mathbb{R}$, satisfying that $\|\boldsymbol{x}_1\|_\infty \leq 1$, $\|\boldsymbol{x}_2\|_\infty \leq 1$, $f(\boldsymbol{x}_1) \leq L$ and $|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_\infty$. Following

[39, Propositions 1 & 6], for o greater than a constant depending only on m_1, a fixed $\gamma > 0$, , there exists some function $h(\boldsymbol{x}) : \mathbb{R}^{m_1} \to \mathbb{R} = \sum_{i=1}^{r} c_i \sigma(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)$, satisfying that $\sum_{i=1}^{r} |c_i| \leq o$, $\left\|(b_i, \boldsymbol{w}_i^T)\right\|_1 = 1$ and $r \leq c_2(m_1)\gamma^{-\frac{2(m_1+1)}{m_1+4}}$, such that

$$\sup_{\|\boldsymbol{x}\|_\infty \leq 1} |f(\boldsymbol{x}) - h(\boldsymbol{x})| \leq o\gamma + c_1(m_1) L (\frac{o}{L})^{-\frac{2}{m_1+1}} \log \frac{o}{L},$$

where $c_1(m_1)$ and $c_2(m_1)$ are some constants depending only on m_1.

By taking $\gamma = c_1(m_1)(o/L)^{-1-2/(m_1+1)} \log \frac{o}{L}$, we have some function $h(\boldsymbol{x}) = \sum_{i=1}^{r} c_i \sigma(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)$, satisfying that $\sum_{i=1}^{r} |c_i| \leq o$, $\left\|(b_i, 2\boldsymbol{w}_i^T)\right\|_1 = 1$ and

$$r \leq C_r(m_1)(\log \frac{o}{L})^{-2(m_1+1)/(m_1+4)} \left(\frac{o}{L}\right)^{2(m_1+3)/(m_1+4)},$$

such that

$$\sup_{\|\boldsymbol{x}\|_\infty \leq 1} |f(\boldsymbol{x}) - h(\boldsymbol{x})| \leq C(m_1) L (\frac{o}{L})^{-\frac{2}{m_1+1}} \log \frac{o}{L},$$

where $C_r(m_1)$ and $C(m_1)$ denote some constants that depend only on m_1.

By Lemma 2, for any integer $k \in [1, r]$, this h could be represented by a neural network in $\mathcal{N}_{p,\infty,1,o}^{k,\boldsymbol{d}^k}$, where $\boldsymbol{d}_0^k = m_1$, $\boldsymbol{d}_i^k = [r/k] + 2m_1 + 3$ for $i = 1, \cdots, k$ and $\boldsymbol{d}_{k+1}^k = 1$. ■

2.7 Technical Lemmas

In this appendix, we provide technical lemmas and their proofs in order to develop the main results.

Lemma 3 (Massart's finite lemma) *Let $\mathcal{A}$ be some finite subset of $\mathbb{R}^m$ and $\epsilon_1.\epsilon_2, \cdots, \epsilon_m$ be independent Radermacher random variables. Let $r = \sup_{\boldsymbol{a} \in \mathcal{A}} \|\boldsymbol{a}\|_2$, then we have*

$$\mathbb{E}\left[\sup_{\boldsymbol{a} \in \mathcal{A}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i a_i\right] = \frac{r\sqrt{2 \log |\mathcal{A}|}}{m}.$$

Lemma 4 *[40] Assume that the hypothesis class $\mathcal{F} \subseteq \{f|f : \mathcal{X} \to \mathbb{R}\}$, and $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{X}$. Let $G : \mathbb{R} \to \mathbb{R}$ be convex and increasing. Assume that the function $\phi : \mathbb{R} \to \mathbb{R}$ is L-Lipschitz continuous, and satisfies that $\phi(0) = 0$. We have:*

$$\mathbb{E}_\epsilon \left[G\left(\sup_{f\in\mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i \phi(f(\boldsymbol{x}_i)) \right) \right) \right] \leq \mathbb{E}_\epsilon \left[G\left(L \sup_{f\in\mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}_i) \right) \right) \right].$$

Lemma 5 *Assume that $\epsilon_1, \epsilon_2. \cdots, \epsilon_n$ are n independent Rademacher random variables. Then for any $t \in \mathbb{R}$,*

$$\mathbb{E}_\epsilon \exp(\mathrm{t} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}}) \leq \exp(\frac{\mathrm{t}^2\mathrm{n}}{2}).$$

Proof Note that $\sum_{i=1}^{n} \epsilon_i$ is also a deterministic function of the i.i.d.random variables $\epsilon_1, \cdots, \epsilon_n$, satisfying that $\mathbb{E}_\epsilon \sum_{i=1}^{n} \epsilon_i = 0$ and

$$|\sum_{i\neq j} \epsilon_i + \epsilon_j - (\sum_{i\neq j} \epsilon_i - \epsilon_j)| \leq 2.$$

Then by the proof of Theorem 6.2 [41],

$$\mathbb{E}_\epsilon \exp(\mathrm{t} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}}) \leq \exp(\frac{\mathrm{t}^2\mathrm{n}}{2}),$$

for any $t \in \mathbb{R}$. ■

Lemma 6 *Assume that $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^{m_1}$ for $i = 1, 2, \cdots, n$. For any $p \in [1, \infty)$, $1/p + 1/p^* = 1$, and any $t \in \mathbb{R}$, we have*

$$\mathbb{E}_\epsilon \exp\left(\mathrm{t} \left\| \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}} \boldsymbol{x}_{\mathrm{i}} \right\|_{\mathrm{p}^*} \right) \leq \exp\left(\frac{\mathrm{t}^2\mathrm{n} \max \|\mathbf{x}_{\mathrm{i}}\|_{\mathrm{p}^*}^2}{2} + \mathrm{t}\mathrm{A}^{\mathrm{p}}_{\mathrm{m}_1,\mathrm{S}} \right),$$

where $A^p_{m_1,S}$ is defined in Equation (2.9).

Proof Define

$$Z_0 = \left\| \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i \right\|_{p^*}.$$

We first show that $\mathbb{E}_\epsilon Z_0 \leq A^p_{m_1,S}$. For $p \in (1,2]$, $\|\cdot\|_{p^*}$ is $2(p^*-1)$-strongly convex with respect to itself on $\mathbb{R}^{m_1}$ [42], thus $\frac{1}{n}\mathbb{E}_\epsilon \left\|\sum_{i=1}^n \epsilon_i \boldsymbol{x}_i\right\|_{p^*} \leq \sqrt{\frac{p^*-1}{n}} \max_i \|\boldsymbol{x}_i\|_{p^*}$ [43].

For $p \in [1,\infty]$, let $\boldsymbol{x}[j] = (\boldsymbol{x}_1[j], \boldsymbol{x}_2[j], \cdots, \boldsymbol{x}_n[j])^T$, where $\boldsymbol{x}_i[j]$ is the jth element of the vector $\boldsymbol{x}_i \in \mathbb{R}^{m_1}$.

$$\begin{aligned}
\frac{1}{n}\mathbb{E}_\epsilon \left\|\sum_{i=1}^n \epsilon_i \boldsymbol{x}_i\right\|_{p^*} &\leq \frac{m_1^{\frac{1}{p^*}}}{n}\mathbb{E}\left\|\sum_{i=1}^n \epsilon_i \boldsymbol{x}_i\right\|_\infty \\
&\leq \frac{m_1^{\frac{1}{p^*}}\sqrt{2\log(2m_1)}}{n}\sup_j \|\mathbf{z}[j]\|_2 \qquad (2.20)\\
&\leq \frac{m_1^{\frac{1}{p^*}}\sqrt{2\log(2m_1)}}{n}\sqrt{n}\sup_j \|\boldsymbol{x}[j]\|_\infty \\
&\leq \frac{m_1^{\frac{1}{p^*}}\sqrt{2\log(2m_1)}}{\sqrt{n}}\max_i \|\boldsymbol{x}_i\|_\infty
\end{aligned}$$

The step in Equation (2.20) follows from Massart's finite lemma.

Note that Z_0 is a deterministic function of the i.i.d.random variables $\epsilon_1, \cdots, \epsilon_n$, and satisfies that

$$|Z_0(\epsilon_1, \cdots, \epsilon_i, \cdots, \epsilon_n) - Z_0(\epsilon_1, \cdots, -\epsilon_i, \cdots, \epsilon_n)| \leq 2\max \|\mathbf{x}_i\|_{p^*},$$

by Minkowski inequality. By the proof of Theorem 6.2 [41], Z_0 satisfies that

$$\begin{aligned}
\mathbb{E}_\epsilon \exp\left(\mathrm{tZ_0}\right) &= \mathbb{E}_\epsilon \exp\left(\mathrm{t(Z_0 - E_\epsilon Z_0)}\right) * \exp\left(\mathrm{tE_\epsilon Z_0}\right) \\
&\leq \exp\left(\frac{\mathrm{t^2 n \max \|x_i\|_{p^*}^2}}{2} + \mathrm{tA^p_{m_1,S}}\right)
\end{aligned}$$

for any $t \in \mathbb{R}$. ■

Lemma 7 *Assume that σ is positive homogeneous. $\forall p, q \geq 1$, $s_1, s_2 \geq 1$, $\{\epsilon_1, \cdots, \epsilon_n\} \subseteq \{-1,+1\}^n$, and for all functions $g : \mathbb{R}^{m_1} \to \mathbb{R}^{s_1}$, we have*

$$\sup_{\mathbf{V}\in\mathbb{R}^{s_1\times s_2}} \frac{1}{\|\mathbf{V}\|_{p,q}} \left\|\sum_{i=1}^n \epsilon_i \sigma\left(\mathbf{V}^T g(\boldsymbol{x}_i)\right)\right\|_{p^*} = s_2^{[\frac{1}{p^*}-\frac{1}{q}]_+} \sup_{\boldsymbol{v}\in\mathbb{R}^{s_1}} \frac{1}{\|\boldsymbol{v}\|_p}\left|\sum_{i=1}^n \epsilon_i \sigma\left(\langle \boldsymbol{v}, g(\boldsymbol{x}_i)\rangle\right)\right|,$$

where $\frac{1}{p} + \frac{1}{p^*} = 1$.

Proof The proof is based on the ideas of [21, Lemma 17]

The right hand side (RHS) is always less than or equal to the left hand side (LHS), since given any vector $\boldsymbol{v}$ we could create a corresponding matrix $\mathbf{V}$ of which each row is $\boldsymbol{v}$.

Then we will show that (LHS) is always less than or equal to (RHS). Let $\mathbf{V}[,j]$ be the jth column of the matrix $\mathbf{V}$. We have $\|\mathbf{V}\|_{p,p^*} \leq \|\mathbf{V}\|_{p,q}$ when $q \leq p^*$, and by Hölder's inequality, $\|\mathbf{V}\|_{p,p^*} \leq s_2^{[\frac{1}{p^*}-\frac{1}{q}]} \|\mathbf{V}\|_{p,q}$ when $q > p^*$. Thus

$$
\begin{aligned}
\text{(LHS)} &\leq \sup_{\mathbf{V}\in\mathbb{R}^{s_1\times s_2}} \frac{s_2^{[\frac{1}{p^*}-\frac{1}{q}]_+}}{\|\mathbf{V}\|_{p,p^*}} \left\| \sum_{i=1}^{n} \epsilon_i \sigma \circ \left(\mathbf{V}^T g(\boldsymbol{x}_i)\right) \right\|_{p^*} \\
&= s_2^{[\frac{1}{p^*}-\frac{1}{q}]_+} \sup_{\mathbf{V}\in\mathbb{R}^{s_1\times s_2}} \frac{1}{\|\mathbf{V}\|_{p,p^*}} \left(\sum_{j=1}^{s_2} \left| \sum_{i=1}^{n} \epsilon_i \sigma \left(\langle \mathbf{V}[,j], g(\boldsymbol{x}_i) \rangle \right) \right|^{p^*} \right)^{1/p^*} \\
&\leq s_2^{[\frac{1}{p^*}-\frac{1}{q}]_+} \sup_{\mathbf{V}\in\mathbb{R}^{s_1\times s_2}} \frac{1}{\|\mathbf{V}\|_{p,p^*}} \left(\sum_{j=1}^{s_2} \left(\|\mathbf{V}[,j]\|_p \frac{\text{(RHS)}}{s_2^{[\frac{1}{p^*}-\frac{1}{q}]_+}} \right)^{p^*} \right)^{1/p^*} \\
&= \text{(RHS)} \sup_{\mathbf{V}\in\mathbb{R}^{s_1\times s_2}} \frac{1}{\|\mathbf{V}\|_{p,p^*}} \left(\sum_{j=1}^{s_2} (\|\mathbf{V}[,j]\|_p)^{p^*} \right)^{1/p^*} \\
&= \text{(RHS)}
\end{aligned}
$$

■

Lemma 8 *Assume that the function* $g(\mathbf{u}) = \max\{\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2, \cdots, \boldsymbol{a}_{2^J}^T\mathbf{u}+b_{2^J}\}$ *is defined on* $\mathbb{R}^s$*, and* $\|\boldsymbol{a}_i\|_1 + |b_i| \leq L$ *for* $i = 1, 2\cdots, 2^J$*. Then* g *could be represented by a neural network* $g = T_{J+1}\circ\sigma_{\text{ReLU}}\circ T_J\circ\cdots\circ\sigma_{\text{ReLU}}\circ T_1\circ\mathbf{u}$*, where* $T_1(\boldsymbol{v}) = (\mathbf{W}_1^{[J]})^T\boldsymbol{v} + \mathbf{B}_1^{[J]}$*,* $\mathbf{W}_1^{[J]} \in \mathbb{R}^{s\times 2^{J+1}}$*,* $T_i(\boldsymbol{v}) = (\mathbf{W}_i^{[J]})^T\boldsymbol{v}$*,* $\mathbf{W}_i^{[J]} \in \mathbb{R}^{2^{J+3-i}\times 2^{J+2-i}}$ *for* $i = 2, \cdots, J$*, and* $\mathbf{W}_{J+1}^{[J]} \in \mathbb{R}^{2^2\times 1}$*. There are in total* $2^{J+2}-4$ *hidden neurons in this neural network. In addition,* $\left\|(\mathbf{B}_1, \mathbf{W}_1^T)^T\right\|_{p,q} \leq L2^{1+\frac{J+1}{q}}$*,* $\left\|\mathbf{W}_i^{[J]}\right\|_{p,q} = 2^{\frac{3}{p}+\frac{J+2-i}{q}-1}$ *for* $i = 2, \cdots, J$*, and* $\mathbf{W}_{J+1}^{[J]} = (0.5, -0.5, 0.5, 0.5)^T$.

Proof The proof is inspired by [30].

We prove the statement by induction on J.

Let $J = 1$. As shown in Figure 2.3, define

$$\mathbf{W}_1 = \begin{pmatrix} \boldsymbol{a}_1 + \boldsymbol{a}_2 & -\boldsymbol{a}_1 - \boldsymbol{a}_2 & \boldsymbol{a}_1 - \boldsymbol{a}_2 & -\boldsymbol{a}_1 + \boldsymbol{a}_2 \end{pmatrix} \in \mathbb{R}^{s\times 4},$$

$$\mathbf{B}_1 = \begin{pmatrix} b_1 + b_2 & -b_1 - b_2 & b_- + b_2 & -b_1 + b_2 \end{pmatrix} \in \mathbb{R}^4$$

and

$$\mathbf{W}_C = \begin{pmatrix} 0.5 & -0.5 & 0.5 & 0.5 \end{pmatrix}^T \in \mathbb{R}^{4\times 1}$$

It is easy to verify that $\mathbf{W}_C^T \left[\sigma_{\text{ReLU}} \circ \left(\mathbf{W}_1^T \mathbf{u} + \mathbf{B}_1\right)\right]$ is equal to $\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2)$.

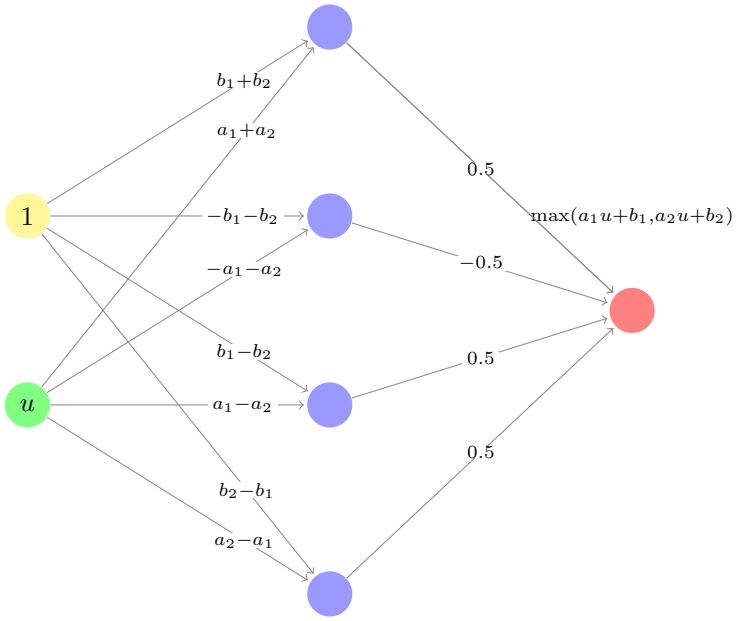


Fig. 2.3.: A WN-DNN representing $\max(\boldsymbol{a}_1^T\mathbf{u} + b_1, \boldsymbol{a}_2^T\mathbf{u} + b_2)$.

By Minkowski inequality and $\|\cdot\|_p \le \|\cdot\|_1$, we have

$$\begin{aligned} \left\|(\mathbf{B}_1, \mathbf{W}_1^T)^T\right\|_{p,q} &\le \left(4(\left\|(\boldsymbol{a}_1^T, b_1)\right\|_p + \left\|(\boldsymbol{a}_2^T, b_2)\right\|_p)^q\right)^{\frac{1}{q}} \\ &\le \left(4(\left\|(\boldsymbol{a}_1^T, b_1)\right\|_1 + \left\|(\boldsymbol{a}_2^T, b_2)\right\|_1)^q\right)^{\frac{1}{q}} \\ &\le L2^{1+\frac{2}{q}}, \end{aligned}$$

and

$$\|\mathbf{W}_C\|_p = 2^{\frac{2}{p}-1}.$$

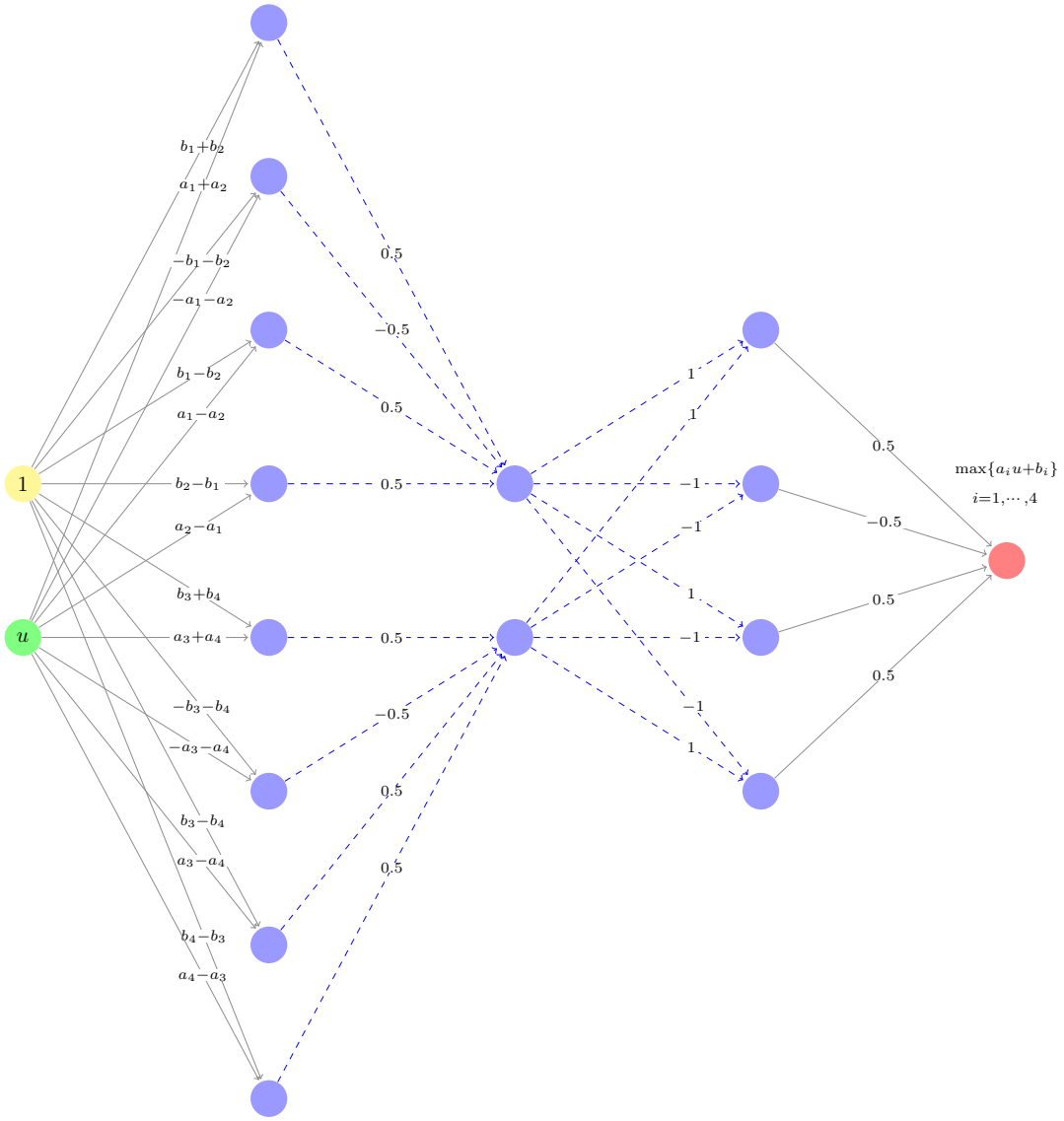


Fig. 2.4.: A WN-DNN representing $\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2, \boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4)$.

Let $J=2$. Treat $\max\{\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2, \boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4\}$ as the maximum of $\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2)$ and $\max(\boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4))$. As shown in Figure 2.4, $\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2) = \mathbf{W}_C{}^T\left[\sigma_{\text{ReLU}} \circ \left((\mathbf{W}_1^a)^T\mathbf{u}+\mathbf{B}_1^a\right)\right]$ and $\max(\boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4) = \mathbf{W}_C{}^T\left[\sigma_{\text{ReLU}} \circ \left((\mathbf{W}_1^b)^T\mathbf{u}+\mathbf{B}_1^b\right)\right]$, where

$$\mathbf{W}_1^a = \begin{pmatrix} \boldsymbol{a}_1+\boldsymbol{a}_2 & -\boldsymbol{a}_1-\boldsymbol{a}_2 & -\boldsymbol{a}_1+\boldsymbol{a}_2 & \boldsymbol{a}_1-\boldsymbol{a}_2 \end{pmatrix},$$

$$\mathbf{W}_1^b = \begin{pmatrix} \boldsymbol{a}_3+\boldsymbol{a}_4 & -\boldsymbol{a}_3-\boldsymbol{a}_4 & -\boldsymbol{a}_3+\boldsymbol{a}_4 & \boldsymbol{a}_3-\boldsymbol{a}_4 \end{pmatrix},$$

$$\mathbf{B}_1^a = \begin{pmatrix} b_1+b_2 & -b_1-b_2 & -b_1+b_2 & b_1-b_2 \end{pmatrix},$$
$$\mathbf{B}_1^b = \begin{pmatrix} b_3+b_4 & -b_3-b_4 & -b_3+b_4 & b_3-b_4 \end{pmatrix}.$$

We then put the two neural networks representing $\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2)$ and $\max(\boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4))$ in parallel and compute $\max\{\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2, \boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4\}$ by

$$(\mathbf{W}_C)^T\sigma_{\text{ReLU}} \circ \left((\mathbf{W}_1^0)^T(\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2), \max(\boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4))^T\right),$$

where

$$\mathbf{W}_1^0 = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}.$$

Consider the neural network in Figure 2.4. Since the second hidden layer is not activated, the two affine transformations (the blue dashed lines) could be combined as one. Equivalently,

$$\max\{\boldsymbol{a}_1^T\mathbf{u}+b_1, \boldsymbol{a}_2^T\mathbf{u}+b_2, \boldsymbol{a}_3^T\mathbf{u}+b_3, \boldsymbol{a}_4^T\mathbf{u}+b_4\} =$$
$$(\mathbf{W}_C)^T\left[\sigma_{\text{ReLU}} \circ \left((\mathbf{W}_2^{[2]})^T\left(\sigma_{\text{ReLU}} \circ (\mathbf{V}_1^{[2]})^T\mathbf{u}\right)\right)\right],$$

where

$$\mathbf{W}_2^{[2]} = \begin{pmatrix} \mathbf{W}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_C \end{pmatrix}\mathbf{W}_1^0, \quad \mathbf{V}_1^{[2]} = \begin{pmatrix} \mathbf{B}_1^a & \mathbf{B}_1^b \\ \mathbf{W}_1^a & \mathbf{W}_1^b \end{pmatrix} \in \mathbb{R}^{(s+1)\times 8}.$$

Note that

$$\left\|\mathbf{V}_1^{[2]}\right\|_{p,q} = \left(\left\|\begin{pmatrix} \mathbf{B}_1^a \\ \mathbf{W}_1^a \end{pmatrix}\right\|_{p,q}^q + \left\|\begin{pmatrix} \mathbf{B}_1^b \\ \mathbf{W}_1^b \end{pmatrix}\right\|_{p,q}^q\right)^{\frac{1}{q}} \le L2^{1+\frac{3}{q}}, \quad \left\|\mathbf{W}_2^{[2]}\right\|_{p,q} = 2^{\frac{3}{p}+\frac{2}{q}-1}.$$

For $J > 2$, we have

$$\max\{\boldsymbol{a}_1^T\mathbf{u}+b_1, \cdots, \boldsymbol{a}_{2^J}^T\mathbf{u}+b_{2^J}\} = (\mathbf{W}_C)^T\sigma_{\text{ReLU}} \circ$$
$$\left((\mathbf{W}_1^0)^T(\max(\boldsymbol{a}_1^T\mathbf{u}+b_1, \cdots, \boldsymbol{a}_{2^{J-1}}^T\mathbf{u}+b_{2^{J-1}}), \max(\boldsymbol{a}_{2^{J-1}+1}^T\mathbf{u}+b_{2^{J-1}+1} \cdots, \boldsymbol{a}_{2^J}^T\mathbf{u}+b_{2^J}))^T\right).$$

By the inductive hypothesis, we could put the two J-layer neural networks: $\max(\boldsymbol{a}_1^T\mathbf{u}+b_1,\cdots,\boldsymbol{a}_{2^{J-1}}^T\mathbf{u}+b_{2^{J-1}})$ and $\max(\boldsymbol{a}_{2^{J-1}+1}^T\mathbf{u}+b_{2^{J-1}+1}\cdots,\boldsymbol{a}_{2^J}^T\mathbf{u}+b_{2^J})$ in parallel. Furthermore, by Theorem 2.3.1 (d), the corresponding $L_{p,q}$ norm of each hidden layer will be $2^{\frac{1}{q}}$ times as large as that of $\max(u_1,\cdots,u_{2^{J-1}})$ or $\max(u_{2^{J-1}+1},\cdots,u_{2^J})$. Finally, similar to the case when $J=2$ in Figure 2.4, we could combine the original output layers of $\max(u_1,\cdots,u_{2^{J-1}})$ and $\max(u_{2^{J-1}+1},\cdots,u_{2^J})$ with $\mathbf{W}_1^0$. Thus complete the proof. ■

3. SPARSE DEEP NEURAL NETWORKS AND OVERFITTING

3.1 Introduction

Deep neural networks have recently attracted a lot of attentions due to their successful applications on many real-world applications [16]. The new advancement on optimization with SGD and graphical processing units (GPUs) makes the DNN training easy to scale to millions of parameters [44–46]. On the other hand, overfitting becomes a notorious feature of DNNs, which may lead to poor generalization. Recent works [47–49] show that the networks can be pruned significantly without any loss in accuracy. In the meanwhile, many other methods have also been developed to address the issue of overfitting, which includes early stopping, weight penalties of various kinds such as L_1 and L_2 regularizations, weight sharing [50], and dropout [51].

Empirical evidence suggests that inducing sparsity can relieve overfitting and save computation resources. A common strategy is to apply sparsity-inducing regularizers such as L_0 penalty [52] or the total number of parameters in the network [53]. However, theoretical investigations or justifications on sparse DNNs are less explored in the literature.

Weight normalization, by bounding the Euclidean norm of the incoming weights of each unit, has shown to be able to accelerate the convergence of stochastic gradient descent optimization across many applications [20]. In this chapter, we borrow the strength of weight normalization and induce the sparsity by bounding the $L_{1,\infty}$ norm of the weight matrix (including bias) for each layer. By doing this, we are able to induce the sparsity in a systematic way. Furthermore, we have developed capacity control for such models. It is shown that the generalization error upper bounds are independent of the network width and $\sqrt{k}$-dependence on the depth k of the network,

which are the best available bounds for networks *with bias neurons*. Our results provide theoretical justifications on the usage of such weight normalization, which leads to a sparse DNN. At the same time the generalization error has the minimal dependence on the network architecture. $L_{1,\infty}$ norm-constrained fully connected DNNs *without bias neurons* were investigated in prior studies [18, 21, 23, 26]. Rademacher complexity bounds in [18, 21, 23] is 2^k times larger than our result in Theorem 3.3.1 even without bias neurons in each hidden layer. Furthermore, it is hard to extend the work of [26] to the fully connected DNNs with bias neurons, especially when the activation function, such as the tanh activation function, fails to map the bias neuron to 1. As a comparison, our result is applicable to all Lipschitz continuous activation functions.

The overall contributions of the chapter are: (a). We have theoretically established the generalization error bounds for both regression and classification under the $L_{1,\infty}$-weight normalization for networks *with bias neurons*; (b). We have developed an easily implemented gradient projection descent algorithm to practically obtain a sparse neural network; (c). We have performed various experiments to validate our theory and demonstrate the effectiveness of the resulting approach.

The chapter is organized as follows. In Section 2, we define the sparse DNNs. Section 3 gives the Rademacher complexities, the generalization bounds for regression, and the generalization bounds for classification. In Section 4, we propose a gradient projection descent algorithm. Section 5 includes both synthetic and real world experiments to validate our theoretical findings.

3.2 The Model

In this section, we introduce the sparse deep neural networks.

3.2.1 Sparse DNNs

We begin with some notations for fully-connected neural networks. A neural network on $\mathbb{R}^{d_0} \to \mathbb{R}^{d_{k+1}}$ with k hidden layers is defined by a set of $k+1$ affine transformations $T_1 : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}, T_2 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}, \cdots, T_{k+1} : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$ and an activation function σ. In this chapter, we consider activation functions satisfying that $\sigma(0) = 0$. Note that this condition holds for widely used activation functions including ReLU and tanh. The affine transformations are parameterized by $T_\ell(\mathbf{u}) = \boldsymbol{W}_\ell^T \mathbf{u} + \mathbf{B}_\ell$, where $\boldsymbol{W}_\ell \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ and $\mathbf{B}_\ell \in \mathbb{R}^{d_\ell}$ for $\ell = 1, \cdots, k+1$. The function represented by this neural network is

$$f(x) = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \circ \boldsymbol{x}.$$

Before introducing the sparse DNNs, we build an augmented layer for each hidden layer by appending the bias neuron 1 to the original layer, and then combine the weight matrix and the bias vector to form a new matrix. We define the first hidden layer as

$$f_1(\boldsymbol{x}) = T_1 \circ \boldsymbol{x} \triangleq \langle \tilde{\mathbf{V}}_1, (1, \boldsymbol{x}^T)^T \rangle,$$

where $\tilde{\mathbf{V}}_1 = (\mathbf{B}_1, \boldsymbol{W}_1^T)^T \in \mathbb{R}^{(d_0+1) \times d_1}$.

Sequentially for $\ell = 2, \cdots, k$, define the ℓth hidden layer as

$$f_\ell(\boldsymbol{x}) = T_\ell \circ \sigma \circ f_{\ell-1}(\boldsymbol{x}) \triangleq \langle \tilde{\mathbf{V}}_\ell, (1, \sigma \circ f_{\ell-1}^T(\boldsymbol{x}))^T \rangle,$$

where $\tilde{\mathbf{V}}_\ell = (\mathbf{B}_\ell, \boldsymbol{W}_\ell^T)^T \in \mathbb{R}^{(d_{\ell-1}+1) \times d_\ell}$. The output layer is

$$f(\boldsymbol{x}) = T_{k+1} \circ \sigma \circ f_k(\boldsymbol{x}) \triangleq \langle \tilde{\mathbf{V}}_{k+1}, (1, \sigma \circ f_k^T(\boldsymbol{x}))^T \rangle,$$

where $\tilde{\mathbf{V}}_{k+1} = (\mathbf{B}_{k+1}, \boldsymbol{W}_{k+1}^T)^T \in \mathbb{R}^{(d_k+1) \times d_{k+1}}$.

The sparsity of the DNN could be controlled by setting proper constraints for the $L_{1,\infty}$ norm of each hidden layer, where the $L_{1,\infty}$ norm of a $s_1 \times s_2$ matrix A is defined as

$$\|A\|_{1,\infty} = \max_j \left(\sum_{i=1}^{s_1} |a_{ij}| \right).$$

Specifically, define $\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ as the collection of all sparse DNNs $f(\boldsymbol{x}) = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \circ \boldsymbol{x}$ satisfying:

(a) It has k hidden layers;

(b) The number of neurons in the ℓth hidden layer is d_ℓ for $\ell = 1, 2, \cdots, k$. The dimension of input is d_0, and output d_{k+1};

(c) $\|T_\ell\|_{1,\infty} \triangleq \left\|\tilde{\mathbf{V}}_\ell\right\|_{1,\infty} \leq c$ for $\ell = 1, \cdots, k$;

(d) The L_1 norm of the jth column of $\tilde{\mathbf{V}}_{k+1}$ is bounded by the jth element of $\boldsymbol{o}$: $\left\|\tilde{\mathbf{V}}_{k+1}[\cdot, j]\right\|_1 \leq o_j$ for $j = 1, \cdots, d_{k+1}$.

We call c the normalization constant in the rest of the chapter. Furthermore, define the collection of the sparse DNNs without any constraint on the output layer as

$$\mathcal{S}_c^{k,\boldsymbol{d},\sigma} = \cup_{\boldsymbol{o} \geq \boldsymbol{0}} \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}.$$

We focus more on the influence of c on the generalization behavior as well as the sparsity of the DNN, and we will provide the generalization bounds for the sparse DNNs with unconstrained output layers.

In order to obtain a sparse neural network, we need to transform our understanding of a problem into a loss function $L(\cdot, \cdot)$. Then it is equivalent to solve the optimization problem

$$\min_f \left\{ \sum_{i=1}^n L(f(\boldsymbol{x}_i), y_i) \,\middle|\, f \in \S_c^{k,\boldsymbol{d},\sigma} \right\}, \tag{3.1}$$

where $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^{m_1 \times 1}$ are the samples, k and $\boldsymbol{d}$ define the depth and widths of the DNNs, and the normalization constant c controls the sparsity. We focus more on the influence of c on the generalization behavior as well as the sparsity of the DNN, and we will provide the generalization bounds for the sparse DNNs with unconstrained output layers.

3.3 The Learning Theory

In this section, assume that $\mathcal{X} = [-1, 1]^{m_1}$, and the activation function σ is ρ_σ-Lipschitz continuous. Note that ReLU and tanh are both 1-Lipschitz continuous.

3.3.1 Rademacher Complexities

The *empirical Rademacher complexity* of the hypothesis class $\mathcal{F}$ with respect to a data set $S = \{z_1 \dots z_n\}$ is defined as:

$$\widehat{\mathfrak{R}}_S(\mathcal{F}) = \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(z_i) \right) \right],$$

where $\epsilon = \{\epsilon_1 \dots \epsilon_n\}$ are n independent Rademacher random variables. The *Rademacher complexity* of the hypothesis class $\mathcal{F}$ with respect to n samples is defined as:

$$\mathfrak{R}_n(\mathcal{F}) = \mathbb{E}_{S \sim \mathcal{D}^n} \left[\widehat{\mathfrak{R}}_S(\mathcal{F}) \right].$$

In the following theorem, we bound the Rademacher complexity of $\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}$ when the output dimension is one, which will be used later to obtain the generalization bounds for both regression and classification.

Theorem 3.3.1 *Fix $k \geq 0, c, o > 0, d_\ell \in \mathbb{N}_+$ for $\ell = 1, \cdots, k$, and $d_{k+1} = 1$, then for any set $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$, we have*

$$\widehat{\mathfrak{R}}_S(\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}) \leq o\sqrt{\frac{(k+1)\log 16}{n}} \left(\sum_{\ell=0}^{k} (c\rho_\sigma)^\ell + (c\rho_\sigma)^k \right) + o(c\rho_\sigma)^k \sqrt{\frac{2\log(2m_1)}{n}}.$$

Furthermore, if $c\rho_\sigma \geq 1$,

$$\widehat{\mathfrak{R}}_S(\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}) \leq \frac{1}{\sqrt{n}} o(c\rho_\sigma)^k (\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}).$$

Remark 1 *When $\log(m_1)$ is small, we briefly summarize the dependence of the above bound on k under different choice of c:*

- *$c\rho_\sigma < 1$: $O(\sqrt{k}\frac{1-(c\rho_\sigma)^{k+1}}{1-c\rho_\sigma})$*

- $c\rho_{\sigma} \geq 1$: $O(\sqrt{k}(c\rho_{\sigma})^k)$

The complexity bound does not depend on the width of the network. In addition to the product of the $L_{1,\infty}$ norms of each layer, the complexity bound depends on the depth k by $O(\sqrt{k})$ when $c\rho_{\sigma} < 1$, and $\sqrt{k}(c\rho_{\sigma})^k$ otherwise.

3.3.2 Generalization Bounds for Regression

In this section, consider a specific case of equation (1.1), where t is an identity transformation, and $m_2 = 1$. Assume the following conditions in this section:

(A1). $(\boldsymbol{x}, y)$ is a random variable of support $\mathcal{X} \times \mathcal{Y}$ and distribution $\mathcal{D}$, and $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is a dataset of n i.i.d. samples drawn from $\mathcal{D}$.

(A2). The normalization constant $c > 0$, the number of hidden layers $k \in [0, \infty)$, and widths $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ with $d_0 = m_1$ and $d_{k+1} = 1$.

(A3). For any $y \in \mathcal{Y}$, $|y| \leq B_0$.

In practice, the output is usually normalized to the range $[-1, 1]$. Thus it is reasonable to set $B_0 = 1$. An alternative way is to choose B_0 as the $(1 - \delta_0)$ quantile of $\{|y_i|, i = 1, \cdots, n\}$, where δ_0 is a constant in $(0, 1)$. In this case, a modified version of the following theorem still holds, as shown later in Remark 4.

Mean square error is defined as $L_S(f(\boldsymbol{x}), y) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$. The following theorem shows the generalization bound that holds uniformly for any sparse DNN in $\S_c^{k,\boldsymbol{d},\sigma}$. Note that the sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is a dataset randomly drawn from $\mathcal{D}$, the following statement holds with a high probability over the choice of the sample S.

Theorem 3.3.2 *Assume A1-A3 hold and* $c\rho_{\sigma} \geq 1$. *Fix* $\delta \in (0,1)$, *then with probability at least* $1-\delta$ *over the choice of the sample* S, *for* **every** *sparse DNN* $f_T \in \S_c^{k,\boldsymbol{d},\sigma}$, *we have*

$$\mathcal{E}_{L_S}(f_T) \leq \frac{\left(B_0 + (\|T_{k+1}\|_1 + 1)(c\rho_{\sigma})^k\right)^2}{\sqrt{n}} * \left(\sqrt{\log\sqrt{\frac{2}{\delta}} + \log(\|T_{k+1}\|_1 + 2)} + 2\sqrt{(k+3)\log 4} + 2\sqrt{2\log(2m_1)}\right). \tag{3.1}$$

Remark 2 *The first term* $\left(B_0 + (\|T_{k+1}\|_1 + 1)(c\rho_{\sigma})^k\right)^2$ *reflects the range of the loss function* L_S. *In addition to this, the generalization error depends on the probability* $1-\delta$ *by* $\sqrt{\log(1/\delta)}$, *the depth by* $\sqrt{k}$, *and the input dimension by* $\sqrt{\log m_1}$. *The case when* $c\rho_{\sigma} < 1$ *is discussed in the supplementary material. When* $\log m_1$ *is small, the generalization bound can be established as* $O\left(\frac{(B_0+\|T_{k+1}\|_1)^2}{\sqrt{n}}\left(\sqrt{\log\frac{1}{\delta}} + \sqrt{k}\frac{1-(c\rho_{\sigma})^{k+1}}{1-c\rho_{\sigma}} + (c\rho_{\sigma})^k\sqrt{\log m_1}\right)\right)$.

Remark 3 *It is similar to obtain the generalization error bound for the mean absolute error loss, defined as* $L_A(f(\boldsymbol{x}), y) = |y - f(\boldsymbol{x})|$. *In addition to the same assumption of Theorem 3.3.2, assume that* B_0 *is a constant. Then*

$$\mathcal{E}_{L_A}(f_T) \leq O\left(\frac{\|T_{k+1}\|_1}{\sqrt{n}}(c\rho_{\sigma})^k\left(\sqrt{\log\frac{1}{\delta}} + \sqrt{k} + \sqrt{\log m_1}\right)\right).$$

Remark 4 *Condition (A3) is not always met, especially when* $y = f(\boldsymbol{x}) + \epsilon$, *where* $\epsilon \sim N(0,1)$. *However, it is still reasonable to assume that* $\mathbb{P}(|y| \leq B_0) \geq 1 - \delta_0$ *for some fixed* $\delta_0 > 0$ *and* $B_0 > 0$. *Under this alternative assumption, Equation 3.1 is replaced by*

$$\mathcal{E}_{L_S}(f_T) \leq (1-\delta_0)\frac{\left(B_0 + (\|T_{k+1}\|_1 + 1)(c\rho_{\sigma})^k\right)^2}{\sqrt{n}} * \left(\sqrt{\log\sqrt{\frac{2}{\delta}} + \log(\|T_{k+1}\|_1 + 2)} + 2\sqrt{(k+3)\log 4} + 2\sqrt{2\log(2m_1)}\right).$$

3.3.3 Generalization Bounds for Classification

In this section, we consider the case of Equation (1.1) when t = argmax and $\mathcal{Y} = \{1, 2, \cdots, m_2\}$. In the rest of the paper, we define the jth element of a vector $\mathbf{z}$ by $z[j]$. In this subsection, assume the following conditions:

(B1). $(\boldsymbol{x}, y)$ is a random variable of support $\mathcal{X}\times\mathcal{Y}$ and distribution $\mathcal{D}$, and $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is a dataset of n i.i.d. samples drawn from $\mathcal{D}$.

(B2). The normalization constant $c > 0$, the number of hidden layers $k \in [0, \infty)$, and widths $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ with $d_0 = m_1$ and $d_{k+1} = m_2$.

The cross-entropy loss function is defined as

$$L_C(f(\boldsymbol{x}), y) = -\log \frac{\exp(\mathrm{f}(\boldsymbol{x})[\mathrm{y}])}{\sum_j \exp \mathrm{f}(\boldsymbol{x})[\mathrm{j}]}.$$

We then extend it to the sparse DNNs with unconstrained output layers. For any transformation $T(\mathbf{u}) = V^T(1, \mathbf{u}^T)^T$, define $T[j]$ as $V[, j]$: the jth column of V.

Theorem 3.3.3 *Assume B1-B2 hold and $c\rho_\sigma \geq 1$. Fix $\delta \in (0, 1)$. Then, with probability at least $1 - \delta$ over the choice of S, every $f_T \in \S_c^{k,\boldsymbol{d},\sigma}$ satisfies that*

$$\mathcal{E}_{L_C}(f_T) \leq O\left(\frac{(c\rho_\sigma)^k}{\sqrt{n}} \|T_{k+1}\|_{1,\infty} \left[\sqrt{\log\frac{1}{\delta}} + \frac{\|T_{k+1}\|_{1,1}}{\|T_{k+1}\|_{1,\infty}}\sqrt{m_2}(\sqrt{k} + \sqrt{\log m_1})\right]\right), \tag{3.2}$$

where $\|T\|_{1,1} = \sum_i \sum_j |v_{ij}|$, if $T(x) = \langle \mathbf{V}, (1, \boldsymbol{x})\rangle$.

Remark 5 *The first term $(c\rho_\sigma)^k \|T_{k+1}\|_{1,\infty}$ reflects the range of the neural network f_T. In addition, the generalization bound depends on the probability $1 - \delta$ by $O\left(\sqrt{\log\frac{1}{\delta}}\right)$. The generalization bound depends on the depth k, the input dimension m_1 and the number of classes m_2 by $O\left(\frac{\|T_{k+1}\|_{1,1}}{\|T_{k+1}\|_{1,\infty}}\sqrt{m_2}(\sqrt{k} + \sqrt{\log m_1})\right)$, respectively. This could be reduced to $O\left(\left(\frac{\|T_{k+1}\|_{1,1}}{\|T_{k+1}\|_{1,\infty}} + \sqrt{m_2}\right)(\sqrt{k} + \sqrt{\log m_1})\right)$ if $\exp(\|\mathrm{T_{k+1}}\|_{1,\infty} (\mathrm{c}\rho_\sigma)^{\mathrm{k}}) = \mathrm{O}(1)$. The case when $c\rho_\sigma < 1$ is discussed in the supplementary material. Under this assumption, the generalization bound rely on $c\rho_\sigma$ by $O(\frac{1-(c\rho_\sigma)^{k+1}}{1-c\rho_\sigma})$.*

Algorithm 1 Gradient Projection Descent Algorithm

In each iteration:

Input: $\tilde{\mathbf{V}}^{(t)} = (\tilde{\mathbf{V}}_1^{(t)}, \cdots, \tilde{\mathbf{V}}_k^{(t)})$

for all $\ell = 1, \ldots, k$ **do**

$\tilde{\mathbf{V}}_\ell^{(t+1)} := \tilde{\mathbf{V}}_\ell^{(t)} - \gamma_t \nabla L(\tilde{\mathbf{V}}_\ell^{(t)})$,

where γ_t is the stepsize at iteration t

for all columns $\boldsymbol{v}$ in $\mathbf{V}_\ell^{(t+1)}$ **do**

if $\|\boldsymbol{v}\|_1 > c$ **then**

$\boldsymbol{v} = \text{proj}_{\|\cdot\|_1 \le c}\boldsymbol{v}$ by **Algorithm 2**

end if

end for

end for

Output: $\tilde{\mathbf{V}}^{(t+1)} = (\tilde{\mathbf{V}}_1^{(t+1)}, \cdots, \tilde{\mathbf{V}}_k^{(t+1)})$

Remark 6 *Condition (A3) is not always met, especially when* $y = f(\boldsymbol{x}) + \epsilon$*, where* $\epsilon \sim N(0,1)$*. However, it is still reasonable to assume that* $\mathbb{P}(|z[j]| \le B_j) \ge 1 - \delta_0$ *for some fixed* $\delta_0 > 0$*. Under this alternative assumption, Equation 3.2 is replaced by*

$$\mathcal{E}_{L_C}(f_T) \le O\left((1-\delta_0)\frac{(c\rho_\sigma)^k}{\sqrt{n}}\left\|T_{k+1}\right\|_{1,\infty}\left[\sqrt{\log\frac{1}{\delta}} + \frac{\left\|T_{k+1}\right\|_{1,1}}{\left\|T_{k+1}\right\|_{1,\infty}}\sqrt{m_2}(\sqrt{k} + \sqrt{\log m_1})\right]\right).$$

3.4 The Algorithm

Algorithm 2 Projection to L_1 norm ball [54]

Input: $\boldsymbol{v} \in \mathbb{R}^s$, c

Sort abs($\boldsymbol{v}$) into $\mu : \mu_1 \geqslant \mu_2 \geqslant \cdots \geqslant \mu_s$

Find $p^* = \max\{p \in [s] : \mu_p - \frac{1}{p}(\sum_{q=1}^{p}\mu_q - c) > 0\}$

Define $\theta = \frac{1}{p^*}(\sum_{q=1}^{p^*}\mu_q - c)$

Output: $\boldsymbol{w}$ s.t. $\boldsymbol{w}_p = \text{sgn}(\boldsymbol{v}_p) \cdot \max\{\text{abs}(\boldsymbol{v}_p) - \theta, 0\}$

In this section, we propose a gradient projection descent algorithm to solve the optimization problem equation (3.1).

Recall that for a neural network $f(x) = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \circ \boldsymbol{x}$ with $T_\ell(\mathbf{u}) = \tilde{\mathbf{V}}_\ell^T(1, \mathbf{u}^T)^T$, we have $f \in \S_c^{k,\boldsymbol{d},\sigma}$ if and only if

$$\left\|\tilde{\mathbf{V}}_\ell\right\|_{1,\infty} \leq c \quad \forall \ell \text{ or } \left\|\tilde{\mathbf{V}}_\ell[\cdot, j]\right\|_1 \leq c \quad \forall\, \ell, j.$$

One idea is to solve the Lagrangian of equation (3.1) by a proximal minimization algorithm. However there is no closed form for the proximal operator with the $L_{1,\infty}$ norm. Another idea is to directly solve the constrained optimization problem by a gradient projection descent algorithm. Under this circumstance, the projection to an L_1 norm ball could be efficiently implemented while inducing the sparsity of its output [54].

Our gradient projection descent algorithm could be easily implemented as a variation of any gradient descent method, as shown in Algorithm 1. In each iteration of the original gradient descent method, we project its output to the sparse DNN function class by Algorithm 2. Note that the uniform convergence of the empirical risk to the true risk holds for any hypothesis defined in Theorems 3.3.2 and 3.3.3. Therefore, it applies to the gradient projection descent algorithm output too.

3.5 Numerical Results

In this section, we validate our theorem using both simulated and real data experiments. We first design two synthetic experiments to demonstrate the theoretical advantage of the sparse DNNs. Especially, we illustrate the power of $L_{1,\infty}$-weight normalization for high dimensional problems. Furthermore, we apply our algorithm on convolutional layers, and validate our theoretical findings on CIFAR-10 datasets. For each setting, we measure the training error, generalization error, test accuracy, and model sparsity in order to demonstrate c's influence on the generalization ability as well as the sparsity of the model. Recall that the generalization error is the difference between training error and test error. Note that we do not have access to the

underlying distribution of the input x and output y. Thus, the generalization error refers to the empirical loss on the test set in all experiments. Besides, test accuracy is the classification accuracy for testing data. The sparsity rate is the ratio of the number of zero parameter estimates to the size of weight matrices. In this section, we will use the following format $d_0 - d_1 - \cdots - d_{k+1}$ to define the architecture of a neural network, where k is the number of hidden layer of the neural network, d_0 is the input dimension, d_{k+1} is the output dimension, and d_i denotes the number of neurons in the the ith hidden layer.

3.5.1 The Regression Experiment

We evaluate our algorithm on a high-dimensional linear regression problem $y = \boldsymbol{x}^T\boldsymbol{\beta} + \varepsilon$, where the coefficient $\boldsymbol{\beta}$ is a sparse vector. We sample 500 random samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^{1000} \times \mathbb{R}, i = 1, \cdots, 500$ for training, and 1500 samples for testing from the distribution below.

1. Generate the coefficient $\boldsymbol{\beta}$ by $\beta_i \sim \text{Unif}(0.15, 150)$ for $i = 1, \cdots, 100$, while setting the rest of $\boldsymbol{\beta}$ to be zero.

2. For $\forall i$, first independently sample an auxiliary variable $\mathbf{z}_i \in \mathbb{R}^{1000}$ from $N(\mathbf{0}, \boldsymbol{I})$. Then generate $\boldsymbol{x}_i$ by $x_{i1} = z_{i1}$, and $x_{ij} = z_{ij} + 0.2(z_{i,j+1} + z_{i,j-1})$ for $j = 2, \cdots, 1000$. Finally sample y_i from $N(\boldsymbol{x}_i^T\boldsymbol{\beta}, 1)$

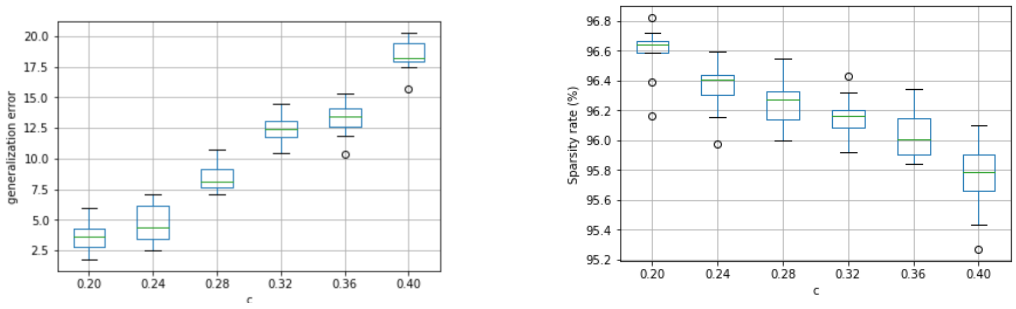
Note that we make the high-dimensional problem even more challenging in the presence of multicollinearity. We train the model with one fully connected layer with 300 output units using ReLU, and the loss function is mean square error. We summarize the results in Table 3.1, which are estimated by the mean of 10 repeated trials.

As shown in Figure 3.1(a) and Figure 3.1(b), as c increases, weight matrices become denser, and the generalization error increases, which matches the conclusion of Theorem 3.3.2.

When $c = \infty$, or equivalently with no regularization, even the model fits the training data perfectly, it suffers from serious overfitting. The above problem could

Table 3.1.: Training error, test error, generalization error and model sparsity for the regression experiment.

	train err	test err	gen err	sparsity %
$c = \infty$	0.000	69.520	69.520	3.02%
$c = 10.00$	0.000	35.571	35.571	41.58%
$c = 2.00$	0.052	8.129	8.077	61.42%
$c = 1.00$	0.131	2.424	2.426	84.69%
$c = 0.90$	0.173	2.424	2.251	87.62%
$c = 0.80$	0.197	2.384	2.186	89.80%
$c = 0.70$	0.235	2.334	2.099	91.09%
$c = 0.60$	0.252	2.247	1.994	91.78%
$c = 0.50$	0.286	2.140	1.854	93.33%
$c = 0.40$	0.387	2.209	1.822	93.88%
$c = 0.30$	0.850	2.526	1.675	94.18%



(a) Generalization error vs. c.

(b) Sparsity rate vs. c.

Fig. 3.1.: Boxplots of generalization error and sparsity rate with different c for the regression experiment.

be solved by applying $L_{1,\infty}$ weight normalization with a proper c, as the test error would be decreased by more than 95% if set $c = 0.20$.

3.5.2 The Classification Experiment

We first consider a high-dimensional nonlinear binary classification problem. We sample 500 random samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^{500} \times \{0, 1\}, i = 1, \cdots, 500$ for training, and 1000 samples for testing from the distribution below.

1. Generate $\alpha \sim N(0, 1)$.

2. For $\forall i$, independently sample x_{ij}: the jth element of $\boldsymbol{x}_i$, from $N(\frac{\alpha}{2}, \frac{1}{4})$ for $j = 1, \cdots, 500$, and

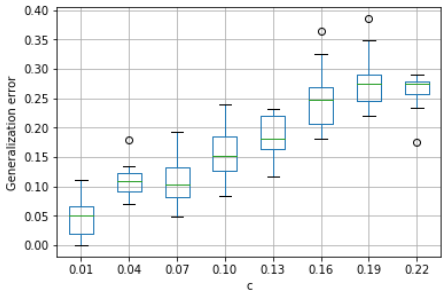
$$y_i = \begin{cases} 1, & e^{x_{i1}} + x_{i2}^2 + 5\sin(x_{i3}x_{i4}) - 3 > 0 \\ 0, & \text{otherwise} \end{cases}$$

We use a 500-100-50-20-2 fully connected neural network with ReLU, and the loss function is cross entropy. We report the results in Table 3.2, which are estimated by the mean of 10 repeated trials.

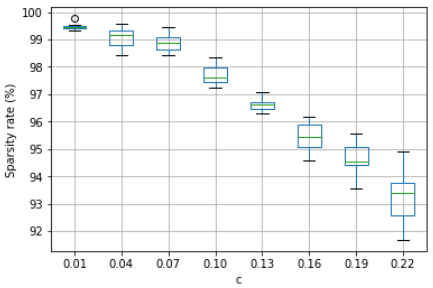
As illustrated in Figure 3.2(a), the generalization error decreases as c decreases, which matches the conclusion of Theorem 3.3.3. In the meanwhile, the network

Table 3.2.: Training error, generalization error, test accuracy, and model sparsity for the classification experiment.

	train err	gen err	test acc(%)	sparsity(%)
$c = \infty$	0.005	0.543	71.60	2.39
$c = 1.00$	0.016	0.624	83.50	66.71
$c = 0.50$	0.087	0.337	87.30	68.43
$c = 0.30$	0.053	0.297	88.40	90.78
$c = 0.22$	0.034	0.280	88.93	93.29
$c = 0.19$	0.046	0.273	89.10	94.53
$c = 0.16$	0.030	0.250	89.23	95.40
$c = 0.13$	0.077	0.177	89.93	96.60
$c = 0.10$	0.121	0.155	90.01	97.53
$c = 0.07$	0.207	0.102	90.12	98.98
$c = 0.04$	0.239	0.112	89.57	99.04
$c = 0.01$	0.265	0.068	88.48	99.49

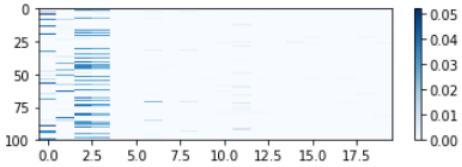


(a) Generalization error vs. c.

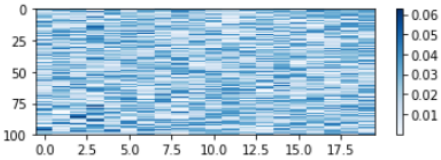


(b) Sparsity rate vs. c.

Fig. 3.2.: Boxplots of generalization error and sparsity rate with different c for the classification experiment.



(a) $L_{1,\infty}$-weight normalization with $c = 0.07$.

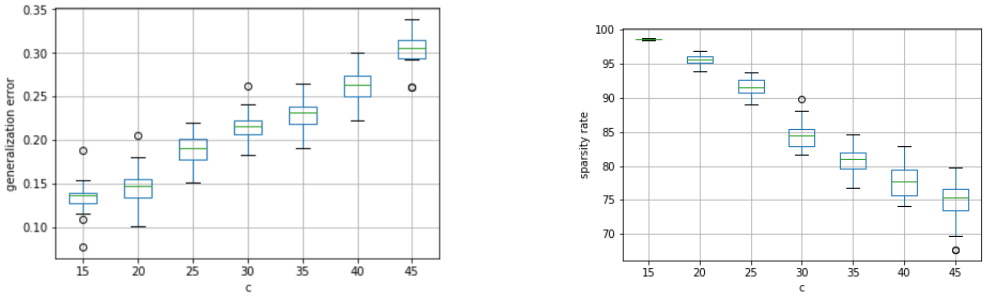


(b) With no regularization.

Fig. 3.3.: Visualization of the first twenty columns of the resulting weight matrix representing the first hidden layer with/without $L_{1,\infty}$-weight normalization.

becomes sparser with a smaller c, which is evident in Figure 3.2(b). However, there is a trade-off between approximation and generalization ability. A smaller c leads to a smaller generalization error. On the other hand, a small c limits the expressive power of the neural network. For example, decreasing c from 0.10 to 0.07 nearly doubles the training error.

With no regularization, the model fits the training data perfectly, however it performs poorly on the test dataset. We could improve the test accuracy by 19.4% using $L_{1,\infty}$-weight normalization with $c = 0.07$, while the resulting weight matrix are much sparser as shown in Figure 3.3. We also observe that the first 4 columns of the



(a) Generalization error vs. c.

(b) Sparsity rate vs. c.

Fig. 3.4.: Boxplots of generalization error and sparsity rate with different c for CIFAR-10 experiment.

resulting weight matrix are dense, while the others are sparse. It is because that only the first four elements of the input are included in the true model.

3.5.3 CIFAR-10

We extend our method to convolutional neural networks in the second experiment. CIFAR-10 [55] consists of 60000 32×32 color images in 10 classes. A small kernel size itself assumes the local sparsity, thus it is not necessary to apply $L_{1,\infty}$-weight normalization to convolutional layers with small kernel sizes. We use a modified VGG-16 to train the model, where the first two 3×3 convolutional layers are replaced by two 21×21 convolutional layers. Note that VGG-16 is a convolutional neural network model proposed by [56]. We have done a 20-fold cross-validation to test the models ability to predict new data.

As shown in Figure 3.4(b), when c increases, the network becomes denser. For example, when $c = 15$, the connection is very sparse as more than 95% of its elements are learned to be zero. However, if we increase c from 15 to 45, the sparsity rate will be reduced by almost 25%. Furthermore, Figure 3.4(a) indicates that picking a larger c might result in poorer generalization. For instance, the generalization error

doubles when c is increased from 15 to 45. Such observation matches the conclusion of Theorem 3.3.3.

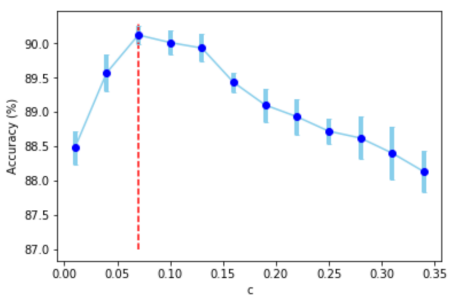
Table 3.3.: Comparison of different regularization on previous classification experiment, MNIST, and CIFAR10. Bold results are the best two methods in each experiment.

Regularizer	Cla. Exp	MNIST	CIFAR10
No	71.6	97.85	93.34
L_1, $\lambda = 0.1$	50.0	13.87	48.32
L_1, $\lambda = 0.01$	50.0	11.35	92.53
L_1, $\lambda = 0.001$	**81.6**	97.33	93.43
L_1, $\lambda = 0.0001$	73.2	97.99	93.64
L_2, $\lambda = 0.1$	70.8	80.30	80.41
L_2, $\lambda = 0.01$	73.4	92.30	90.38
L_2, $\lambda = 0.001$	73.2	94.43	91.41
L_2, $\lambda = 0.0001$	71.8	97.90	**93.72**
Dropout, $r_d = 0.5$	73.5	**98.11**	93.42
Our Approach	**91.0**	**98.03**	**93.75**

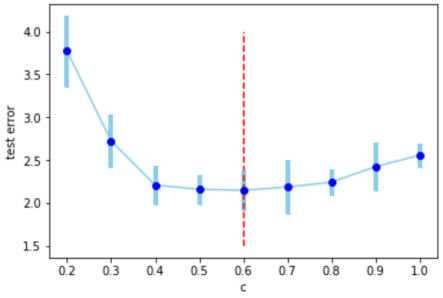
3.5.4 Selection of the Normalization Constant

The optimal normalization constant is chosen by k-fold cross validation.

We give examples of the selection of c on the classification and regression problem in Section 5.1 and 5.2. As shown in Figure 3.5, we plot the average cross validation score against the normalization constant c for both of the experiments. Then choose the optimal c that corresponds to the largest average cross validation score.



(a) Accuracy on test dataset vs. c in classification experiment.



(b) Test error vs. c in regression experiment.

Fig. 3.5.: Examples on the selection of c

3.5.5 Comparison with Other Regularizers

While establishing strong theoretical foundations for $L_{1,\infty}$-weight normalization, we show that our method performs well in practice by comparing with popular regularization techniques including L_1, L_2 (weight decay), and dropout regularizations on previous classification example, MNIST, and CIFAR-10 in terms of classification accuracy. These additional experiments are not intended to show the supreme of some well-tuned neural network architectures, but to illustrate the comparative performance of the $L_{1,\infty}$-weight normalization against other regularization methods via a fair comparison. Therefore, we only use some simple architectures for demonstration. In MNIST experiment, the input image is resized to 784×1, and then passed to a 900-10 fully connected neural network with ReLU. The loss function is cross entropy.

By using L_1 regularization with the hyperparameter λ, a penalty term $\lambda \sum \|\mathbf{W}\|_{1,1}$ is added to the original loss function, where $\mathbf{W}$ is the weight matrix. By implementing the L_2 regularization with the hyperparameter λ, a penalty term $\frac{1}{2}\lambda \sum \|\mathbf{W}\|_{2,2}^2$ is added to the original loss function. For each experiment, we compare different regularizers with various hyperparameters on the same baseline model to make a fair comparison. It is shown in Table 3.3 that our method is competitive with methods with other common regularizers.

3.6 Concluding Remarks

We have developed a systematic framework for sparse DNNs through $L_{1,\infty}$ weight normalization. We have established the Rademacher complexity of the related sparse DNN space. Based on this result, we have derived generalization error bounds for both regression and classification. The easily implemented gradient projection descent algorithm allows us to obtain a sparse DNN in practice. In experiments we have shown that the proposed $L_{1,\infty}$ minimization process leads to neural network sparsification, and is competitive with current approaches while empirically validating our theoretical findings.

We have so far used a single c to control the sparsity of the network. It is interesting to extend the current framework to the network with different cs at different layers. This poses additional challenges for computation to tune these hyperparameters. We are trying to use Bayesian optimization [57] to automatically select these hyperparameters. This research is currently under investigation and will be presented in another report.

3.7 Technical Lemmas

Lemma 9 *[40] Assume that the hypothesis class $\mathcal{F} \subseteq \{f|f : \mathcal{X} \to \mathbb{R}\}$, and $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{X}$. Let $G : \mathbb{R} \to \mathbb{R}$ be convex and increasing. Assume that the function $\phi : \mathbb{R} \to \mathbb{R}$ is L-Lipschitz continuous, and satisfies that $\phi(0) = 0$. We have:*

$$\mathbb{E}_\epsilon \left[G\left(\sup_{f\in\mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i \phi(f(\boldsymbol{x}_i)) \right) \right) \right] \leq \mathbb{E}_\epsilon \left[G\left(L \sup_{f\in\mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}_i) \right) \right) \right]$$

Lemma 10 *For any $f \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ and $\boldsymbol{x} \in \mathcal{X}$,*

$$\|f(\boldsymbol{x})\|_\infty \leq \|\boldsymbol{o}\|_\infty \max\left(1, (c\rho_\sigma)^k\right).$$

Proof We instead prove the result for any

$$f \in \mathcal{D}_{c,r}^{k,\boldsymbol{d},\sigma} \triangleq \{g : g \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}, \forall \boldsymbol{o} : \|\boldsymbol{o}\|_\infty \leq r\},$$

and complete the proof by induction on depth $k+1$. When $k=0$,

$$\begin{aligned}
\sup_{f\in\mathcal{D}_{c,r}^{0,}\boldsymbol{d},\sigma}\|f(\boldsymbol{x})\|_\infty &= \sup_{f\in\mathcal{D}_{c,r}^{0,}\boldsymbol{d},\sigma}\left\|\tilde{\mathbf{V}}_1^T(1,f_0^T(\boldsymbol{x}))^T\right\|_\infty \\
&= r\sup_{f\in\mathcal{D}_{c,r}^{0,}\boldsymbol{d},\sigma}\frac{\left\|\left(\tilde{\mathbf{V}}_1^T(1,f_0^T(\boldsymbol{x}))^T\right)\right\|_{p^*}}{\left\|\tilde{\mathbf{V}}_1\right\|_{1,\infty}} \\
&\le r\sup_{f\in\mathcal{D}_{c,r}^{0,}\boldsymbol{d},\sigma}\frac{1}{\left\|\tilde{\mathbf{V}}_1\right\|_{1,\infty}}\left\|\tilde{\mathbf{V}}_1^T(1,\boldsymbol{x}^T)^T\right\|_\infty \\
&\le r\max(1,\|\boldsymbol{x}\|_\infty) \\
&\le r.
\end{aligned}$$

Define $\boldsymbol{d}_{k+}=(d_0,\cdots,d_{k-1},d_k+1)$.

$$\begin{aligned}
\sup_{f\in\mathcal{D}_{c,r}^{k,}\boldsymbol{d},\sigma}\|f(\boldsymbol{x})\|_\infty &= \sup_{f\in\mathcal{D}_{c,r}^{k,}\boldsymbol{d},\sigma}\left\|\tilde{\mathbf{V}}_{k+1}^T(1,\sigma\circ f_k^T(\boldsymbol{x}))^T\right\|_\infty \\
&= r\sup_{f\in\mathcal{D}_{c,r}^{k,}\boldsymbol{d},\sigma}\frac{\left\|\left(\tilde{\mathbf{V}}_{k+1}^T(1,\sigma\circ f_k^T(\boldsymbol{x}))^T\right)\right\|_\infty}{\left\|\tilde{\mathbf{V}}_{k+1}\right\|_{1,\infty}} \\
&\le r\sup_{f\in\mathcal{D}_{c,r}^{k,}\boldsymbol{d},\sigma}\frac{1}{\left\|\tilde{\mathbf{V}}_{k+1}\right\|_{1,\infty}}\left\|\tilde{\mathbf{V}}_{k+1}^T(1,\sigma\circ f_k^T(\boldsymbol{x}))^T\right\|_\infty \\
&= r\sup_{f\in\mathcal{D}_{p,q,c,r}^{k,}\boldsymbol{d},\sigma}\frac{1}{\|\boldsymbol{v}\|_1}\left|\langle\boldsymbol{v},(1,\sigma\circ f_k^T(\boldsymbol{x}))^T\rangle\right| \\
&\le r\left\|(1,\sigma\circ f_k^T(\boldsymbol{x}))\right\|_\infty \\
&\le r\left\|(1,\rho_{\sigma}f_k^T(\boldsymbol{x}))\right\|_\infty \\
&\le r\max(1,c\rho_{\sigma})\sup_{f\in\mathcal{D}_{c,1}^{k-1,}\boldsymbol{d}_{k+},\sigma}\|f(\boldsymbol{x})\|_\infty
\end{aligned}$$

The penultimate step follows from the fact that

$$(1,\rho_{\sigma}f_k^T)^T\in\max(1,c\rho_{\sigma})\mathcal{D}_{c,1}^{k-1,}\boldsymbol{d}_{k+},\sigma.$$

Finally, the proof is completed by the induction assumption. ■

Lemma 11 *Assume A1-A2 hold. In addition, the loss function* $L(f(\boldsymbol{x}), y) : \mathcal{Y} \times \mathcal{Y} \to [0, A_0]$*, is* ρ*-Lipschitz continuous on its first argument. Fix* $\delta \in (0, 1)$ *and* $o > 0$*, then with probability at least* $1 - \delta$ *over the choice of the sample, every* $f \in \mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}$ *satisfies that*

$$\mathcal{E}_L(f) \leq A_0 \sqrt{\frac{\log(2/\delta)}{2n}} + \frac{2o\rho}{\sqrt{n}} \left[\sqrt{(k+1)\log 16} \left(\sum_{\ell=0}^{k} (c\rho_\sigma)^\ell + (c\rho_\sigma)^k \right) + (c\rho_\sigma)^k \sqrt{2\log(2m_1)} \right].$$

Furthermore, if $c\rho_\sigma \geq 1$*, with probability at least* $1 - \delta$ *over the choice of the sample, every* $f \in \mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}$ *satisfies that*

$$\mathcal{E}_L(f) \leq A_0 \sqrt{\frac{\log(2/\delta)}{2n}} + \frac{2o\rho}{\sqrt{n}} (c\rho_\sigma)^k (\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}).$$

Proof First, we upper bound $\mathfrak{R}_n(\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma})$ by the same bounds in Theorem 1, as Theorem 1 holds for any sample S under our assumptions. Then we could further bound the Rademacher complexity of the corresponding hypothesis class according to Lemma 9 and A2. Finally we get the desired result by Theorem 1.3.1. ■

Lemma 12 *Assume B1-B2 hold. In addition, the loss function* $L(f(\boldsymbol{x}), y) : \mathcal{Z} \times \mathcal{Y} \to [0, A_0]$*, satisfies that*

$$|L(f_1(\boldsymbol{x}), y) - L(f_2(\boldsymbol{x}), y)| \leq \rho \, \|f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})\|_2$$

for any $\boldsymbol{x} \in \mathcal{X}, y \in \mathcal{Y}$*. Fix* $\delta \in (0, 1)$ *and* $\boldsymbol{o} \geq \boldsymbol{0}$*, then with probability at least* $1 - \delta$ *over the choice of the sample, every* $f \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ *satisfies that*

$$\mathcal{E}_L(f) \leq A_0 \sqrt{\frac{\log(2/\delta)}{2n}} + \frac{2\sqrt{2}\rho}{\sqrt{n}} \left(\sum_{j=1}^{m_2} o_j \right) \left[\sqrt{(k+1)\log 16} \left(\sum_{\ell=0}^{k} (c\rho_\sigma)^\ell + (c\rho_\sigma)^k \right) + (c\rho_\sigma)^k \sqrt{2\log(2m_1)} \right].$$

Furthermore, if $c\rho_\sigma \geq 1$*, with probability at least* $1 - \delta$ *over the choice of the sample, every* $f \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ *satisfies that*

$$\mathcal{E}_L(f) \leq A_0 \sqrt{\frac{\log(2/\delta)}{2n}} + \frac{2\sqrt{2}\rho}{\sqrt{n}} \left(\sum_{j=1}^{m_2} o_j \right) (c\rho_\sigma)^k (\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}).$$

Proof First, we upper bound $\mathfrak{R}_n(\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma})$ by the same bounds in Theorem 1, as Theorem 1 holds for any sample S under our assumptions. Then we could further bound the Rademacher complexity of the corresponding hypothesis class by [36, Corollary 1] and B2. Finally get the desired result by Theorem 1.3.1. ■

3.8 Detailed Proofs

Proof of Theorem 3.3.1

Proof It is a direct conclusion of Theorem 2.4.1 and Corollary 1. ■

Proof of Theorem 2

We provide a general version of Theorem 2 with no assumption on the values of c or ρ_σ. Theorem 2 is the direct conclusion of the proposition below.

Proposition 3.8.1 *Assume A1-A3 hold. Fix $\delta \in (0,1)$, then with probability at least $1-\delta$ over the choice of the sample, for* **every** *sparse DNN $f_T \in \S_c^{k,\boldsymbol{d},\sigma} = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1$, we have*

$$\begin{aligned}\mathcal{E}_{L_S}(f_T) \leq & \sqrt{\frac{\log(\frac{2}{\delta}) + 2\log(\|T_{k+1}\|_1 + 2)}{2n}} \left(B_0^2 + (\|T_{k+1}\|_1 + 1)^2 \max\left(1, (c\rho_\sigma)^{2k}\right)\right) + \\ & \frac{2}{\sqrt{n}} \left(B_0 + (\|T_{k+1}\|_1 + 1)\max\left(1, (c\rho_\sigma)^k\right)\right)(\|T_{k+1}\|_1 + 1) * \\ & \left[\sqrt{(k+1)\log 16}\left(\sum_{\ell=0}^{k}(c\rho_\sigma)^\ell + (c\rho_\sigma)^k\right) + (c\rho_\sigma)^k\sqrt{2\log(2m_1)}\right].\end{aligned}$$

Furthermore, if $c\rho_\sigma \geq 1$, With probability at least $1-\delta$ over the choice of the sample, for **every** *sparse DNN $f_T \in \S_c^{k,\boldsymbol{d},\sigma} = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1$, we have*

$$\begin{aligned}\mathcal{E}_{L_S}(f_T) \leq & \sqrt{\frac{\log(\frac{2}{\delta}) + 2\log(\|T_{k+1}\|_1 + 2)}{2n}} \left(B_0^2 + (\|T_{k+1}\|_1 + 1)^2 (c\rho_\sigma)^{2k}\right) + \\ & \frac{2}{\sqrt{n}} \left(B_0 + (\|T_{k+1}\|_1 + 1)(c\rho_\sigma)^k\right)(\|T_{k+1}\|_1 + 1)(c\rho_\sigma)^k(\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}).\end{aligned}$$

Proof The proof is inspired by [24]. Given a positive integer l, Define a set

$$\mathcal{B}(l) = \mathcal{SN}_{c,l}^{k,\boldsymbol{d},\sigma}.$$

Correspondingly subdivide δ as

$$\delta(l) = \frac{\delta}{l(l+1)}.$$

Fix any l, we could get the corresponding generalization bounds as an instance of Lemma 11. By Lemma 3, for any $f \in \mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}, \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}$, we have

$$\left|\frac{\partial L_S(f(\boldsymbol{x}), y)}{\partial f(\boldsymbol{x})}\right| = |f(\boldsymbol{x}) - y| \leq o \max\left(1, (c\rho_{\sigma})^k\right) + B_0 \tag{3.1}$$

and

$$|L_S(f(\boldsymbol{x}), \boldsymbol{y})| = \frac{1}{2}(y - f(\boldsymbol{x}))^2 \leq \left(B_0^2 + o^2 \max\left(1, (c\rho_{\sigma})^{2k}\right)\right). \tag{3.2}$$

Thus for the mean square error, we could replace ρ and A_0 in Lemma 4 with equations (3.1) and (3.2), respectively, and get the corresponding generalization bound.

As $\sum_{l \in \mathcal{N}_+} \delta(l) = \delta$, the preceding bound holds simultaneously for all functions in the union $\cup\{\mathcal{B}(l) : l \in \mathcal{N}_+\}$ with probability at least $1 - \delta$. Thus given f_T, choose the smallest l such that $f_T \in \mathcal{B}(l)$. As $T_{k+1}(\mathbf{u}) = \tilde{V}_{k+1}^T(1, \mathbf{u}^T)^T$, then the smallest l satisfies that

$$l \leq \|T_{k+1}\|_1 + 1.$$

Further replace the l's with $\|T_{k+1}\|_1 + 1$, thus we get the desired result. ■

3.8.1 Generalization Bounds for The Mean Absolute Error Loss

Proposition 3.8.2 *Assume A1-A3 hold. Fix $\delta \in (0, 1)$, then with probability at least $1 - \delta$ over the choice of the sample, for* **every** *sparse DNN $f_T \in \S_c^{k,\boldsymbol{d},\sigma} = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1$, we have*

$$\begin{aligned}\mathcal{E}_{L_A}(f_T) \leq & \sqrt{\frac{\log(\frac{2}{\delta}) + 2\log(\|T_{k+1}\|_1 + 2)}{2n}} \left(B_0 + (\|T_{k+1}\|_1 + 1)\max\left(1, (c\rho_{\sigma})^k\right)\right) + \\ & \frac{2}{\sqrt{n}}(\|T_{k+1}\|_1 + 1)\left[\sqrt{(k+1)\log 16}\left(\sum_{\ell=0}^{k}(c\rho_{\sigma})^{\ell} + (c\rho_{\sigma})^k\right) + (c\rho_{\sigma})^k\sqrt{2\log(2m_1)}\right].\end{aligned}$$

Furthermore, if $c\rho_{\sigma} \geq 1$*, With probability at least* $1-\delta$ *over the choice of the sample, for* **every** *sparse DNN* $f_T \in \S_c^{k,\boldsymbol{d},\sigma} = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1$*, we have*

$$\mathcal{E}_{L_A}(f_T) \leq \sqrt{\frac{\log(\frac{2}{\delta}) + 2\log(\|T_{k+1}\|_1 + 2)}{2n}} \left(B_0 + (\|T_{k+1}\|_1 + 1)(c\rho_{\sigma})^k\right) + \frac{2}{\sqrt{n}}(\|T_{k+1}\|_1 + 1)(c\rho_{\sigma})^k(\sqrt{(k+3)\log 4} + \sqrt{2\log(2m_1)}).$$

Proof For any $f \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}, \boldsymbol{x} \in \mathcal{X}, y \in \mathcal{Y}$, we have

$$1 \in |\partial_s L_A(f(\boldsymbol{x}), y)|, \tag{3.3}$$

where ∂_s is the subgradient of L at $f(\boldsymbol{x})$, and by Lemma 3,

$$|L_A(f(\boldsymbol{x}), y)| \leq \left(B_0 + o\max\left(1, (c\rho_{\sigma})^k\right)\right). \tag{3.4}$$

Thus it is straightforward to apply Lemma 4 to this specific case with ρ and A_0 given in equations (3.3) and (3.4) respectively. Furthermore, we could derive the corresponding generalization bounds for the sparse DNNs in $\S_c^{k,\boldsymbol{d},\sigma}$ using the same proof technique as in Theorem 2. ■

Proof of Theorem 3

We provide a general version of Theorem 3 with no assumption on the values of c or ρ_{σ}. Theorem 3 is the direct conclusion of the proposition below.

Proposition 3.8.3 *Assume B1-B2 hold. Fix* $\delta \in (0,1)$*,* $c > 0$*, the number of hidden layers* $k \in [0, \infty)$*, and widths* $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ *with* $d_0 = m_1$ *and* $d_{k+1} = 1$*. With probability*

at least $1-\delta$ *over the choice of the sample, for* **every** *sparse DNN* $f_T \in \S_c^{k,\boldsymbol{d},\sigma} = T_{k+1}\circ\sigma\circ T_k\circ\cdots\circ\sigma\circ T_1$*, we have*

$$\begin{aligned}\mathcal{E}_{L_C}(f_T) \le \left(2(\|T_{k+1}\|_{1,\infty}+\frac{1}{m_2})\max\left(1,(c\rho_{\boldsymbol{\sigma}})^k\right)+\log m_2\right)*\\ \sqrt{\frac{\log\sqrt{\frac{2}{\delta}}+\sum_{j=1}^{m_2}\log(m_2\|T_{k+1}[j]\|_1+2)}{n}}+\frac{2\sqrt{2}}{\sqrt{n}}\left(\|T_{k+1}\|_{1,1}+1\right)*\\ \left(1+\frac{\sqrt{m_2-1}}{1+(m_2-1)\exp(-2(\|\mathrm{T_{k+1}}\|_{1,\infty}+\frac{1}{\mathrm{m_2}})\max\left(1,(\mathrm{c}\rho_{\boldsymbol{\sigma}})^{\mathrm{k}}\right))}\right)*\\ \left[\sqrt{(k+1)\log 16}\left(\sum_{\ell=0}^{k}(c\rho_{\boldsymbol{\sigma}})^{\ell}+(c\rho_{\boldsymbol{\sigma}})^k\right)+(c\rho_{\boldsymbol{\sigma}})^k\sqrt{2\log(2m_1)}\right].\end{aligned}$$

Furthermore, if $c\rho_{\boldsymbol{\sigma}}\ge 1$*, With probability at least* $1-\delta$ *over the choice of the sample, for* **every** *sparse DNN* $f_T \in \S_c^{k,\boldsymbol{d},\sigma} = T_{k+1}\circ\sigma\circ T_k\circ\cdots\circ\sigma\circ T_1$*, we have*

$$\begin{aligned}\mathcal{E}_{L_C}(f_T) \le \left(2(\|T_{k+1}\|_{1,\infty}+\frac{1}{m_2})(c\rho_{\boldsymbol{\sigma}})^k+\log m_2\right)\sqrt{\frac{\log\sqrt{\frac{2}{\delta}}+\sum_{j=1}^{m_2}\log(m_2\|T_{k+1}[j]\|_1+2)}{n}}\\ +\frac{2\sqrt{2}}{\sqrt{n}}\left(\|T_{k+1}\|_{1,1}+1\right)\left(1+\frac{\sqrt{m_2-1}}{1+(m_2-1)\exp\left(-2(\|\mathrm{T_{k+1}}\|_{1,\infty}+\frac{1}{\mathrm{m_2}})(\mathrm{c}\rho_{\boldsymbol{\sigma}})^{\mathrm{k}}\right)}\right)*\\ (c\rho_{\boldsymbol{\sigma}})^k(\sqrt{(k+3)\log 4}+\sqrt{2\log(2m_1)}).\end{aligned}$$

Proof The proof is inspired by [24]. Given positive integers $\boldsymbol{l}=(l_1,\cdots,l_{m_2})$, define a set

$$\mathcal{B}(\boldsymbol{l}) = \mathcal{SN}_{c,\boldsymbol{l}/m_2}^{k,\boldsymbol{d},\sigma}.$$

Correspondingly subdivide δ as

$$\delta(\boldsymbol{l}) = \frac{\delta}{l_1(l_1+1)\cdots l_{m_2}(l_{m_2}+1)}.$$

Fix any $\boldsymbol{l}$, we get the corresponding generalization bound as an instance of Lemma 12. Consider $f\in\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}, \boldsymbol{x}\in\mathcal{X}, y\in\mathcal{Y}$. For $j'\neq y$,

$$\left|\frac{\partial L_C(f(\boldsymbol{x}),y)}{\partial f(\boldsymbol{x})[j']}\right| \le 1/(1+\sum_{j\neq j'}\exp\left(-(\mathrm{o_j}+\mathrm{o_{j'}})\max\left(1,(\mathrm{c}\rho_{\boldsymbol{\sigma}})^{\mathrm{k}}\right)\right).$$

For y,

$$\left|\frac{\partial L_C(f(\boldsymbol{x}),y)}{\partial f(\boldsymbol{x})[y]}\right| = \left|1-\frac{1}{1+\sum\limits_{j\neq y}\exp\left(\mathrm{f}(\boldsymbol{x})[\mathrm{j}]-\mathrm{f}(\boldsymbol{x})[\mathrm{y}]\right)}\right|$$
$$\leq \left|1-\frac{1}{1+\sum\limits_{j\neq y}\exp\left((\mathrm{o_j}+\mathrm{o_y})\max\left(1,(\mathrm{c}\rho_{\boldsymbol{\sigma}})^{\mathrm{k}}\right)\right)}\right|.$$

Additionally,

$$|L_C(f(\boldsymbol{x}),y)| \leq \max_{j'}\log\left(\sum_{j=1}^{m_2}\exp\{(\mathrm{o_j}+\mathrm{o_j'})\max\left(1,(\mathrm{c}\rho_{\boldsymbol{\sigma}})^{\mathrm{k}}\right)\}\right).$$

For simplicity, we assume $o_j \leq o_0$ for $j = 1, \cdots, m_2$, then

$$\left\|\frac{\partial L_C(f(\boldsymbol{x}),y)}{\partial f(\boldsymbol{x})}\right\|_2 \leq 1+\frac{\sqrt{m_2-1}}{1+(m_2-1)\exp(-2\mathrm{o}_0\max\left(1,(\mathrm{c}\rho_{\boldsymbol{\sigma}})^{\mathrm{k}}\right))} \tag{3.5}$$

and

$$|L_C(f(\boldsymbol{x}),y)| \leq 2o_0\max\left(1,(c\rho_{\boldsymbol{\sigma}})^k\right)+\log m_2. \tag{3.6}$$

We could replace ρ and A_0 in Lemma 12 with equations (3.5) and (3.6), respectively, and get the corresponding generalization bound for $\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$.

As $\sum_{\boldsymbol{l}\in\mathcal{N}_+^{m_2}}\delta(\boldsymbol{l})=\delta$, the preceding bound holds simultaneously for all functions in the union $\cup\{\mathcal{B}(\boldsymbol{l}):\boldsymbol{l}\in\mathcal{N}_+^{m_2}\}$ with probability at least $1-\delta$. Thus given f_T, choose the smallest $\boldsymbol{l}$ such that $f_T\in\mathcal{B}(\boldsymbol{l})$. As $T_{k+1}(\mathbf{u})=\tilde{V}_{k+1}^T(1,\mathbf{u}^T)^T$, then the smallest $\boldsymbol{l}$ satisfies that

$$l_j \leq m_2\left\|T_{k+1}[j]\right\|_1+1, \forall j.$$

Therefore

$$\sum_{j=1}^{m_2}\frac{l_j}{m_2}\leq\left\|T_{k+1}\right\|_{1,1}+1, \quad \max_j l_j\leq m_2\left\|T_{k+1}\right\|_{1,\infty}+1.$$

Therefore we get the desired result. ■

3.9 Additional Experiments

We extend the classification experiment in Section 5.2.

Firstly, we examine the effect of the sample size on generalization. As shown in Table 3.4, when the sample size increases, the generalization error becomes smaller, while having the normalization constant c fixed.

Table 3.4.: Generalization error/test accuracy for the classification experiment with different values of c and sample sizes.

	size=500	size=1000	size=1500	size=2000	size=2500
$c = \infty$	1.674/69.90	1.576/71.00	1.528/72.20	1.508/75.70	1.489/76.60
$c = 0.16$	0.441/87.03	0.343/88.10	0.258/89.30	0.208/91.50	0.199/92.74
$c = 0.13$	0.376/87.23	0.334/87.80	0.252/89.47	0.171/91.76	0.171/92.80
$c = 0.10$	0.324/87.78	0.280/87.60	0.223/90.30	0.176/90.70	0.169/90.80
$c = 0.07$	0.260/88.34	0.241/87.80	0.189/90.80	0.162/91.21	0.133/91.86
$c = 0.04$	0.134/89.57	0.112/89.94	0.102/91.31	0.084/91.72	0.073/92.24
$c = 0.01$	0.068/88.48	0.079/89.15	0.034/90.30	0.036/91.00	0.024/91.47

Secondly, we check the relationship between the depth of the neural network and the generalization error. The result is shown in Table 3.5. When c is relatively large, the generalization error increases, as the neural network grows deeper. On the contrary, when $c = 0.04, 0.01$, the generalization error might even decrease, as the depth increases. This might be caused by the shrinkage of the term (c^k).

Thirdly, we show that the projection gradient descent algorithm is not sensitive to the initial step size γ_0, as shown in Table 3.6.

Table 3.5.: Generalization error/test accuracy for the classification experiment in Section 5.2 with different network structures and sample sizes.

	100-20-2	100-50-20-2	100-100-50-20-2
∞	1.535/70.30	1.674/69.90	1.710/69.10
$c = 0.50$	0.461/83.14	0.478/84.78	0.542/82.42
$c = 0.16$	0.351/84.67	0.441/87.03	0.456/84.41
$c = 0.13$	0.322/85.35	0.376/87.23	0.431/84.89
$c = 0.10$	0.312/86.10	0.324/87.78	0.383/86.03
$c = 0.07$	0.245/88.42	0.260/88.34	0.274/87.98
$c = 0.04$	0.103/89.12	0.134/89.57	0.131/88.33
$c = 0.01$	0.072/87.74	0.068/88.48	0.094/87.52

Table 3.6.: Effect of the initial step size γ_0 on the algorithm.

γ_0	0.050	0.045	0.040	0.035	0.030
Training error (%)	90.15	90.04	90.12	90.07	90.10

4. ON THE STATISTICAL EFFICIENCY OF COMPOSITIONAL NONPARAMETRIC PREDICTION

4.1 Introduction

Nonparametric methods, such as spline-based methods and kernel-based methods, have been widely used in the past 20 years. Most existing methods make assumptions regarding the structure of the model in terms of interactions. For instance, the work of [58] assumes an additive structure of the predictor function, while in [59] the kernel family is defined as polynomial combinations of base kernels of a fixed degree. On the one hand, there is usually insufficient evidence from the data to support the assumption of a specific structure. On the other hand, inclusion of all interactions especially of high order terms would be burdensome for computing especially when the data is high dimensional. A commonly used strategy is to only include low order interactions into the model [59]. However, this would still be a restrictive assumption.

Our goal is to discover the complex structure of the *predictor* function in a concise manner. In contrast, existing methods focus on the discovery of the structure of *kernels* [59, 60]. As an illustrative example for predictor functions, consider the work of Schmidt et al. [61], which discovered physical laws from experimental data, and provided concise analytical expressions that are amenable to human interpretation.

We build our model by compositionally adding or multiplying basis functions applied to specific dimensions of the covariate. This model is structurally equivalent to a labeled binary tree. The sum-product structure has demonstrated its versatility for several problems. Examples include sum-product networks for computation of partition functions and marginals of high-dimensional distributions [62] and structure discovery in nonparametric regression for automatic selection of the kernel family [60].

Our model is a generalization of several popular methods. For illustration, consider the following examples:

- Tensor product spline surfaces [63]: Assume there are two covariates $\boldsymbol{x} = (x_1, x_2)$, and define
$$f(\boldsymbol{x}) = \sum_{i=1}^{q} \sum_{j=1}^{q} \beta_{ij} \phi_i(x_1) \phi_j(x_2),$$
given the basis functions $\phi_1, \ldots, \phi_q : \mathbb{R} \to \mathbb{R}$. For simplicity, assume $q = 2$, then Figure 4.1(a) is one visualization of f, where $\beta_{11} = w_1 w_3, \beta_{12} = w_1 w_4, \beta_{21} = w_2 w_3, \beta_{22} = w_2 w_4$.

- Sparse additive models [58]: Assume that $f(\boldsymbol{x})$ has an additive decomposition, where $\boldsymbol{x} = (x_1, \ldots, x_{m_1})$. Define
$$f(\boldsymbol{x}) = \sum_{j=1}^{m_1} \phi_{a_j}(x_j),$$
where $a_1, \ldots, a_{m_1} \in \{1, \ldots, q\}$ and such that $\sum_{j=1}^{m_1} \mathbb{I}(\phi_{a_j} \neq 0) \leq s$ for some integer $s \ll m_1$.

- Tensor decomposition: Given a set of q functions $\phi_1, \ldots, \phi_q$ and a tensor y_{ijk} for $i, j, k = 1, \ldots, m_1$. The problem is to find the indices $a_r, b_r, c_r \in \{1, \ldots, q\}$ for $r = 1, \ldots, R$, that minimize:
$$\sum_{i=1}^{m_1} \sum_{j=1}^{m_1} \sum_{k=1}^{m_1} \left(\sum_{r=1}^{R} w_r \phi_{a_r}(i) \phi_{b_r}(j) \phi_{c_r}(k) - y_{ijk} \right)^2.$$
Note that $\sum_{r=1}^{R} w_r \phi_{a_r}(i) \phi_{b_r}(j) \phi_{c_r}(k)$ can be written as a fixed weighted labeled binary tree. Figure 4.1(b) illustrates the case when $R = 2$.

Our contribution is as follows. First, we propose a general compositional sum-product nonparametric method, in which a model is expressed as a weighted labeled binary tree. Second, we provide a generalization bound that holds for any data distribution and any weighted labeled binary tree. We show that $O(k \log(m_1 q) + \log k!)$ samples

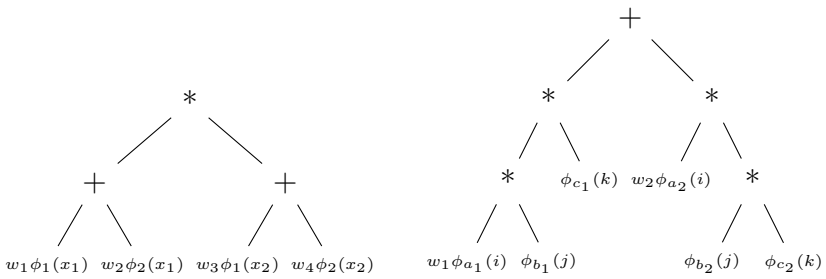
are sufficient, by using Rademacher-complexity arguments. Third, we further show that $\Omega(k \log(m_1 q) - \log k!)$ samples are necessary, by using information-theoretic arguments. Thus, our sample complexity bounds are tight. Furthermore, since the sample complexity is *logarithmic* in m_1 and q, our method is statistically suitable for high dimensions and a large number of basis functions. Finally, we propose a well-motivated greedy algorithm for regression in order to validate our theoretical findings.

For comparison with results on sparse additive models, the work of [58] presents an L_1-regularization approach. Additionally, a sample complexity of $O(q \log((m_1 - s)q))$ was shown to be sufficient for the correct identification of the basis functions in the sparse additive model. Note that in our work, we are interested in generalization bounds for the prediction error. The necessary number of samples for sparse additive models was analyzed in [64], where a sample complexity of $\Omega(s \log m_1)$ was found for the recovery of a function that is close to the true function in L_2-norm. Our sample complexity guarantee of $O(k \log m_1)$ matches this bound.

The chapter is structured as follows. In Section 4.2, we propose the compositional nonparametric trees for the general prediction problem. In Section 4.3, we provide a generalization bound. Section 4.4 discusses the necessary number of samples. In Section 4.5, we propose a greedy search algorithm for regression. In Section 4.6, we validate our theoretical results through synthetic experiments and apply our methods on two real-world data sets. Section 4.7 is the concluding remarks, and Section 4.8 contains the detailed proofs. Finally, Section 4.9 includes the detailed greedy search algorithm.

4.2 Compositional Nonparametric Trees for the General Prediction Problem

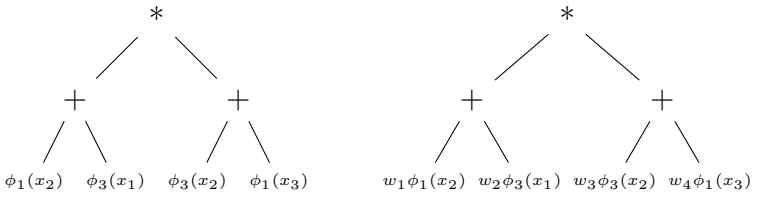
In this section, we propose a solution to the general prediction problem - Equation (1.1) via a compositional nonparametric method, in which a model is defined as a



(a) Tensor product spline surfaces. (b) Tensor decomposition.

Fig. 4.1.: Examples of tensor product spline surfaces and tensor decomposition.

weighted labeled binary tree. In this tree, each node represents a multiplication, an addition, or the application of a basis function to a particular covariate. Note that we assume that $m_2 = 1$ in this section. We provide two examples in order to illustrate how to adopt Equation (1.1) to different settings. For regression, we define $t(z) = z$, while for classification, we define $t(z) = \text{sign}(z)$.



(a) A labeled binary tree. (b) A weighted labeled binary tree.

Fig. 4.2.: Two tree examples.

The Labeled Binary Tree. We define a functional structure built compositionally by adding and multiplying a small number of basis functions. A straightforward visualization of this structure is a labeled binary tree. Given an infinite set of basis functions $\Phi = \{\phi_l, l = 1, 2, \cdots, \infty\}$ on $\mathbb{R} \to [-1, 1]$ and a truncation parameter q, $\mathcal{G}_{2k+1}$ is a set of binary trees where:

1. there are no more than $2k + 1$ nodes,

2. the labels of non-leaf nodes can be either "+" or "*",

3. the label of a leaf node can only be a function in Φ on a specific dimension of the covariate $\boldsymbol{x} = (x_1, \ldots, x_{m_1})$, that is $\phi_i(x_j)$ for any $i = 1, \ldots, q$ and $j = 1, \ldots, m_1$,

Figure 4.2(a) gives an example of a labeled binary tree with seven nodes. All the leaves are $\phi_i(x_j)$s, while all non-leaf nodes are operations. Note that if we switch the left sub-tree and the right sub-tree, we obtain an equivalent structure.

As pointed out later in Remark 7, in the nonparametric setting, both k and q are allowed to grow as a function of n.

The Weighted Labeled Binary Tree. It is easy to show that a labeled binary tree with $2k+1$ nodes has the following properties:

1. It includes k operations.

2. It has $k+1$ leaves.

An easy way to add weights is to directly add weights to each leaf node, as shown in Figure 4.2(b). So given a tree structure $g \in \mathcal{G}_{2k+1}$, we can define $\mathcal{W}(g)$ as the set of all weighted labeled binary trees given g, with constraint $\|\boldsymbol{w}\|_1 \leq 1$. Additionally, we define

$$\mathcal{W}_{2k+1} = \bigcup_{g \in \mathcal{G}_{2k+1}} \mathcal{W}(g). \tag{4.1}$$

For a fixed $g \in \mathcal{G}_{2k+1}$, any $h \in \mathcal{W}(g)$ can be rewritten as a summation of some basis functions and some productions of basis functions. For instance, given $\boldsymbol{w}$ and the labeled binary tree structure g_0 in Figure 4.2(a), Figure 4.2(b) represents a function $h(x; g_0, \boldsymbol{w}) = (w_1\phi_1(x_2) + w_2\phi_3(x_1)) * (w_3\phi_3(x_2) + w_4\phi_1(x_3))$, and it is the summation of 4 interactions $w_1w_3\phi_1(x_2)\phi_3(x_2)$, $w_1w_4\phi_1(x_2)\phi_1(x_3)$, $w_2w_3\phi_3(x_1))\phi_3(x_2)$, and $w_2w_4\phi_3(x_1)\phi_1(x_3)$. Equivalently, $h(x; g_0, \boldsymbol{w}) = \langle \boldsymbol{v}, \mathbf{u} \rangle$, where $\boldsymbol{v} = \psi^v_{g_0}(\boldsymbol{w}) = (w_1w_3, w_1w_4, w_2w_3, w_2w_4)$ and

$$\mathbf{u} = \psi^u_{g_0}(\boldsymbol{x}) = (\phi_1(x_2)\phi_3(x_2), \phi_1(x_2)\phi_1(x_3), \phi_3(x_1)\phi_3(x_2), \phi_3(x_1)\phi_1(x_3).$$

Similarly, for any labeled binary tree g, we could write $h = h(x; g, \boldsymbol{w}) \in \mathcal{W}(g)$ as an inner product of two vectors $\boldsymbol{v}$ and $\mathbf{u}$:

$$h(x; g, \boldsymbol{w}) = \langle \boldsymbol{v}, \mathbf{u} \rangle, \quad \boldsymbol{v} = \psi_g^v(\boldsymbol{w}), \quad \mathbf{u} = \psi_g^u(\boldsymbol{x}), \tag{4.2}$$

where the transformation function ψ_g^v and ψ_g^u depend on g. Define the length of the vector $\boldsymbol{v}$ and $\mathbf{u}$ as M_g, and M_g also depends on g. Define

$$M_{2k+1} = \max_{g \in \mathcal{G}_{2k+1}} M_g. \tag{4.3}$$

Lemma 13 *If* $\|\boldsymbol{w}\|_1 \leq 1$ *and* $\|\phi_i\|_\infty \leq 1 \; \forall i$*, regardless of* g*, we always have* $\|\boldsymbol{v}\|_1 \leq 1$ *and* $\|\mathbf{u}\|_\infty \leq 1$.

4.3 Sufficient Number of Samples

In this section, we provide a generalization bound that holds for any data distribution and any labeled binary tree. This not only implies the sufficient number of samples to recover a labeled binary tree from a given dataset, but also guarantees that the empirical risk (i.e., the risk with respect to a training set) is a consistent estimator of the true risk (i.e., the risk with respect to the data distribution). We first bound the size of $\mathcal{G}_{2k+1}$, and then show a Rademacher-based uniform convergence guarantee.

Properties of the Labeled Binary Tree Set. Let $|\mathcal{G}_{2k+1}|$ denote the size of $\mathcal{G}_{2k+1}$: the labeled binary tree set with no more than $2k+1$ nodes. The lemma below gives the upper bound of the size of the functional space, which will be used later to show the uniform convergence.

Lemma 14 *For* $k \geq 1$*, we have* $|\mathcal{G}_{2k+1}| \leq 4k(k)!(m_1 q)^{k+1}$.

The lemma below gives the upper bound of M_{2k+1}, which is used to later to bound the Rademacher complexity. Remind that M_{2k+1} is defined in equation 4.3.

Lemma 15 $M_{2k+1} < (1.45)^{k+1}$.

Rademacher-based Uniform Convergence. Next, we present our first main theorem, which guarantees a uniform convergence of the empirical risk to the true risk, regardless of the tree structure and weights.

Assume that $d : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ is a 1-Lipschitz function related to the prediction problem. For regression, we assume $\mathcal{Y} = \mathbb{R}$, and $d(y, y') = \min(1, (y - y')^2/2)$, while for classification, we assume $\mathcal{Y} = \{-1, 1\}$, and $d(y, y^{'}) = \min(1, \max(0, 1 - yy^{'}))$. Let $z = (x, y) \in \mathcal{Z}$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Furthermore, let $\mathcal{H}(g) = \{h(z) = d(y, f(x)), f \in \mathcal{W}(g)\}$ for a fixed labeled binary tree g. Let $\mathcal{H}_{2k+1}$ be a hypothesis class satisfying

$$\mathcal{H}_{2k+1} = \bigcup_{g \in \mathcal{G}_{2k+1}} \mathcal{H}(g).$$

For every $h \in \mathcal{H}(g)$, we define the true and empirical risks as

$$\mathbb{E}_{\mathcal{D}}[h] = \mathbb{E}_{z \sim \mathcal{D}}[h(z)], \quad \widehat{\mathbb{E}}_S[h] = \frac{1}{n} \sum_{i=1}^{n} h(z_i). \tag{4.4}$$

Next, we state our generalization bound that shows that $O(k \log(m_1 q) + \log k!)$ samples are sufficient for learning.

Theorem 4.3.1 *Let $z = (x, y)$ be a random variable of support $\mathcal{Z}$ and distribution $\mathcal{D}$. Let $S = \{z_1 \dots z_n\}$ be a dataset of n i.i.d. samples drawn from $\mathcal{D}$. Fix $\delta \in (0, 1)$. With probability at least $1 - \delta$ over the choice of S, we have:*

$$(\forall g \in \mathcal{G}_{2k+1}, \forall h \in \mathcal{H}(g))$$
$$\mathbb{E}_{\mathcal{D}}[h] \leq \widehat{\mathbb{E}}_S[h] + 2\sqrt{\frac{k+1}{n}} + \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}}$$

Proof Given a function $h : \mathcal{Z}^n \to \mathbb{R}$, we define $\mathbb{E}_S[h(S)] = \mathbb{E}_{S \sim \mathcal{D}^n}[h(S)]$. The function $\varphi_g(S) = \sup_{h \in \mathcal{H}(g)} \left(\mathbb{E}_{\mathcal{D}}[h] - \widehat{\mathbb{E}}_S[h] \right)$ fulfills the condition in McDiarmid's inequality and $\mathcal{H}(g) \subseteq \{h | h : \mathcal{Z} \to [0, 1]\}$, by Lemma 16, therefore $\mathbb{P}[\varphi_g(S) - \mathbb{E}_S[\varphi_g(S)] \geq \varepsilon] \leq \exp\left(\frac{-2\varepsilon^2}{\sum_{\mathrm{i}=1}^{\mathrm{n}} (1/\mathrm{n})^2}\right) = \exp(-2\mathrm{n}\varepsilon^2)$. Furthermore, by applying the union bound for all $g \in \mathcal{G}_{2k+1}$, by Lemma 14, and by Hoeffding's inequality, we have:

$$\mathbb{P}[(\exists g \in \mathcal{G}_{2k+1}), \varphi_g(S) - \mathbb{E}_S[\varphi_g(S)] \geq \varepsilon]] \leq \sum_{g \in \mathcal{G}_{2k+1}} \mathbb{P}[\varphi_g(S) - \mathbb{E}_S[\varphi_g(S)] \geq \varepsilon]$$
$$\leq 2|\mathcal{G}_{2k+1}| e^{-2n\epsilon^2} \leq 8k(k)!(m_1 q)^{k+1} e^{-2n\epsilon^2}$$

Equivalently, $\mathbb{P}[(\forall g \in \mathcal{G}_{2k+1}), \varphi_g(S) - \mathbb{E}_S[\varphi_g(S)] \leq \varepsilon]] \geq 1 - 8k(k)!(m_1 q)^{k+1} e^{-2n\epsilon^2}$.

Setting $8k(k)!(m_1 q)^{k+1} e^{-2n\epsilon^2} = \delta$, we get $\varepsilon = \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}}$. Thus:

$$\begin{aligned}\mathbb{P}\left[(\forall g \in \mathcal{G}_{2k+1}), \varphi_g(S) < \mathbb{E}_S[\varphi_g(S)] + \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}}\right] \\ \geq 1 - \delta \end{aligned} \tag{4.5}$$

Note that by the definition of the supremum, by the definition of the function $\varphi_g : \mathcal{Z}^n \to \mathbb{R}$, and by Equation (4.5), with probability at least $1 - \delta$, simultaneously for all $g \in \mathcal{G}_{2k+1}$ and $h \in \mathcal{H}(g)$

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[h] - \widehat{\mathbb{E}}_S[h] &\leq \sup_{h \in \mathcal{H}(g)} \left(\mathbb{E}_{\mathcal{D}}[h] - \widehat{\mathbb{E}}_S[h]\right) \\ &= \varphi_g(S) \\ &< \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}} + \mathbb{E}_S[\varphi_g(S)] \end{aligned} \tag{4.6}$$

The next step is to bound $\mathbb{E}_S[\varphi_g(S)]$ in Equation (4.6) in terms of the Rademacher complexity of $\mathcal{W}(g)$. By the definition of φ_g, by the ghost sample technique, the Ledoux-Talagrand Contraction Lemma, we can show that

$$\mathbb{E}_S[\varphi_g(S)] = 2\mathfrak{R}_n(\mathcal{H}(g)) \leq 2\mathfrak{R}_n(\mathcal{W}(g))$$

The final step is to bound $\mathfrak{R}_n(\mathcal{W}(g))$, and it is sufficient to bound $\hat{\mathfrak{R}}_S(\mathcal{W}(g))$ for any $g \in \mathcal{G}_{2k+1}$. Then for a fixed $g \in \mathcal{G}_{2k+1}$, any $f \in \mathcal{W}(g)$ can be rewritten as a summation of no more than $[(1.45)^{k+1}]$ productions of basis functions, where $[m]$ denotes that largest integer smaller than or equal to m according to Lemma 15. We could decompose $h = h(\boldsymbol{x}; g, \boldsymbol{w})$ as in Equation (4.2), thus $h = h(\boldsymbol{x}; g, \boldsymbol{w}) = \langle \boldsymbol{v}, \mathbf{u} \rangle$, where $||\boldsymbol{v}||_1 \leq 1$ and $||\mathbf{u}||_\infty \leq 1$ by Lemma 13. By using a technique similar to [65] for linear prediction, we have

$$\begin{aligned}\hat{\mathfrak{R}}_S(\mathcal{W}(g)) &= \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{W}(g)} \left(\frac{1}{n}\sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}^i)\right)\right] \\ &= \mathbb{E}_\sigma \left[\sup_{\|\boldsymbol{w}\|_1 \leq 1} \left(\frac{1}{n}\sum_{i=1}^{n} \sigma_i h(\boldsymbol{x}^{(i)}; g, \boldsymbol{w})\right)\right] \end{aligned}$$

$$
\begin{aligned}
&\leq \frac{1}{n}\ \mathbb{E}_\sigma\left[\sup_{\|\boldsymbol{v}\|_1\leq 1}\left(\sum_{i=1}^{n}\sigma_i\langle \boldsymbol{v}, \mathbf{u}^{(i)}\rangle\right)\right]\\
&= \frac{1}{n}\ \mathbb{E}_\sigma\left[\sup_{\|\boldsymbol{v}\|_1\leq 1}\left\langle \boldsymbol{v}, \textstyle\sum_{i=1}^{n}\sigma_i\mathbf{u}^{(i)}\right\rangle\right]\\
&= \frac{\|\boldsymbol{v}\|_1}{n}\ \mathbb{E}_\sigma\left[\left\|\textstyle\sum_{i=1}^{n}\sigma_i\mathbf{u}^{(i)}\right\|_\infty\right]\\
&= \frac{1}{n}\ \mathbb{E}_\sigma\left[\sup_j \textstyle\sum_{i=1}^{n}\sigma_i[\mathbf{u}^{(i)}]_j\right]\\
&= \frac{\sqrt{2\log M_{2k+1}}}{n}\ \sup_j\sqrt{\textstyle\sum_{i=1}^{n}[\mathbf{u}^{(i)}]_j^2}\\
&\leq \frac{\sqrt{2\log M_{2k+1}}}{n}\ \sqrt{n\,\|\mathbf{u}\|_\infty^2}\\
&\leq \sqrt{\frac{2\log M_{2k+1}}{n}}\\
&\leq \sqrt{\frac{2(k+1)\log 1.45}{n}}\\
&< \sqrt{\frac{k+1}{n}}
\end{aligned}
$$

Finally, we have $\mathfrak{R}_n(\mathcal{W}(g)) = \mathbb{E}_{S\sim\mathcal{D}^n}[\hat{\mathfrak{R}}_S(\mathcal{W}(g))] < \sqrt{\frac{k+1}{n}}$ ■

Corollary 3 *Define* $\hat{h} = \underset{h\in\mathcal{H}_{2k+1}}{\operatorname{argmin}}\,\widehat{\mathbb{E}}_S[h]$, *and* $\bar{h} = \underset{h\in\mathcal{H}_{2k+1}}{\operatorname{argmin}}\,\mathbb{E}_{\mathcal{D}}[h]$. *Then under the same setting of Theorem 4.3.1, fix* $\delta\in(0,1)$. *With probability at least* $1-2\delta$ *over the choice of* S, *we have:*

$$\mathbb{E}_{\mathcal{D}}[\hat{h}] - \mathbb{E}_{\mathcal{D}}[\bar{h}] \leq 2\sqrt{\frac{k+1}{n}} + \sqrt{\frac{\log(1/\delta)}{2n}} + \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}}$$

Proof By Theorem 4.3.1, with probability at least $1-\delta$ over the choice of S,

$$\mathbb{E}_{\mathcal{D}}[\hat{h}] \leq \widehat{\mathbb{E}}_S[\hat{h}] + 2\sqrt{\frac{k+1}{n}} + \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}}$$

By Hoeffding's inequality, with probability at least $1-\delta$ over the choice of S,

$$\widehat{\mathbb{E}}_S[\bar{h}] - \mathbb{E}_{\mathcal{D}}[\bar{h}] \leq \sqrt{\frac{\log(1/\delta)}{2n}}$$

Since $\hat{h}$ minimizes $\widehat{\mathbb{E}}_S[h]$, $\widehat{\mathbb{E}}_S[\hat{h}] \leq \hat{\mathbb{E}}_S[\bar{h}]$. With probability at least $1 - 2\delta$ over the choice of S,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[\hat{h}] - \mathbb{E}_{\mathcal{D}}[\bar{h}] &= \mathbb{E}_{\mathcal{D}}[\hat{h}] - \widehat{\mathbb{E}}_S[\bar{h}] + \widehat{\mathbb{E}}_S[\bar{h}] - \mathbb{E}_{\mathcal{D}}[\bar{h}] \\
&\leq \mathbb{E}_{\mathcal{D}}[\hat{h}] - \widehat{\mathbb{E}}_S[\hat{h}] + \widehat{\mathbb{E}}_S[\bar{h}] - \mathbb{E}_{\mathcal{D}}[\bar{h}] \\
&\leq 2\sqrt{\frac{k+1}{n}} + \sqrt{\frac{\log(1/\delta)}{2n}} + \sqrt{\frac{(k+1)\log m_1 q + \log 8k(k)! + \log(1/\delta)}{2n}}
\end{aligned}$$

■

Next, we present a useful remark in the nonparametric setting, where both k and q are allowed to grow as a function of n.

Remark 7 *If $k \in O(\min(n^{1/2-\epsilon}, \frac{n^{1-2\epsilon}}{\log m_1}))$, $q \in O(e^{n^{1/2-\epsilon}})$ for any $\epsilon \in (0, 1/2)$, then the generalization error in Theorem 4.3.1 could be uniformly bounded by $O(n^{-\epsilon})$.*

4.4 Necessary Number of Samples

In this section, we analyze the necessary number of samples to recover a labeled binary tree from a given dataset. To show the necessary number of samples, we restrict the operation to multiplications only, and consider unit weights. Note that the necessary number of samples in restricted ensembles yields a lower bound for the original problem. The use of restricted ensembles is customary for information-theoretic lower bounds [66, 67]. We utilize Fano's inequality as the main proof technique.

We construct a restricted ensemble as follows. Define a sequence of basis functions $\phi_i(z) = \sqrt{2}\cos(i\pi z)$, where $z \in [-1, 1]$ for $i = 1, \ldots, q$. Furthermore, let

$$\boldsymbol{x}_i \sim Unif[-1, 1]^{m_1}, \epsilon_i \sim N(0, \sigma_\epsilon^2).$$

Let $S = \{(\boldsymbol{x}_i, z_i) : z_i = f(\boldsymbol{x}_i) + \epsilon_i, i = 1, \ldots, n\}$, and $S' = \{(\boldsymbol{x}_i, y_i) : y_i = t(z_i), i = 1, \ldots, n\}$, where $t : \mathbb{R} \to \mathcal{Y}$ is a fixed function related to the prediction problem, as introduced in Section 2. This defines a Markov chain $f \to S \to S' \to \hat{f}$. To apply Fano's inequality, we need to further bound the mutual information $\mathbb{I}(f, S')$ by

a sum of Kullback-Leibler (KL) divergences of the form $KL(P_{\boldsymbol{x},y|f_i}|P_{\boldsymbol{x},y|f_i'})$ where f_i and f_i' are two different compositional trees. Consider a labeled binary tree subspace $\mathcal{F}^*_{2k+1}$ of $\mathcal{F}_{2k+1}$, where we only allow for multiplication nodes (i.e., additions are not allowed) and where each covariate x_j of the independent variable $\boldsymbol{x}$ is used only once. Furthermore, we consider a restricted ensemble with unit weights. Equivalently,

$$\mathcal{F}^*_{2k+1} = \{f_{\mathcal{A}}(\boldsymbol{x}) = \prod_{(i,j)\in\mathcal{A}} \phi_i(x_j) : \mathcal{A} \subseteq \{1,\dots,q\} \times \{1,\dots,p\},$$
$$|\mathcal{A}| \leq k+1, \forall (i,j) \in \mathcal{A},\ l \neq i \Rightarrow (l,j) \notin \mathcal{A}\}.$$

Let $c = |\mathcal{F}_{2k+1}| = \sum_{i=1}^{k} q^{i+1}\binom{p}{i+1}$.

Next, we state our information-theoretic lower bound that shows that $\Omega(k\log(m_1 q) - \log k!)$ samples are necessary for learning.

Theorem 4.4.1 *Assume nature uniformly picks a true hypothesis $\bar{f}$ from $\mathcal{F}_{2k+1}$. For any estimator $\hat{f}$, if*

$$n \leq (\log(q^{k+1}\binom{p}{k+1}) - 2\log 2)\sigma_\epsilon^2/2,$$

then $\mathbb{P}[\hat{f} \neq \bar{f}] \geq \frac{1}{2}$.

Proof Any $f_{\mathcal{A}} \in \mathcal{F}_{2k+1}$ can be decomposed by the dimension of x:

$$f_{\mathcal{A}}(\boldsymbol{x}) = \prod_{j=1}^{p} f_j^{\mathcal{A}}(x_j),$$

where $f_j^{\mathcal{A}} = \phi_{i_j}$ if $\exists (i_j, j) \in \mathcal{A}$, and $f_j^{\mathcal{A}} \equiv 1$ if $(i,j) \notin \mathcal{A}$ for any i. In addition, $\int_{-1}^{1} \frac{1}{2}\phi_i(x)dx = 0$ and $\langle \phi_i, \phi_{i'} \rangle = \int_{-1}^{1} \frac{1}{2}\phi_i(x)\phi_{i'}(x)dx = I(i = i')$. Thus,

$$\begin{aligned}
\langle f_{\mathcal{A}}, f_{\mathcal{A}'} \rangle &= \int_{-1}^{1} \cdots \int_{-1}^{1} \frac{1}{2^{m_1}} f_j^{\mathcal{A}}(x_j) f_j^{\mathcal{A}'}(x_j) dx_1 \cdots dx_{m_1} \\
&= \prod_{j=1}^{m_1} \int_{-1}^{1} \frac{1}{2} f_j^{\mathcal{A}}(x_j) f_j^{\mathcal{A}'}(x_j) dx_j \\
&= \prod_{j=1}^{m_1} I(f_j^{\mathcal{A}} = f_j^{\mathcal{A}'}) \\
&= I(f_{\mathcal{A}} = f_{\mathcal{A}'})
\end{aligned}$$

Furthermore,

$$\begin{aligned} ||f_{\mathcal{A}} - f_{\mathcal{A}'}||^2 &= \langle f_{\mathcal{A}}, f_{\mathcal{A}}\rangle + \langle f_{\mathcal{A}'}, f_{\mathcal{A}'}\rangle - 2\langle f_{\mathcal{A}}, f_{\mathcal{A}'}\rangle \\ &= 2I(f_{\mathcal{A}} = f_{\mathcal{A}'}) \end{aligned} \tag{4.9}$$

By the data processing inequality [68] in the Markov chain $f \to S \to S' \to \hat{f}$, and since the mutual information can be bounded by a pairwise KL bound [69], we have

$$\begin{aligned} \mathbb{I}(\bar{f}, S^{'}) &\leq \mathbb{I}(\bar{f}, S) \\ &\leq \frac{1}{c^2}\sum_{\mathcal{A}}\sum_{\mathcal{A}'} KL(P_{S|f_{\mathcal{A}}}|P_{S|f_{\mathcal{A}'}}) \\ &= \frac{n}{c^2}\sum_{\mathcal{A}}\sum_{\mathcal{A}'} KL(P_{\boldsymbol{x},y|f_{\mathcal{A}}}|P_{\boldsymbol{x},y|f_{\mathcal{A}'}}) \\ &= \frac{n}{c^2}\sum_{\mathcal{A}}\sum_{\mathcal{A}'} KL(\mathcal{N}(f_{\mathcal{A}}, \sigma_\epsilon^2)|\mathcal{N}(f_{\mathcal{A}'}, \sigma_\epsilon^2)) \\ &= \frac{n}{c^2}\sum_{\mathcal{A}}\sum_{\mathcal{A}'} \frac{||f_{\mathcal{A}} - f_{\mathcal{A}'}||^2}{2\sigma_\epsilon^2} \\ &\leq \frac{n}{c^2} * c^2 * \frac{2}{2\sigma_\epsilon^2} \\ &= \frac{n}{\sigma_\epsilon^2} \end{aligned}$$

By the Fano's inequality [68] on the Markov chain $f \to S \to S^{'} \to \hat{f}$, we have

$$\mathbb{P}[\hat{f} \neq \bar{f}] \geq 1 - \frac{\mathbb{I}(\bar{f}, S^{'}) + \log 2}{\log c} \geq 1 - \frac{n/\sigma_\epsilon^2 + \log 2}{\log c}$$

By making

$$\frac{1}{2} = \mathbb{P}[\hat{f} \neq \bar{f}] \geq 1 - \frac{n/\sigma_\epsilon^2 + \log 2}{\log c},$$

we have

$$n \leq (\log c - 2\log 2)\sigma_\epsilon^2/2$$

Since $c \geq q^{k+1}\binom{p}{k+1}$, $n \leq (\log(q^{k+1}\binom{p}{k+1}) - 2\log 2)\sigma_\epsilon^2/2$ implies $\mathbb{P}[\hat{f} \neq \bar{f}] \geq \frac{1}{2}$. If $p \gg k$, the above is equivalent to

$$\begin{aligned} n &= \Omega\left(\frac{\sigma_\epsilon^2}{2}(\log[q^{k+1}p^{k+1}/(k+1)!] - 2\log 2)\right) \\ &\in \Omega\left((k+1)\log(m_1 q) - \log(k+1)!\right) \end{aligned}$$

■

Corollary 4 *Assume nature uniformly picks a true function $\bar{f}$ from $\mathcal{F}_{2k+1}$. For each $f \in \mathcal{F}_{2k+1}$, define a corresponding $h(\boldsymbol{x}, y) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$. The corresponding true hypothesis is $\bar{h} = \bar{h}(\boldsymbol{x}, y) = \frac{1}{2}(y - \bar{f}(\boldsymbol{x}))^2$. Let*

$$\mathcal{H}_{2k+1} = \{h(\boldsymbol{x}, y) = \frac{1}{2}(y - f(\boldsymbol{x}))^2, f \in \mathcal{F}_{2k+1}\}.$$

For any estimator $\hat{h} = \hat{h}(\boldsymbol{x}, y) = \frac{1}{2}(y - \hat{f}(\boldsymbol{x}))^2$, if

$$n \leq (\log(q^{k+1}\binom{p}{k+1}) - 2\log 2)\sigma_\epsilon^2/2,$$

then

$$\mathbb{E}_{\mathcal{D}}[\hat{h}] - E_{\mathcal{D}}[\bar{h}] \geq 1$$

with probability at least $\frac{1}{2}$.

Proof $\bar{f}$ is the true function, so $y = \bar{f}(\boldsymbol{x}) + \epsilon$, where $\epsilon \sim N(0, \sigma_\epsilon^2)$. Recall that by Theorem 2, if

$$n \leq (\log(q^{k+1}\binom{p}{k+1}) - 2\log 2)\sigma_\epsilon^2/2,$$

then $P[\bar{f} \neq \hat{f}] \geq 1/2$. Thus, assuming that $\bar{f} \neq \hat{f}$, we have

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[\hat{h}] - E_{\mathcal{D}}[\bar{h}] &= \frac{1}{2}\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[(y - \hat{f}(\boldsymbol{x}))^2 - (y - \bar{f}(\boldsymbol{x}))^2] \\
&= \frac{1}{2}\mathbb{E}_{\substack{\boldsymbol{x}\sim Unif[-1,1]^{m_1} \\ \epsilon\sim N(0,\sigma_\epsilon^2)}}[(\bar{f}(\boldsymbol{x}) + \epsilon - \hat{f}(\boldsymbol{x}))^2 - \epsilon^2] \\
&= \frac{1}{2}\mathbb{E}_{\boldsymbol{x},\epsilon}[(\bar{f}(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}))^2 + 2\epsilon(\bar{f}(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}))] \\
&= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}}[(\bar{f}(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}))^2] + \\
&\quad \mathbb{E}_{\epsilon}[\epsilon] * \mathbb{E}_{\boldsymbol{x}}[(\bar{f}(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}))] \\
&= \frac{1}{2}||\bar{f} - \hat{f}||^2 \\
&= \frac{1}{2} * 2I(\bar{f} \neq \hat{f}) \\
&= 1
\end{aligned}$$

■

Remark 8 *Excess risk measures how well the empirical risk minimizer performs when compared to the best candidate in the hypothesis class. On the one hand, Corollary 3 discusses the upper bound of the excess risk, and indicates that the sufficient sample complexity is* $O(k\log(m_1q)+\log k!)$. *On the other hand, Corollary 4 discusses the lower bound of the excess risk, and shows that the necessary sample complexity is* $\Omega(k\log(m_1q)-\log k!)$. *Especially when* $k \ll m_1q$, *both the sufficient sample complexity and necessary sample complexity are* $\Theta(k\log(m_1q))$.

4.5 Greedy Search Algorithm for Regression

In this section, we propose a greedy search algorithm to recover a weighted labeled binary tree for regression. As mentioned in Section 3.2, for regression, we define $d(y,y') = \min(1,(y-y')^2/2)$. For simplicity, we assume $\mathcal{Y} = [-1,1]$, thus $d(y,y') = (y-y')^2/2$. Consequently, we have $\mathcal{H}(g) = \{h(z) = h(x,y) = (y-f(x))^2/2, f \in \mathcal{W}(g)\}$ for a fixed labeled binary tree g. The true risk and the empirical risk are defined as $\mathbb{E}_{\mathcal{D}}[h] = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[(y-f(\boldsymbol{x}))^2/2]$, and $\hat{\mathbb{E}}_S[h] = \sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2/2$.

Based on Theorem 4.3.1, it is straightforward to have a brute-force algorithm to traverse all possible trees in $\mathcal{G}_{2k+1}$, and to compute the best weights for each tree. Theorem 4.3.1 could guarantee that the risk at the empirical risk minimizer is close to the minimum possible risk over all functions in $\mathcal{W}_{2k+1}$, given enough training samples. However the space of trees grows exponentially with the number of nodes, as shown in Lemma 14, and therefore the brute-force algorithm is exponential-time.

After decades of work, the literature in tensor decomposition has still failed to provide polynomial-time algorithms with guarantees, for a general nonsymmetric tensor decomposition problem. In general, it has been shown that most tensor problems are NP-hard [70]. Therefore most existing literature considers a specific tensor structure like the symmetric orthogonal decomposition [71]. As shown in Figure 4.1(b), we can model the tensor decomposition problem in our framework, for a fixed tree. However

in our problem, we learn the tree structure. Thus, our problem is harder than tensor decomposition.

Given the above, we propose a greedy search algorithm for learning the structure of *predictor functions*. A greedy approach was also taken in [60] for learning the structure of *kernels*. Before we proceed, note that the uniform convergence of the empirical risk to the true risk holds for any $h \in \mathcal{H}_{2k+1}$ and therefore, it applies to the greedy algorithm output, which is an element of $\mathcal{H}_{2k+1}$.

Our algorithm begins by applying all basis functions to all input dimensions, and picking the one that minimizes $\sum_{m=1}^{n}(y_m - w'\phi_{i'}(x_{j'}))^2/2$ among all function indices $i' \in \{1, \ldots, q\}$ and coordinates $j' \in \{1, \ldots, p\}$, where w' is estimated separately for each candidate option (i', j'). This produces a tree with a single node. After this, we repeat the following search operators over the leaves of the current tree: Any leaf $\mathcal{V}$ can be replaced with $\mathcal{V} + \mathcal{V}'$, or $\mathcal{V} * \mathcal{V}'$, where $\mathcal{V}' = w'\phi_{i'}(x_{j'})$.

Our algorithm searches over the space of trees using a greedy search approach. At each stage, we evaluate the replacement of every leaf by either a summation or multiplication, and compute the weight for the new candidate leaf while fixing all the other weights. Then we take the search operation with the lowest score among all leaves, and adjust all weights by coordinate descent at each iteration, as shown in Algorithm 3.

Computing the Weight. A main step in our main algorithm is the computation of the weight of a new candidate leaf, while fixing all the other weights. Fortunately, computing the new weight turns out to be a simple least square problem, but involves traversing the tree from the root to the candidate node being evaluated, as shown in Algorithm 4.

Computational Complexity. Next, we analyze the time complexity of our method. In iteration D, we solve $O(m_1 qD)$ single-dimensional closed-form optimization problems: for all the D tree leaves, our algorithm tries to insert a new node with either "+" or "*", all q basis functions, and all m_1 dimensions of $\boldsymbol{x}$. In addition, it takes $O(nD)$ time to compute the optimal weight (in closed-form) for a specific

basis function of a specific dimension of $\boldsymbol{x}$ at a specific insert position on a dataset of size n. Finally, it takes $O(nD)$ to adaptively update all weights at each step by coordinate descent. The computational complexity of our algorithm for k iterations is thus $O(m_1qn(1^2+2^2+\cdots+k^2)) \in O(m_1qnk^3)$. This can be reduced by processing the tree leaves (or alternatively, batches of data samples) in parallel.

4.6 Experiments

In this section, we demonstrate our theorem in four simulation experiments, and then apply our methods to real world problems.

We use a function $f(\boldsymbol{x}) = 0.3sin(3\pi x_1)cos(2\pi x_2) + 0.4x_3^2 - 0.3x_4$, and noise standard deviation $\sigma = 0.05$. Our choice of the set of basis functions Φ include B-spline of degree 1, Fourier basis functions: $\{\sin(i\pi x), \cos(i\pi x)\}_{i=1,\ldots,\infty}$ and truncated polynomials: $\{x, x^2, x^3, (x-t)_+^3, t \in \mathbb{R}\}$, where $(x)_+ = max(x, 0)$. We designed four different experiments to demonstrate our theoretical contributions. For each setting, the generalization error is estimated by the mean of 20 repeated trials in order to show error bars at 95% confidence level.

Experiment 1. We set the dimension of the explanatory variables $p = 100$, the number of basis functions: $q = 40$, and the number of iterations $k = 10$. For each value of $n \in \{50, 100, 150, 200, 250\}$, we sampled n random samples $\boldsymbol{x}_i$, $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, $i = 1, \cdots, n$ for training, and $n/3$ samples for testing. In Figure 4.3, we observe that the generalization error has a sharp decline when n increases from 50 to 100, and a slower decline for higher values of n. This demonstrates that the generalization error $\propto \sqrt{\frac{1}{n}}$ as prescribed by Theorem 4.3.1.

Experiment 2. We set the sample size $n = 250$, the number of basis functions: $q = 40$, and the number of iterations $k = 10$. For each value of $p \in \{10, 20, 50, 100, 200\}$, we sampled 250 $p - dimensional$ random samples $\boldsymbol{x}_i$, $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, $i = 1, \cdots, n$ for training, and 83 samples for testing. Figure 4.4 shows that the generalization

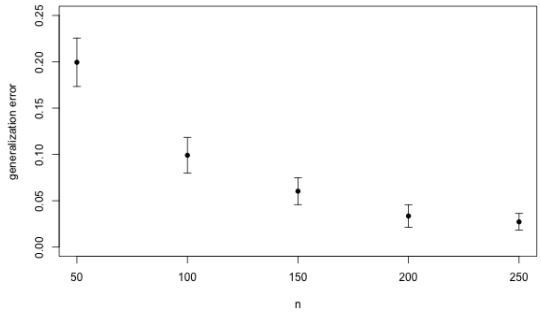


Fig. 4.3.: Generalization error vs. sample size n.

error grows rapidly when $p \in (0, 50)$, and the growth slows down as m_1 increases. This finding matches the conclusion of Theorem 4.3.1 that the generalization error $\propto \sqrt{\log m_1}$.

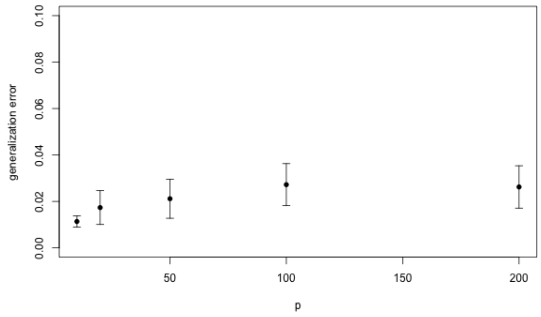


Fig. 4.4.: Generalization error vs. dimension of the explanatory variable m_1.

Experiment 3. We set the dimension of the explanatory variables $p = 100$, the number of basis functions: $q = 40$, and the sample size $n = 250$. For each value of the number of iterations $k \in \{1, 5, 10, 20\}$, we sampled 250 random samples $\boldsymbol{x}_i$, $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, $i = 1, \cdots, n$ for training, and 83 samples for testing. As shown in Figure 4.5, the generalization error grows almost linearly as k increases when k is

small, but the growth rate decreases apparently when $k > 15$. This is consistent with the theoretical result that the generalization error $\propto \sqrt{k}$.

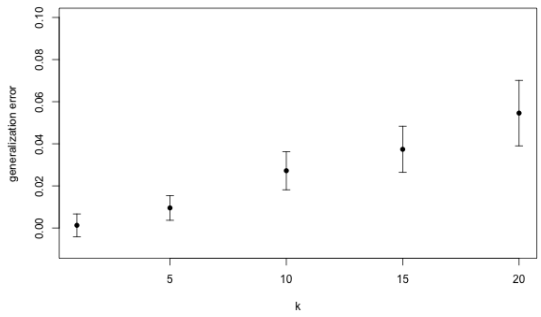


Fig. 4.5.: Generalization error vs. number of iterations k.

Experiment 4. We set the dimension of the explanatory variables $p = 20$, the sample size $n = 250$, and the number of iterations $k = 10$. For each value of $q \in \{10, 20, 50, 100\}$, we sampled 250 random samples $\boldsymbol{x}_i$, $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, $i = 1, \cdots, n$ for training, and 83 samples for testing. Figure 4.6 indicates that the generalization error grows rapidly when q is small, and the growth slows down as q continue to increase. This matches the conclusion of Theorem 4.3.1 that the generalization error $\propto \sqrt{\log q}$.

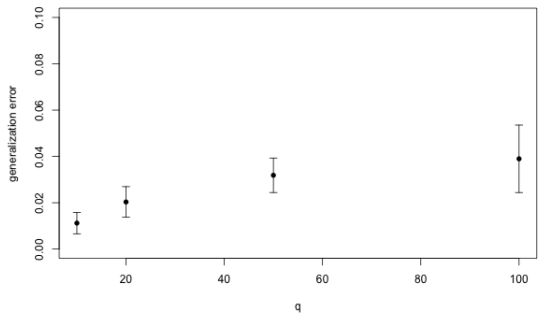


Fig. 4.6.: Generalization error vs. number of basis functions q.

Our methods are comparative to methods like *Gaussian processes* for two real-world data sets, although our model sizes are much smaller.

Airline Delays. For real-world experiments, we evaluate our algorithm on the US flight dataset. We use a subset of the data with flight arrival and departure times for commercial flights in 2008. The dataset is publicly available at http://stat-computing.org/dataexpo/2009/. The flight delay is the response variable, which is predicted by using the following variables: the age of the aircraft, distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month, and month. We randomly select 800,000 datapoints, using a random subset of 700,000 samples to train the model and 100,000 to test it. Although our method uses only $k = 10$ (i.e., $2k + 1 = 21$ nodes, or $k + 1 = 11$ functions of features), we obtain a test RMSE of 34.89. For comparison, the authors in [72] also randomly selected 800,000 samples (700,000 for training, 100,000 for testing) and obtained an RMSE between 32.6 and 33.5 with 1200 iterations on a *Gaussian processes* approach. In general, Gaussian processes predict the output by memorization of the 700,000 training points. Our tree depends only on evaluating $k+1 = 11$ functions of features. When predicting, our tree does not need to remember the training set.

World Weather. The world weather dataset contains monthly measurements of temperature, precipitation, vapor, cloud cover, wet days and frost days from Jan 1990 to Dec 2002 (156 months) on a 5×5 degree grid that covers the entire world. The dataset is publicly available at http://www.cru.uea.ac.uk/. The response variable is temperature. We use 19,000 samples for training, 8000 samples for testing, and run 30 iterations. Although our method uses only $k = 30$ (i.e., $2k + 1 = 61$ nodes, or $k+1 = 31$ functions of features), we obtain a test RMSE of 1.319. Gaussian processes obtained a test RMSE of 1.23. Since the standard deviation of the output variable is 16.98, both our method and Gaussian processes obtain a coefficient of determination of 0.99.

4.7 Concluding Remarks

There are several ways of extending this research. While we focused on the sample complexity for trees of predictor functions, it would be interesting to analyze trees of kernels as well, as many popular kernel structures [60] are equivalent to a labeled binary tree. Additionally, while we focused on learning trees, it would be interesting to propose methods for learning general directed acyclic graphs.

4.8 Detailed Proofs

4.8.1 Proof for Lemma 13

Proof We first show $||\mathbf{u}||_\infty \leq 1$:

For any production of finite basis functions from Φ,

$$||\prod_{1=1}^{L} \phi_{i_l}(x_{j_l})||_\infty \leq \prod_{1=1}^{L} ||\phi_{i_l}(x_{j_l})||_\infty \leq 1$$

Each component of $\mathbf{u}$ is a production of finite basis functions from Φ. Thus $||\mathbf{u}||_\infty \leq 1$.

Then we show $||\boldsymbol{v}||_1 \leq ||\boldsymbol{w}||_1$ if $||\boldsymbol{w}||_1 \leq 1$ by induction:

$k = 0$, $||v||_1 = ||w||_1$;

Assume that for any $k < K$ and any weighted labeled binary tree $h \in \mathcal{W}_{2k+1}$, $||v_h||_1 \leq ||w_h||_1$. For $k = K$, decompose the tree $h(\boldsymbol{x}; g, \boldsymbol{w}) \in \mathcal{W}_{2K+1}$ by the left subtree $h_l(\boldsymbol{x}; g_l, \boldsymbol{w}_l) = \langle \boldsymbol{v}_l, \mathbf{u}_l \rangle$ and the right subtree as $h_r(\boldsymbol{x}; g_r, \boldsymbol{w}_r) = \langle \boldsymbol{v}_r, \mathbf{u}_r \rangle$.

If the root is a "+", then $||\boldsymbol{v}||_1 = ||\boldsymbol{v}_l||_1 + ||\boldsymbol{v}_r||_1 \leq ||\boldsymbol{w}_l||_1 + ||\boldsymbol{w}_r||_1 = ||\boldsymbol{w}||_1$.

If the root is a "*", then

$$\begin{aligned}
||\boldsymbol{v}||_1 &= \sum_t \sum_s |v_l^t v_r^s| \\
&= \sum_t |v_l^t| \sum_s |v_r^s| \\
&= \sum_t |v_l^t| ||\boldsymbol{v}_r||_1 \\
&= ||\boldsymbol{v}_l||_1 ||\boldsymbol{v}_r||_1 \\
&\leq ||\boldsymbol{w}_l||_1 ||\boldsymbol{w}_r||_1 \\
&\leq ||\boldsymbol{w}||_1^2 \\
&\leq ||\boldsymbol{w}||_1
\end{aligned}$$

■

4.8.2 Proof for Lemma 14

Proof Remind that m_1 is the dimension of the covariate, and q is the number of basis functions. We define $\mathcal{G}'_{2k+1} \subset \mathcal{G}_{2k+1}$ as the set of labeled binary trees with exactly $2k+1$ nodes. In this step, we will show that $|\mathcal{G}'_{2k+1}| \leq 2^k (k)! (m_1 q)^{k+1}$.

We first show $|\mathcal{G}'_{2k+1}| \leq (m_1 q)^{k+1} (k)! 2^k$ for all $k = 0, 1, \cdots$:

$k = 0$, $|\mathcal{G}'_{2*0+1}| = m_1 q \leq (m_1 q)^{0+1} (0)! 2^0$;

$k = 1$, $|\mathcal{G}'_{2*1+1}| = 2(m_1 q)^2 - 2 m_1 q < (m_1 q)^{1+1} (1)! 2^1$;

Assume that $|\mathcal{G}'_{2*k+1}| \leq (m_1q)^{k+1}(k)!2^k$ for all $k < K$, then for $k = K$,

$$\begin{aligned}
|\mathcal{G}'_{2K+1}| &= 2 \sum_{i \in \{1,3,\cdots,2K-1\}} |\mathcal{G}'_i||\mathcal{G}'_{2K-i}| \\
&\leq 2 \sum_{i=0,\cdots,K-1} (m_1q)^{i+1}(i)!2^i \\
&\quad (m_1q)^{K-i-1+1}(K-i-1)!2^{K-i-1} \\
&= (m_1q)^{K+1}2^K \sum_{i=0,\cdots,K-1} (i)!(K-i-1)! \\
&\leq (m_1q)^{K+1}2^K \sum_{i=0,\cdots,K-1} (K-1)! \\
&\leq (m_1q)^{K+1}2^K(K)!
\end{aligned}$$

Since for $k \geq 1$, we have $2^{k-1} = \sum_{i=0}^{k-1} \frac{(k-1)!}{i!(k-1-i)!}$, or equivalently, $\frac{2^{k-1}}{(k-1)!} = \sum_{i=0}^{k-1} \frac{1}{i!(k-1-i)!}$, and since $1/x$ is concave, by Jensen's inequality, we have that $\frac{2^{k-1}}{(k)!} = \sum_{i=0}^{k-1} \frac{1}{k} \frac{1}{i!(k-1-i)!} \leq \frac{1}{\sum_{i=0}^{k-1} i!(k-1-i)!/k}$. Thus $\sum_{i=0}^{k-1} i!(k-1-i)! \leq k\frac{(k)!}{2^{k-1}}$ for $k \geq 1$. Except for the root node, a labeled binary tree consists of the left subtree and the right subtree. Thus

$$\begin{aligned}
|\mathcal{G}'_{2k+1}| &= 2 \sum_{i \in \{1,3,\cdots,2k-1\}} |\mathcal{G}'_i||\mathcal{G}'_{2k-i}| \\
&\leq (m_1q)^{k+1}2^k \sum_{i=0,\cdots,k-1} (i)!(k-i-1)! \\
&\leq (m_1q)^{k+1}2^k k\frac{(k)!}{2^{k-1}} \\
&= 2k(k)!(m_1q)^{k+1}
\end{aligned}$$

Finally, we will prove that $|\mathcal{G}_{2k+1}| \leq 4k(k)!(m_1q)^{k+1}$.

$$\begin{aligned}
|\mathcal{G}_{2k+1}| &= \sum_{i=0}^{k} |\mathcal{G}'_{2i+1}| \\
&\leq \sum_{i=1}^{k-1} 2i(i)!(m_1q)^{i+1} + m_1q + 2k(k)!(m_1q)^{k+1} \\
&\leq k * 2(k-1)(k-1)!(m_1q)^{k-1+1} + 2k(k)!(m_1q)^{k+1} \\
&\leq 4k(k)!(m_1q)^{k+1}
\end{aligned}$$

■

4.8.3 Proof for Lemma 15

Proof Define $M^*_{2k+1} = \max_{f \in \mathcal{G}'_{2k+1}} M_g$. Since $M^*_{2k+1} = M_{2k+1}$, it is equivalent to show $M^*_{2k+1} < (1.45)^{k+1}$. We will prove the lemma by induction.

$k = 0$, $M^*_{2*0+1} = 1 < (1.45)^1$;

$k = 1$, $M^*_{2*1+1} = max(1, 1+1) = 2 < (1.45)^2$;

$k = 2$, $M^*_{2*2+1} = 3 < (1.45)^3$;

Assume that $M^*_{2k+1} < (1.45)^{k+1}$ for all $k < K$, where $K \geq 3$, then for $k = K$,

$$\begin{aligned} M^*_{2k+1} &= \max_{i \in \{1,3,\cdots,2K-1\}} [\max(M^*_i M^*_{2K-i}, M^*_i + M^*_{2K-i})] \\ &< \max_{i \in \{1,3,\cdots,2K-1\}} [\max(1.45^{\frac{i-1}{2}+1} 1.45^{\frac{2K-i-1}{2}+1}, \\ &\qquad\qquad 1.45^{\frac{i-1}{2}+1} + 1.45^{\frac{2K-i-1}{2}+1})] \\ &= (1.45)^{K+1} \end{aligned}$$

■

4.8.4 Technical Lemma

The following technical lemma regarding the McDiarmid's condition for the supremum can be found in [73].

Lemma 16 *Let z be a random variable of support $\mathcal{Z} = (\mathbb{R}^{m_1}, \mathcal{Y})$ and distribution $\mathcal{D}$. Let $S = \{z_1 \dots z_n\}$ be a dataset of n samples. Let $\mathcal{H}$ be a hypothesis class satisfying $\mathcal{H} \subseteq \{h \mid h : \mathcal{Z} \to [0,1]\}$. The function:*

$$\varphi(S) = \sup_{h \in \mathcal{H}} \left(\mathbb{E}_{\mathcal{D}}[h] - \widehat{\mathbb{E}}_S[h] \right) \tag{4.10}$$

satisfies the following condition:

$$|\varphi(z_1, \dots, z_i, \dots, z_n) - \varphi(z_1, \dots, \widetilde{z}_i, \dots, z_n)| \leq 1/n$$
$$(\forall i, \forall z_1 \dots z_n, \widetilde{z}_i \in \mathcal{Z})$$

4.9 Detailed Greedy Search Algorithm and Illustration Example

For completeness, we present our main greedy search algorithm in detail in Algorithm 3, as well as the algorithm to compute the node weights in Algorithm 4. For simplicity, we assume the covariate $\boldsymbol{x}_m \in [0,1]^{m_1}$. As for the set of basis functions Φ, piecewise linear functions, Fourier basis functions, or truncated polynomials could be good choices in practice. We first define $g_{\boldsymbol{w}}(\boldsymbol{x})$ as the output of tree structure g with weights $\boldsymbol{w}$ for input $\boldsymbol{x}$. For instance, let g be the tree structure of Figure 4.2(a). With a corresponding weight for each leaf, $g_{\boldsymbol{w}}$ can be visualized as in Figure 4.2(b). Thus $g_{\boldsymbol{w}}(\boldsymbol{x}) = (w_1\phi_1(x_2) + w_2\phi_3(x_1)) * (w_3\phi_3(x_2) + w_4\phi_1(x_3))$ in this specific case. The loss function is defined as $L(g_{\boldsymbol{w}}; \boldsymbol{x}, \boldsymbol{y}) = \sum_{m=1}^{n} (y_m - \hat{y}_m)^2/2$, where $\hat{y}_m = g_{\boldsymbol{w}}(\boldsymbol{x}_m)$. We could explore the interaction structure g by adding and multiplying a basis function on a single dimension of covariate $\boldsymbol{x}$.

An example to illustrate Algorithm 4. Take Figure 4.7 for example, and assume we are trying to insert a new leaf wx_4^3 with either a "+" or "*" at the Node E, that is to replace the weighted leaf $-.05x_1$ with either $-.05x_1 + wx_4^3$ or $-.05x_1 * wx_4^3$. With an unknown weight w and an unknown intercept w_0, the output $\hat{y}_m$ for the input $\boldsymbol{x}_m$ of the new tree is

$$\begin{aligned} & w_0 + [.1x_{m2}^2 - .05x_{m1} + wx_{m4}^3](.3\sin(\pi x_{m2}) + .02x_{m3}) \\ & \triangleq w_0 + b(\boldsymbol{x}_m) + k(\boldsymbol{x}_m)(wx_{m4}^3 - .05x_{m1}) \end{aligned}$$

for "+", and

$$\begin{aligned} & w_0 + [.1x_{m2}^2 + wx_{m4}^3(-.05)x_{m1}](.3\sin(\pi x_{m2}) + .02x_{m3}) \\ & = w_0 + b(\boldsymbol{x}_m) + k(\boldsymbol{x}_m)(-.05wx_{m4}^3 x_{m1}) \end{aligned}$$

for "*".

Note that $b(\boldsymbol{x}_m)$ and $k(\boldsymbol{x}_m)$ are constant with respect to the to-be-defined weight, and thus, the optimization problems $\min_w \sum_{m=1}^{n} (y_m - w_0 - b(\boldsymbol{x}_m) - k(\boldsymbol{x}_m)(wx_{m4}^3 - .05x_{m1}))^2$

Algorithm 3 Greedy search algorithm

Input: $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)' \in \mathbb{R}^{n \times p}$: n data points

$\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$: n observations

Φ: a set of q basis functions, k: the number of iterations

Initialize the tree $g_{\boldsymbol{w}} = w_0 + w_1 \phi_{i_1}(x_{j_1})$, where

$(w_0, w_1, i_1, j_1) = \underset{(w_0', w', i', j')}{\operatorname{argmin}} \sum_{m=1}^{n} (y_m - w_0' - w' \phi_{i'}(x_{j'}))^2$

for $iters = 1$ **to** $k - 1$ **do**

for $node$ in $g_{\boldsymbol{w}}.leaves$ **do**

$path = path(g_{\boldsymbol{w}}.root, node)$

for $m = 1$ **to** n **do**

Algorithm 4 with input $(\boldsymbol{x}_m, g_{\boldsymbol{w}}, path)$:

$b_m = b(\boldsymbol{x}_m)$, $k_m = k(\boldsymbol{x}_m)$

$c_m = node(\boldsymbol{x}_m)$ (If $node$ is $w\phi_i(x_j)$, then $node(\boldsymbol{x}_m) = w\phi_i(x_{mj})$)

end for

$(w_0, w_+, i_+, j_+) = \underset{(w_0', w' \leq 1, i', j')}{\operatorname{argmin}} \sum_{m=1}^{n} (y_m - w_0' - b_m - k_m(c_m + w' \phi_{i'}(x_{mj'})))^2$, and define r_+ as the corresponding minimum value attained.

$(w_0, w_*, i_*, j_*) = \underset{(w_0', w' \leq 1, i', j')}{\operatorname{argmin}} \sum_{m=1}^{n} (y_m - w_0' - b_m - k_m(c_m w' \phi_{i'}(x_{mj'})))^2$, and define r_* as the corresponding minimum value attained.

if $r_+ < r_*$ **then**

Insert the new leaf $w_+ \phi_{i_+}(x_{j_+})$ at $node$ with "+", and call the new tree $g_{\boldsymbol{w}}^{node}$

$r_{node} = r_+$

else

Insert the new leaf($w_* \phi_{i_*}(x_{j_*})$ at $node$) with "*", and call the new tree $g_{\boldsymbol{w}}^{node}$

$r_{node} = r_*$

end if

Adjust all weights

end for

if $r_{node} < r_{best}$ **then**

$r_{BEST} = r_{node}$, $g_{\boldsymbol{w}}^{best} = g_{\boldsymbol{w}}^{node}$

end if

Update $g_{\boldsymbol{w}}$ with $g_{\boldsymbol{w}}^{best}$

end for

Output: $g_{\boldsymbol{w}}$

Algorithm 4 Compute $b(\boldsymbol{x}_m)$ and $k(\boldsymbol{x}_m)$

Input: $\boldsymbol{x}_m \in \mathbb{R}^{m_1}$: data point

$g_{\boldsymbol{w}}$: current weighted labeled tree

path: path from the root to the insert position

Initialize *root* as the root of $g_{\boldsymbol{w}}$, k=1, b=0

while *path* is not empty **do**

Define *subtree* as the !*path*[1] subtree of *root*

$val = evaluate(subtree, \boldsymbol{x}_m)$, where *evaluate* gives the output of the weighted labeled tree *subtree* with input $\boldsymbol{x}_m$

if *root* = " + " **then**

$b = b + val * k$

else if *root* = " ∗ " **then**

$k = val * k$

end if

Update *root* as its *path*[1] child

Remove the first element of *path*

end while

Output: (b, k)

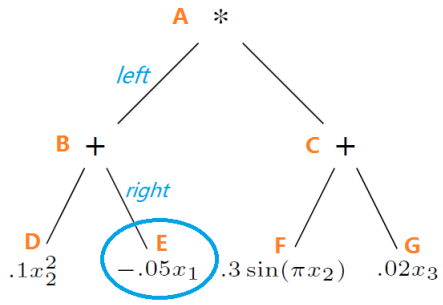


Fig. 4.7.: Inserting a new leaf at Node E.

and $\min_{w} \sum_{m=1}^{n} (y_m - w_0 - b(\boldsymbol{x}_m) - k(\boldsymbol{x}_m)(-.05wx_{m4}^3 x_{m1}))^2$ are both least square problems. We add a constraint $|w| \leq 1$ according to the assumption of Theorem 4.3.1, to ensure the uniform convergence. However, it is not straightforward to compute $b(\boldsymbol{x}_m)$ and $k(\boldsymbol{x}_m)$. As shown in Algorithm 4, we compute the value of $b(\boldsymbol{x}_m)$ and $k(\boldsymbol{x}_m)$ iteratively along the path from the root to the insert position. We continue with our current setting, and move on to compute $b(\boldsymbol{x}_m)$ and $k(\boldsymbol{x}_m)$ according to Algorithm 4, assuming $\boldsymbol{x}_m = (1, 1, 1)$.

1. Input: $\boldsymbol{x}_m = (1, 1, 1)$, $g_{\boldsymbol{w}}$ is the tree in Figure 4.7, $path = (left, right)$

2. Initialize: $root$ =Node A, $k = 1$,$b = 0$

3. In a first iteration $path[1] = left$, so define $subtree$ as the $right$ =!$left$ subtree of $root$(consisting of Nodes C, F, G),

   $val_m = evaluate(subtree, \boldsymbol{x}_m) = .3\sin(\pi x_{m2}) + .02x_{m3} = .02$

4. Since $root =$ " $*$ ", $k = val_m * k = .02$

5. Update $root$ as its left child: $root$ =Node B, $path = (right)$ after removing the first element of path

6. In a second iteration $path[1] = right$, so update $subtree$ as the $left$ =!$right$ subtree of $root$ (consisting of Node D only)

$val_m = evaluate(subtree, \boldsymbol{x}_m) = .1x_{m2}^2 = .1$

7. Since $root =$ " $+$ ", $b = b + val_m * k = .002$
8. Update $root$ as its right child, $path = ()$ after removing the first element of path
9. Stop the iterations since $path$ is empty
10. Return $(b(\boldsymbol{x}_m) = .002, k(\boldsymbol{x}_m) = .02)$

REFERENCES

REFERENCES

[1] J. H. Friedman *et al.*, "Multivariate adaptive regression splines," *The annals of statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[2] S. Miyata and X. Shen, "Adaptive free-knot splines," *Journal of Computational and Graphical Statistics*, vol. 12, no. 1, pp. 197–213, 2003.

[3] X. Wang, "Bayesian free-knot monotone cubic spline regression," *Journal of Computational and Graphical Statistics*, vol. 17, no. 2, pp. 373–387, 2008.

[4] C. Gu, *Smoothing spline ANOVA models*. Springer Science & Business Media, 2013, vol. 297.

[5] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[6] I. Barandiaran, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, 1998.

[7] L. Breiman, "Arcing the edge," Technical Report 486, Statistics Department, University of California at , Tech. Rep., 1997.

[8] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[9] ——, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[10] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent," in *Advances in neural information processing systems*, 2000, pp. 512–518.

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein, "On the expressive power of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2847–2854.

[14] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.

[15] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, pp. 463–482, 2002.

[16] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[17] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[18] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[19] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[20] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[21] B. Neyshabur, R. Tomioka, and N. Srebro, "Norm-based capacity control in neural networks," in *Conference on Learning Theory*, 2015, pp. 1376–1401.

[22] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks," in *International Conference on Learning Representations*, 2018.

[23] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu, "On the depth of deep neural networks: A theoretical view." in *AAAI*, 2016, pp. 2066–2072.

[24] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 6241–6250.

[25] X. Li, J. Lu, Z. Wang, J. Haupt, and T. Zhao, "On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond," *arXiv preprint arXiv:1806.05159*, 2018.

[26] N. Golowich, A. Rakhlin, and O. Shamir, "Size-independent sample complexity of neural networks," in *Proceedings of the 31st Conference On Learning Theory*, 2018.

[27] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[28] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[29] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[30] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," in *International Conference on Learning Representations*, 2018.

[31] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on Learning Theory*, 2016, pp. 907–940.

[32] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" in *International Conference on Learning Representations*, 2017.

[33] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 2979–2987.

[34] M. Telgarsky, "benefits of depth in neural networks," in *29th Annual Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, vol. 49. Columbia University, New York, New York, USA: PMLR, 2016, pp. 1517–1539.

[35] D. Yarotsky, "Error bounds for approximations with deep relu networks," *Neural Networks*, vol. 94, pp. 103–114, 2017.

[36] A. Maurer, "A vector-contraction inequality for rademacher complexities," in *International Conference on Algorithmic Learning Theory*, 2016, pp. 3–17.

[37] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 28, no. 3. PMLR, 2013, pp. 1319–1327.

[38] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optimization and Engineering*, vol. 10, no. 1, pp. 1–17, 2009.

[39] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *Journal of Machine Learning Research*, vol. 18, no. 19, pp. 1–53, 2017.

[40] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

[41] S. Boucheron, G. Lugosi, and O. Bousquet, "Concentration inequalities," in *Summer School on Machine Learning*, 2003, pp. 208–240.

[42] S. Shalev-Shwartz and Y. Singer, "A primal-dual perspective of online learning algorithms," *Machine Learning*, vol. 69, no. 2-3, pp. 115–142, 2007.

[43] S. M. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," in *Advances in Neural Information Processing Systems*, 2009, pp. 793–800.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[45] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[47] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations*, 2016.

[48] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 3288–3298.

[49] D. Molchanov, A. Ashukha, and D. P. Vetrov, "Variational dropout sparsifies deep neural networks," in *ICML*, 2017.

[50] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural computation*, vol. 4, no. 4, pp. 473–493, 1992.

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[52] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," in *International Conference on Learning Representations*, 2018.

[53] S. Srinivas, A. Subramanya, and R. V. Babu, "Training sparse neural networks," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 455–462.

[54] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l 1-ball for learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 272–279.

[55] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2015.

[57] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[58] P. Ravikumar, H. Liu, J. D. Lafferty, and L. A. Wasserman, "Spam: Sparse additive models." in *Neural Information Processing Systems*, 2007, pp. 1201–1208.

[59] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Advances in Neural Information Processing Systems*, 2009, pp. 396–404.

[60] D. K. Duvenaud, J. R. Lloyd, R. B. Grosse, J. B. Tenenbaum, and Z. Ghahramani, "Structure discovery in nonparametric regression through compositional kernel search." in *International Conference on Machine Learning (3)*, 2013, pp. 1166–1174.

[61] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[62] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 689–690.

[63] C. D. Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[64] G. Raskutti, B. Yu, and M. J. Wainwright, "Lower bounds on minimax rates for nonparametric regression with additive sparsity and smoothness," in *Advances in Neural Information Processing Systems*, 2009, pp. 1563–1570.

[65] S. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," *Neural Information Processing Systems*, vol. 21, pp. 793–800, 2008.

[66] N. P. Santhanam and M. J. Wainwright, "Information-theoretic limits of selecting binary graphical models in high dimensions," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4117–4134, 2012.

[67] W. Wang, M. Wainwright, and K. Ramchandran, "Information-theoretic bounds on model selection for Gaussian Markov random fields," *IEEE International Symposium on Information Theory*, pp. 1373 – 1377, 2010.

[68] T. Cover and J. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, 2006.

[69] B. Yu, "Assouad, Fano and Le Cam," in *Festschrift for Lucien Le Cam*. Springer, 1997, pp. 423–435.

[70] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, p. 45, 2013.

[71] A. Anandkumar, R. Ge, D. J. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2773–2832, 2014.

[72] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," *Uncertainty in Artificial Intelligence*, 2014.

[73] P. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 463–482, 2002.

VITA

VITA

Yixi Xu received a bachelor's degree in Applied Math from University of Science and Technology of China in 2009 and earned a doctoral degree in Statistics from Purdue University in 2019. Her research interests include deep learning, data science, machine learning and non-parametric statistics.