NEURAL NETWORKS MODELS WITH APPLICATIONS TO GENETIC STUDIES

By

Shan Zhang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Statistics — Doctor of Philosophy

2021

ABSTRACT

NEURAL NETWORKS MODELS WITH APPLICATIONS TO GENETIC STUDIES

By

Shan Zhang

Artificial intelligence (AI) is a thriving research field with many successful applications in areas such as computer vision and speech recognition. Machine learning methods, such as neural networks (NN), play a central role in modern AI technology. When it comes to genetic studies, a vanilla NN model suffers from small genetic effects without considering the underlying genetic structure (e.g., linkage disequilibrium). Furthermore in emerging research fields (e.g., imaging genetics), researchers need to deal with different types of data.

To address these challenges, I propose a functional neural networks (FNN) method which combines functional linear model and neural networks structure. FNN uses a series of basis functions to model genetic effects within gene sequences and phenotype data with spatial-temporal structure . Through simulations and real data applications, I demonstrate the advantages of FNN for high-dimensional genetic data analysis in terms of robustness and accuracy. The source code of FNN is available on https://github.com/szhang0629/FNN. The conditions for the consistency of proposed FNN model are also provided.

Transfer learning has been widely used in text and image classification and has demonstrated its outstanding performance in applications. Nonetheless, it has been rarely used in genetic research, and its performance on genetic data is still largely unknown. I use vanilla neural networks to investigate whether the knowledge learned from UKBiobank databases or Caucasian samples can be used to facilitate genetic research in small-scale studies or in minority populations. The experiments shows that transfer learning is useful in most genes.

I dedicate this dissertation to my parents, Houfa Zhang, Rongqun Shan and my friends, Xiaoxi, Peng, Xiaoran, Yuan, Chang and many others.

ACKNOWLEDGMENTS

Here, I would like to express my deepest gratitude to my advisors Dr. Qing Lu and Dr. Yuehua Cui for their support and guidance towards my studies and researches. Dr. Lu and Dr. Cui are extremely kind and knowledgeable. They always provide valuable insights and suggestions on the improvements of my work.

I would also like to extend my sincere thanks to my dissertation committee members, Dr. Lyudmila Sakhanenko and Dr. Haolei Weng. Their comments and suggestions are beneficial for my studies.

I am also grateful to the help I obtained from all the professors in the Department of Statistics and Probability. During my Ph.D. studies, I took most of the courses offered by our department.

During my five years at Michigan State University, I made lots of friends and because of them, I never feel lonely in these years. Many thanks to my senior classmates: Dr. Jingyi Zhang, Dr. Xiaoran Tong and Dr. Liangliang Zhang. They have become successful faculty members and statisticians in big companies. My thanks also go to my friends: Jian Song, Ken Lee, Runze Su, Peide Li, Chang Jiang, Yuan Zhou, Jinghang Lin and Di Wu. Without your help, I could not be who I am now and I sincerely wish all of you have a wonderful future.

Last but not least, I would like to express my sincere thanks to my parents for their support and confidence in me. I would like to treat this dissertation as my unique gift to both of you.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

In recent years, we have witnessed the emerging development of artificial intelligence (AI). In 2016, the AI AlphaGO beats professional Go player and dominates the game ([35]). Chatbot has become a basic tool for customer service in many companies ([17]). In the field of high technology, Tesla is a leading electric vehicles company in self-driving technology, and its Autopilot technology is based on AI ([5]). With these successful applications, AI has become a thriving research field, which attracts a lot of researchers from various fields.

Although AI technology has achieved impressive performance in gaming, natural language processing, image classification, and many other areas, it has been less popularly used in genetic studies, partially due to the high noise and small sample size of the genetic data. Normally a large amount of samples are required to train a good AI model. While it is easy to find millions of pictures of cars on the Internet, it is time consuming and costly to collect genetic data, along with individuals' clinical records and demographic information. Most genetic studies have a sample size of around several thousand, which is under-powered to train a sophisticated AI model, and the dimension of genetic data is also huge. The genetic information is stored in the nucleotide sequences of DNA, which has over one billion nucleotides. Even if we focus on only one gene, there are still possible over ten thousand

nucleotides. It is hard to build an AI model with ten thousand variables given limited sample size. Finally, the genetic information alone can only explain the limited variance of phenotypes. Complex phenotypes, such as height, are determined by both gene and environmental factors. The AI model intends to explain all the variance with the given genetic data, which can lead to great prediction in the training dataset, while has low performance in the testing dataset. Many regularization methods are available to alleviate this problem, but overfitting remains a challenging problem for AI methods, especially in the high-dimensional setting.

In this chapter, the background of genetic research is introduced in section 1.2. In section 1.3, the vanilla neural networks models are briefly introduced. The objective and organization of the dissertation are given in section 1.4.

1.2 Genetic Data

I start with a brief introduction to genetic data. Although DNA has a huge number of nucleotides, over 99% of them are the same among all human beings. So, I only consider the nucleotide with a variation into consideration, which is called single nucleotide polymorphisms (SNPs). Most SNPs have two copies of alleles. Assuming an additive mode of inheritance, SNPs can be coded as {0, 1, 2} based on the number of minor alleles. In my dissertation, I focus on studying genes, which involve multiple SNPs (e.g., around 30 SNPs). In a gene, linkage disequilibrium (LD) exists among SNPs [7]. Based on LD, I can assume the genetic effect of an SNP is related to that of its nearby SNPs. Under this assumption, I develop a functional neural network method, which is introduced in detail in Chapter 2.

A variety of phenotypes, such as brain volume, weight, or smoking frequency, are used

in genetic studies. If the age is considered as the index, these phenotypes can be naturally modeled as a function of age. The temporal relationship within the phenotype can be considered by assuming the continuity of the function. This concept has been used in the proposed model of the dissertation.

1.3 Neural Networks

The neural networks (NN) model plays the fundamental role in the AI field. A basic NN model $f : \mathbb{R}^d \to \mathbb{R}$ can be realized by:

$$f(x) = \sum_{j=1}^{J} a_j \sigma(w_j x + b_j) + c,$$

where J, a_j c, w_j and b_j are parameters of the NN model. Among them, w_j is a d-dimension vector while all other parameters are scalars. Based on the universal approximation theorem, given enough hidden units, the NN model can approximate to any continuous functions which have compact supports on $\mathbb{R}^d \to \mathbb{R}$.

The domain of the NN function is on a finite dimensional space. In this dissertation, a new neural network model defined on a functional space is proposed and its statistical properties such as consistency are developed.

1.4 Organization

The remaining dissertation is organized as follows. In Chapter 2, I propose a new neural network model called the functional neural networks model, which can deal with functional data and take the spatial and temporal relationships into consideration. In Chapter 3, I

prove the consistency of the proposed model. In Chapter 4, I apply the transfer learning idea to genetic studies. In Chapter 5, I discuss possible extensions of these methods and future work.

Chapter 2

Functional Neural Networks for High-dimensional Genetic Data Analysis

2.1 Introduction

With the recent advancement in high-throughput technologies, genome-wide association studies (GWAS) and sequencing studies have been commonly adopted to uncover new genetic variants predisposing to common complex diseases ([15], [32]). During the past decade, thousands of genetic variants have been identified through GWAS and sequencing studies, some with compelling biological plausibility for a role in disease etiology ([22]). Despite such success, for most complex diseases, identified genetic variants are associated with small effect sizes and only explain a small fraction of total heritability ([20]). Many common diseases are influenced by the interplay of multiple genes and other risk factors (e.g., environmental determinants) in a complex manner ([4]). This complexity, however, has not been taken fully into account by many existing approaches, which often assume genetic variants related to disease phenotypes in an additive and linear manner.

Machine learning methods, such as the neural network (NN) method, holds great promise for genetic research ([6]). Based on its hierarchical structure, NN learns complicated features from simpler ones, making it capable of capturing non-linear and non-additive genetic effects

(e.g., gene-gene interaction effects). Given its exceptional performance, NN has been increasingly used in genomics, especially in regulatory genomics, variant calling, and pathogenicity scores ([39]). Despite its successful applications in these areas, the use of NN in revealing the complex relationships between genetic variants and common diseases is still limited.

While the great potential of NN in genetic data analysis of complex diseases is reasonably expected, the high dimensionality of the genetic data and the complexity of genetic structure bring tremendous analytic challenges. The high-throughput technology allows us to simultaneously evaluate the role of thousands or even millions of variants in complex diseases ([10]). Nevertheless, fitting an NN on such a large number of genetic variables could bring a serious overfitting issue. As for genetics studies, linkage disequilibrium (LD) exists among neighboring variants, and disease-associated variants are often in an LD block. Considering the underlying genetic structure can help us to combat the overfitting issue. Moreover, different types of phenotypes are often collected in a study. Besides the measurement of a disease phenotype, which is typically a scalar variable, researchers are also interested in studying a vector of variables (e.g., the progression of disease measured over time). With the rapidly evolving technologies and ever-decreasing cost, studies are starting to collect omics and imaging data, which are often high-dimensional and stored as matrices or tensors. While omics and imaging data provide us a great opportunity to study complex diseases, few methods are currently available for high-dimensional genetic data analysis of complex phenotypes (e.g., vectors and matrices) with the consideration of non-linear and non-additive effects.

To address these emerging challenges and facilitate the high-dimensional genetic data analysis of complex diseases, I propose a functional neural network (FNN) that inherits the strengths from both NN and the functional linear model (FLM). FLM is a popularly used method in functional data analysis (FDA) ([25]; [24]), which deals with data in the form of

functions. FLM has been commonly used to analyze data measured over time and has been increasingly used in high-dimensional data analysis, such as genetic data analysis ([7]; [33]) and imaging data analysis ([36]). Specifically, with the high-dimensional genetic variants as the input layer and various types of phenotypes as the output layer, the proposed FNN uses a series of basis functions to obtain a representation of functional data in each layer ([27]), and further builds multiple hidden layers via functional linear models. FNN has a number of attractive features: 1) it has a built-in facility to account for the underlying structures of complex phenotypes and genetic data (e.g., LD), which helps capture disease-related variants and overcome the curse of dimensionality; 2) it uses a functional neural network to model the complex relationship between genetic variants and disease phenotypes; and 3) it can be used to analyze different types of phenotypes (e.g., scalar, vector, and matrix). Through simulations and two real data applications, I show that FNN outperforms conventional methods, such as NN and FLM. I have also shown, at certain conditions, FNN can be simplified to NN. In other words, FNN can be viewed as the generalization of NN to high-dimensional data with complex phenotypes.

The remaining chapter is organized as follows: the FNN method is proposed in Section 2. Section 3 uses simulations to compare the performance of FNN with those of FLM and NN. In Section 4, I illustrate the methods via two real data applications. Summary and discussion are provided in Section 5. The technical details of the methods are described in the Appendix.

2.2 Method

In this section, I introduce NN and two types of FLMs for different types of phenotypes at first. Based on the concepts of FLM and NN, I develop FNN for high-dimensional genetic data analysis.

Throughout the Chapter 2, I use lower case letters, bold lower case letters, upper case letters and bold upper case letters to denote scalars, vectors, matrices and functions respectively when it comes to Roman alphabet. For example, b is a scalar. $\mathbf{b}$ is a vector with $\mathbf{b}_j$ as the j-th element of $\mathbf{b}$. B is a matrix with B_i as the ith row of matrix B. B_{ij} denotes a scalar whose value is the j-th element of B_i. $\mathbf{B}$ is a function. Due to large amount of parameters and functional representations, Greek alphabets are also used to denote values or functions.

2.2.1 Functional Linear Model

This section will introduce the functional linear models for a scalar phenotype and complex phenotypes separately. The details on the solutions of them are provided in Appendix A.

2.2.1.1 FLM for a scalar phenotype

I briefly introduce FLM in the setting of genetic data analysis with a scalar phenotype ([26], [34]). Let y_i and $Z_i = (z_{i1}, \ldots, z_{iq})$ denote a scalar phenotype (e.g., systolic blood pressure) and quantitative/qualitative covariates (e.g., age and gender) for the i-th individual. Let $[G_{i1}, G_{i2}, ..., G_{ip}]$ be the single-nucleotide variants (SNVs) in a SNV set (e.g., a gene), coded as additive (i.e., $G_{ik} \in \{0, 1, 2\}$, refers to the number of minor allele), and t_k be the corresponding position scaled into $[0, 1]$. Given the genotypes and the positions, I model the

genetic variant function $\mathbf{G}_i(t), t \in [0,1]$ as a linear combination of Dirac Delta functions, i.e.

$$\mathbf{G}_i(t) = \sum_{k=1}^{p} G_{ik}\delta_{t_k}(t), \tag{2.1}$$

where $\delta_{t_k}(t)$ satisfies $\int_{t=-\infty}^{\infty} f(t)\delta_{t_k}(t)\mathrm{d}t = f(t_k)$. The functional linear model can then be used to model the relationship between the scalar phenotype y_i and the genetic variant function $\mathbf{G}_i(t)$ with the consideration of Z_i:

$$\hat{y}_i = \theta_0 + Z_i\boldsymbol{\theta} + \int \mathbf{G}_i(t)\beta(t)\mathrm{d}t, \tag{2.2}$$

where $\beta(t)$ is the coefficient function measuring the genetic effects in a genetic region. The parameters θ_0 and $\boldsymbol{\theta}$ are the intercept and coefficients of covariates, respectively.

2.2.1.2 FLM for complex phenotypes

FLM can be easily extended to handle complex phenotypes (e.g., disease phenotypes measured over time) by modeling complex phenotypes as functions. For instance, I can modify the phenotype as $\mathbf{Y}_i(s)$ and use the following model ([14]) to consider the spatial/temporal dependence at location/time s,

$$\hat{\mathbf{Y}}_i(s) = Z_i\boldsymbol{\theta} + \alpha_0(s) + \int \alpha(s,t)\mathbf{G}_i(t)\mathrm{d}t, \tag{2.3}$$

where $\alpha(s,t)$ is a bivariate function and $\alpha_0(s)$ is an intercept function. The observed phenotype is treated as the function realization in discrete points s_{ij}, denoted as

$$\hat{y}_{ij} = \hat{\mathbf{Y}}_i(s_{ij}) = Z_i\boldsymbol{\theta} + \alpha_0(s_{ij}) + \int \alpha(s_{ij},t)\mathbf{G}_i(t)\mathrm{d}t. \tag{2.4}$$

FLM has many desirable features for high-dimensional genetic data analysis. By considering the effects of SNVs as functions, I could utilize information from adjacent SNVs (i.e., nearby SNVs tend to have similar effects due to LD) and reduce the number of parameters, which help us capture true signals and overcome the curse of dimensionality. Moreover, FLM has its own uniqueness of handling measurement errors and missing data, which are quite common in genetic data ([34]).

Despite these advantages, FLM is not suitable to model complex relationships between SNVs and phenotypes or the inter-relationship between SNVs (e.g., interactions). In order to address these issues, I integrate the idea of NN into FLM to improve its capacity of modeling complex non-linear and non-additive genetic effects.

2.2.2 Neural networks

Neural networks can be viewed as multi-stage nonlinear regression models. A D-layer NN model $\mathbf{F}_N : \mathbb{R}^{p+q} \to \mathbb{R}$ can be expressed in a recursive manner:

$$
\begin{aligned}
X_i^{(1)} &= \sigma(b^{(1)} + Z_i\boldsymbol{\theta} + G_i W^{(1)}), \\
X_i^{(d)} &= \sigma(X_i^{(d-1)} W^{(d)} + \mathbf{b}^{(d)}), \quad d = 2, \ldots, D-1, \\
\hat{y}_i &= f(X_i^{(D-1)} W^{(D)} + \mathbf{b}^{(D)}),
\end{aligned}
$$

where $Z = \{z_{ik}\}_{n\times m}$, $G = \{G_{ik}\}_{n\times p} = \{\mathbf{G}_i(t_k)\}_{n\times p}$ and $\{X^{(d)}, d = 1, ..., D-1\}$ are covariates, genetic variables (e.g., SNVs) and hidden layers, respectively. $\sigma^{(d)}$ is the activation function (e.g.,the sigmoid function) and f is the link function (e.g., the identity function). $W^{(d)}$ and $b^{(d)}$ are the weight matrix and bias vector for the d-th layer. The detail of NN is described in Goodfellow et al [9].

A L^2 penalty is commonly used in NN to control the model complexity. The penalized mean square error loss function is thus defined as,

$$\mathbf{R}(W, \mathbf{b}) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{d=1}^{D}(\|W^{(d)}\|_F^2 + \|\mathbf{b}^{(d)}\|_2^2).$$

The back-propagation and gradient descent algorithms can then be used to solve $(W, \mathbf{b})$. In this chapter, the learning rate of NN is tuned by using the adaptive optimization method, Adadelta ([37]). The number of hidden units and the parameter λ are also tuning parameters in this chapter. To determine the value of λ, I use the hyperparameter optimization algorithm introduced in [9]. Specifically, I choose the value of λ in a logarithmic scale so that the range of the value is wide enough, i.e., $\{10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}\}$. The training data is split into the subtrain dataset and the validation dataset in the ratio of $4:1$. I apply NN to the subtrain dataset with all possible λ values, and choose the best λ value based on the mean square error in the validation dataset. I then use the selected λ value to train the final model in the whole training set and obtain the model parameters. The same procedure can be used to determine the number of hidden units, which is chosen from $\{2, 4, 8, 16, 32, 64\}$.

2.2.3 Functional neural networks

Functional neural networks can be viewed as an extension of NN, where the weights and biases are modeled as functions. By treating the weights and biases as functions, FNN can consider LD among SNVs and spatial-temporal relation of phenotypes, as well as reduce the number of parameters.

In FNN, a genetic variant function $\mathbf{G}_i(t)$ based on SNVs is obtained at first as described in 2.1. Given the genetic variant function $\mathbf{G}_i(t)$, the first hidden layer $X_i^{(1)}(t^{(1)})$ can be

constructed by using equation (2.5). The additional $D-2$ hidden layers, $X_i^{(d)}(t^{(d)})$, are then built recursively with different functional coefficients as shown in (2.6). Depending on the types of phenotypes, equation (2.7) or (2.8) can be used to fit a scalar phenotype or a complex phenotype, respectively. The FNN model can thus be expressed as:

$$\mathbf{X}_i^{(1)}(t^{(1)}) = \sigma\Big(Z\boldsymbol{\theta} + \alpha_0^{(1)}(t^{(1)}) + \int \alpha^{(1)}(t^{(1)}, t)\mathbf{G}(t)\mathrm{d}t\Big), \tag{2.5}$$

$$\mathbf{X}_i^{(d)}(t^{(d)}) = \sigma\Big(\alpha_0^{(d)}(t^{(d)}) + \int \alpha^{(d)}(t^{(d)}, t^{(d-1)})X_i^{(d-1)}(t^{(d-1)})\mathrm{d}t^{(d-1)}\Big), \quad 1 < d < D, \tag{2.6}$$

$$\hat{y}_i = f\Big(a_0^{(D)} + \int \alpha^{D}(t^{(D-1)})\mathbf{X}_i^{(D-1)}(t^{(D-1)})\mathrm{d}t^{(D-1)}\Big), \tag{2.7}$$

$$\hat{y}_{ij} = \hat{\mathbf{Y}}_i(s_{ij}) = f\Big(\alpha_0^{(D)}(s_{ij}) + \int \alpha^{(D)}(s_{ij}, t^{(D-1)})\mathbf{X}_i^{(D-1)}(t^{(D-1)})\mathrm{d}t^{(D-1)})\Big), \tag{2.8}$$

where σ and f are the activation function and the link function. In this chapter, I use the sigmoid function as the activation function and the identity function as the link function. $\alpha_0^{(1)}(t^{(1)})$, $\alpha_0^{(d)}(t^{(d)})$, $a_0^{(D)}$ and $\alpha_0^{(D)}(s_{ij})$ are respectively bias functions or bias for the input layer, the hidden layer, the output layer with the scalar phenotype and the output layer with the complex phenotype, where t, $t^{(d)}$ and s_{ij} are respectively the coordinate systems for the genetic variant function (e.g., locus positions), d-th layer and the response function (e.g., time). $\alpha^{(1)}(t^{(1)}, t)$, $\alpha^{(d)}(t^{(d)}, t^{(d-1)})$, $\alpha^{D}(t^{(D-1)})$ and $\alpha^{(D)}(s_{ij}, t^{(D-1)})$ are weight functions for the input layer, the hidden layer, the output layer with the scalar phenotype and the output layer with the complex phenotype, respectively. The explicit forms of the bias and weight function are written below,

$$\alpha_0^{(d)}(t^{(d)}) = \sum_{k^{(d)}=1}^{l^{(d)}} \boldsymbol{b}_{k^{(d)}}^{(d)} \beta_{k^{(d)}}^{(d)}(t^{(d)}),$$

$$\alpha^{(d)}(t^{(d)}, t^{(d-1)}) = \sum_{k^{(d)}=1}^{l^{(d)}} \sum_{k^{(d-1)}=1}^{l^{(d-1)}} W_{k^{(d-1)}k^{(d)}}^{(d)} \beta_{k^{(d-1)}}^{(d-1)}(t^{(d-1)}) \beta_{k^{(d)}}^{(d)}(t^{(d)}),$$

where $(W, \boldsymbol{b}) = \{W^{(d)}, \boldsymbol{b}^{(d)} | d = 1, \ldots, D\}$ are parameters of interest. $\{\beta_{k^{(d)}}^{(d)}(t^{(d)}) | k^{(d)} = 1, \ldots, l^{(d)}\}$ is the predetermined basis functions of the d-th layer.

When Y is a scalar, $a_0^{(D)}$ is a scalar and $\alpha^{(D)}$ is a univariate function. If Y is a vector (e.g., measurements over different time points), then $\alpha_0^{(D)}$ is a univariate function and $\alpha^{(D)}$ is a bivariate function. The model can be extended to a variety of phenotypes. For instance, if Y is a matrix, then the weight function can be expressed as a Cartesian product of three basis function systems.

As shown in Figure 1, NN and FNN have different types of weights and biases. While weights and biases in NN are vectors or scalars, weights and biases in FNN are functions. Correspondingly, NN uses multiplication on the weight matrix $W^{(d)}$, while FNN performs integration on the functional coefficient $\alpha^{(d)}$. In practice, integration is realized by the numeric integration in the form of summation. In this sense, the weights and biases in NN are independent values, while those in FNN are realizations of a parameterized curve function.

To estimate the parameters in the FNN model, I define the loss function $\mathbf{L}(y, \hat{y})$, cost function $\mathbf{R}(\alpha, \alpha_0)$ and objective function $\mathbf{O}(\alpha, \alpha_0)$ as follows:

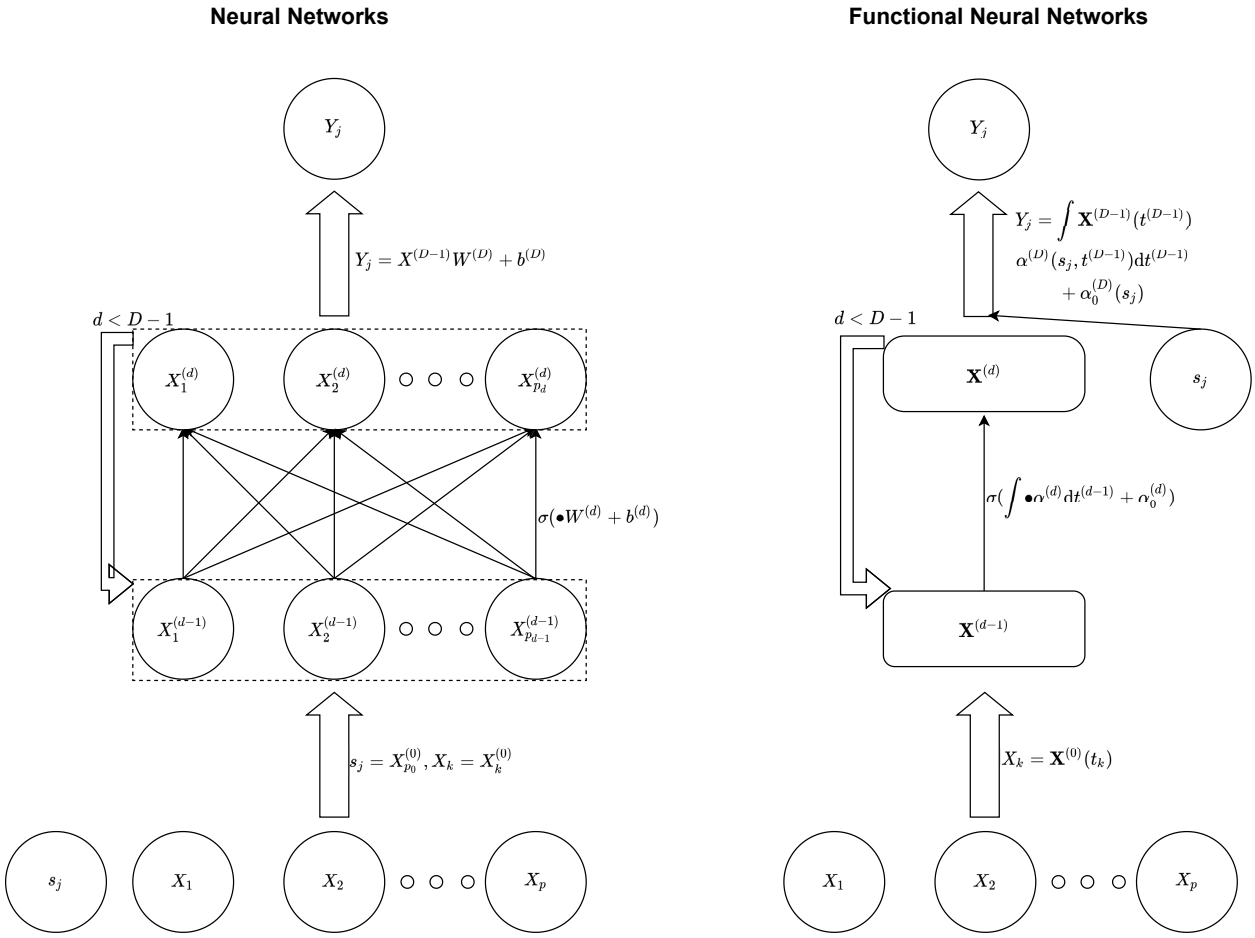

Figure 2.1: An illustration of neural networks and functional neural networks

$$\mathbf{L}(y, \hat{y}) = (y - \hat{y})^2,$$

$$\mathbf{R}(\alpha, \alpha_0) = \sum_{i,j} \mathbf{L}(y(t_{ij})\hat{y}(t_{ij})),$$

$$\mathbf{O}(\alpha, \alpha_0) = \mathbf{R}(\alpha, \alpha_0) + \lambda \mathbf{J}(\alpha, \alpha_0).$$

A penalty $\mathbf{J}(\alpha, \alpha_0)$ term is added to the error function to address the overfitting issue raised by the high-dimensional genetic data. Instead of imposing a penalty (e.g., the lasso penalty) on the parameters in NN, I impose a penalty on the second-order derivative of hidden functions to control the smoothness of the functions,

$$\mathbf{J}(\alpha, \alpha_0) = \sum_{d=1}^{D} \mathbf{J}^{(d)}(\alpha^{(d)}, \alpha_0^{(d)}),$$

$$\mathbf{J}^{(d)}(\alpha^{(d)}, \alpha_0^{(d)}) = \int \int \left[\left(\frac{\partial^2 \alpha^{(d)}}{\partial t^{(d)2}} \right)^2 + \left(\frac{\partial^2 \alpha^{(d)}}{\partial t^{(d-1)2}} \right)^2 \right] \mathrm{d}t^{(d)} \mathrm{d}t^{(d-1)},$$

where λ is a hyper-parameter determined by cross-validation. The penalized loss function can be optimized via a back-propagation algorithm, which is described in the appendix B.

2.3 Simulations

Simulations were conducted to compare the performance of FNN with those of FLM and NN. In order to mimic the real structure of genetic data (e.g., allele frequencies and LD), the genetic data was obtained from the real sequencing data located on Chromosome 17: 7344328 - 8344327 from the 1,000 Genome project [3]. Specifically, I randomly chose a segment of SNVs, 200 training samples, and 50 testing samples. Due to the intensive computation time

of analyzing complex phenotypes (e.g., matrices), 100 replicates are simulated to illustrate the performance.

Based on the genetic data, we simulated different types of phenotypes (i.e., scalars, vectors, and matrices), and evaluated both linear and non-linear relationships between genotypes and phenotypes. To reflect the real disease scenarios, we added random noise to the simulated data, and compared the performance of three methods by gradually increasing the noise variance level. A general form of the simulation model is given below,

$$Y_{ij} = \mathbf{Y}_i(s_{ij}) + \varepsilon_{ij}, j = 1, 2, \ldots, m,$$

where $\mathbf{Y}_i(s) = \mathbf{F}(s; \mathbf{G}_i(t))$ corresponds to a function-function mapping $L^2([0,1]) \to L^2([0,1])$. It models a functional phenotype $Y_i(s)$ with the genetic variant function $\mathbf{G}_i(t)$, where s is the coordinate system of the phenotype function and s_{ij} is a point in the coordinate space, The noise ε_{ij} follows an i.i.d normal distribution, $N(0, \sigma^2)$, and m is the dimension of the phenotype. The explicit expression of the model is given in the following simulations.

In all simulations, the function has K points as discrete realizations of the function, and the points are randomly generated from $[0, 1]$. When $m = 1$, $Y_i(x)$ is a scalar. For vector types of phenotypes, I set $m = 20$ (i.e., the phenotype of ith individual is a vector of 20 elements). In the matrix setting, I apply outer product on two functions in the form of (2.9). The discrete observational points are randomly generated from a rectangle in $[0, 1] \times [0, 1]$, where $m = 400$.

For simplicity, I compare NN and FNN with one hidden layer. The number of hidden units in NN, the number of basis functions for FLM and the number of basis functions for the hidden layer in FNN are fixed at 40. The numbers of basis functions of the first

layer and the last layer of FNN are 60 and 20, respectively. The basis function system used in FLM and FNN is a fourth-order B-spline basis function system. I use an adaptive method, Adadelta [37], to determine the learning rate. The hyper parameter λ is chosen from $\{10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}\}$ and is determined by the validation process.

2.3.1 Simulation 1: The effect of the phenotype dimension

In this simulation, I evaluated the performance of three methods under different types of phenotypes (i.e., scalar, vector, and matrix). For each replicate, I randomly chose 200 samples as training data and 50 samples as testing data, each with 200 SNVs obtained from the 1,000 Genome project. Based on the genotypes, I simulated phenotypes using the following model,

$$\mathbf{F}(x; \mathbf{G}(t)) = \sum_{l=1}^{20} \int_0^1 c_l \mathbf{G}^{e_l}(t) \mathbf{B}_{1l}(t) \mathbf{B}_{2l}(x) \mathrm{d}t, \tag{2.9}$$

where B_{1l} and B_{2l} are two 4th-order B-spline basis functions with 10 knots randomly generated from $[0, 1]$. The coefficient c_i and e_i are randomly generated from uniform distributions, $U[-2, 2]$ and $U[1/3, 3]$, respectively.

To compare the performance of the three methods, I computed the mean squared error (MSE) on both training and testing samples for three different types of phenotypes (i.e., scalar, vector, and matrix) and three different levels of noise variance (i.e., σ =0.3, 0.6, and 1.2). The left group of box plots and the right group of box plots in each panel of Figure 2 summarize the MSE of three methods calculated from the training data and testing data, respectively. Overall, FNN attains better or comparable performance than FLM and NN in the testing data. NN performs the best in training data but is subject to low performance in

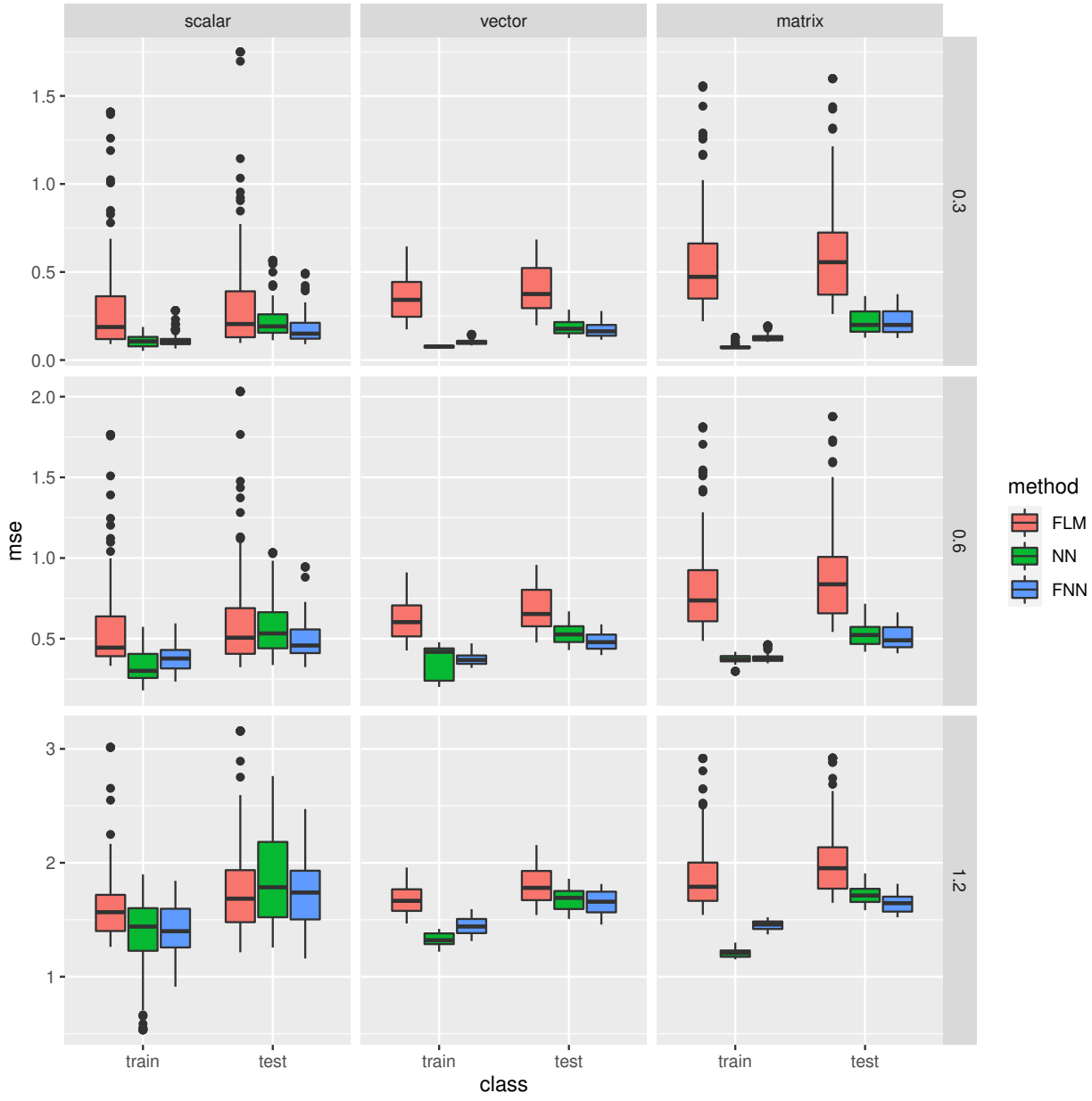

Figure 2.2: Accuracy comparison of three methods for three different types of phenotypes (i.e., scalar, vector, and matrix) and three different levels of noise (i.e., σ =0.3, 0.6, and 1.2)

testing data, which can be potentially explained by overfitting caused by hundreds of genetic variables. The overfitting issue becomes more serious with the increased noise level. On the other hand, FLM provides robustness against the overfitting issue due to its simplicity and fewer parameters. However, the linear structure of FLM fails to capture complex features of the data, leading to low performance in both training data and testing data. FNN has a good balance between bias and variance. It is able to capture complex features of data while remains robust to the overfitting issue.

2.3.2 Simulation 2: The effect of the underlying function

I further evaluated the performance of three methods under different underlying functions (i.e., polynomial, logistic, and linear functions). I denote the polynomial function as "polyno", which is described in equation (2.9). In addition, we simulated a "logist" function, which performs the logit transformation on the genetic variables,

$$\mathbf{F}(x;\mathbf{G}(t)) = \sum_{l=1}^{20} \int_0^1 c_l(1-\exp(e_l\mathbf{G}(t)))\cos(a_l t + u_l)\cos(b_l x + v_l)\mathrm{d}t, \tag{2.10}$$

where $a_l, b_l, c_l \sim U[-12, 12]$, $e_l \sim U[0.1, 1]$, and $u_l, v_l \sim U[-\pi, \pi]$. To evaluate the method's performance in a linear setting, we also simulated a linear function denoted by "linear",

$$\begin{aligned}\mathbf{F}(x;\mathbf{G}(t)) = \sum_{l=1}^{20} \int_0^1 (c_l\mathbf{G}(t)\cos(a_l t + u_l)\cos(b_l x + v_l) \\ + d_l\mathbf{G}(t)\mathbf{B}_{1l}(t)\mathbf{B}_{2l}(x))\mathrm{d}t,\end{aligned} \tag{2.11}$$

where $a_l, b_l, c_l, d_l \sim U[-12, 12]$, $e_l \sim U[0.1, 1]$, and $u_l, v_l \sim U[-\pi, \pi]$, B_{1l} and B_{2l} are two fourth-order B-spline basis functions with 10 knots generated randomly from $[0, 1]$.

As shown in Figure 3, overall, FNN has better or at least comparable performance than the other two methods. As expected, FLM performs the best in the linear setting, although the performance of FNN is close to that of FLM. FNN is slightly better than NN but outperforms FLM in the non-linear settings. While NN also outperforms FLM in the non-linear settings, it suffers the overfitting issue due to the high dimensionality of the data.

2.3.3 Simulation 3: The effect of the input dimension

In this simulation, I compared the performance of three methods with the increased dimension of input variables (i.e., $p = 80, 200, 500$, and $m = 20$). The simulation setting is the same as the one used in simulation 2 with the polynomial function, except for the dimension of input variables.

As shown in Figure 4, when the number of input variable p and the noise level increase, all three methods have decreased accuracy in terms of testing MSE. Among the three methods, FLM has the most robust performance due to the linear structure it imposes. Nonetheless, such a structure also limits its performance for modeling non-linear effects. While NN is powerful for capturing non-linear effects, it suffers from the overfitting issue, especially with the increase of the number of input variables and the noise level. Overall, FNN attained the best performance. FNN inherits the advantages of both NN and FLM. It has the capacity of capturing non-linearity as NN and is generally robust to the increased number of input variables and the increased noise level as FLM.

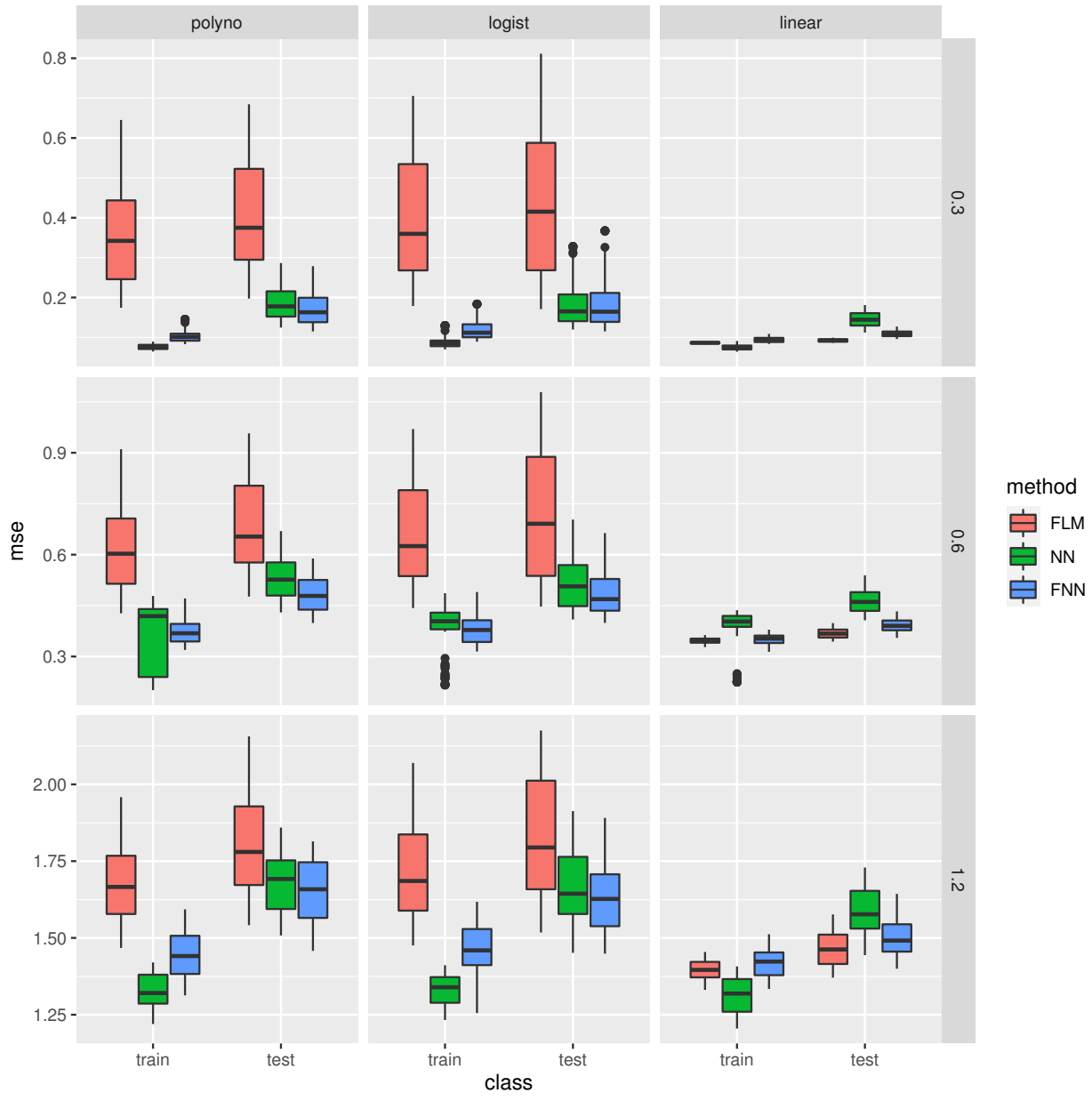

Figure 2.3: Accuracy comparison of three methods for different underlying functions

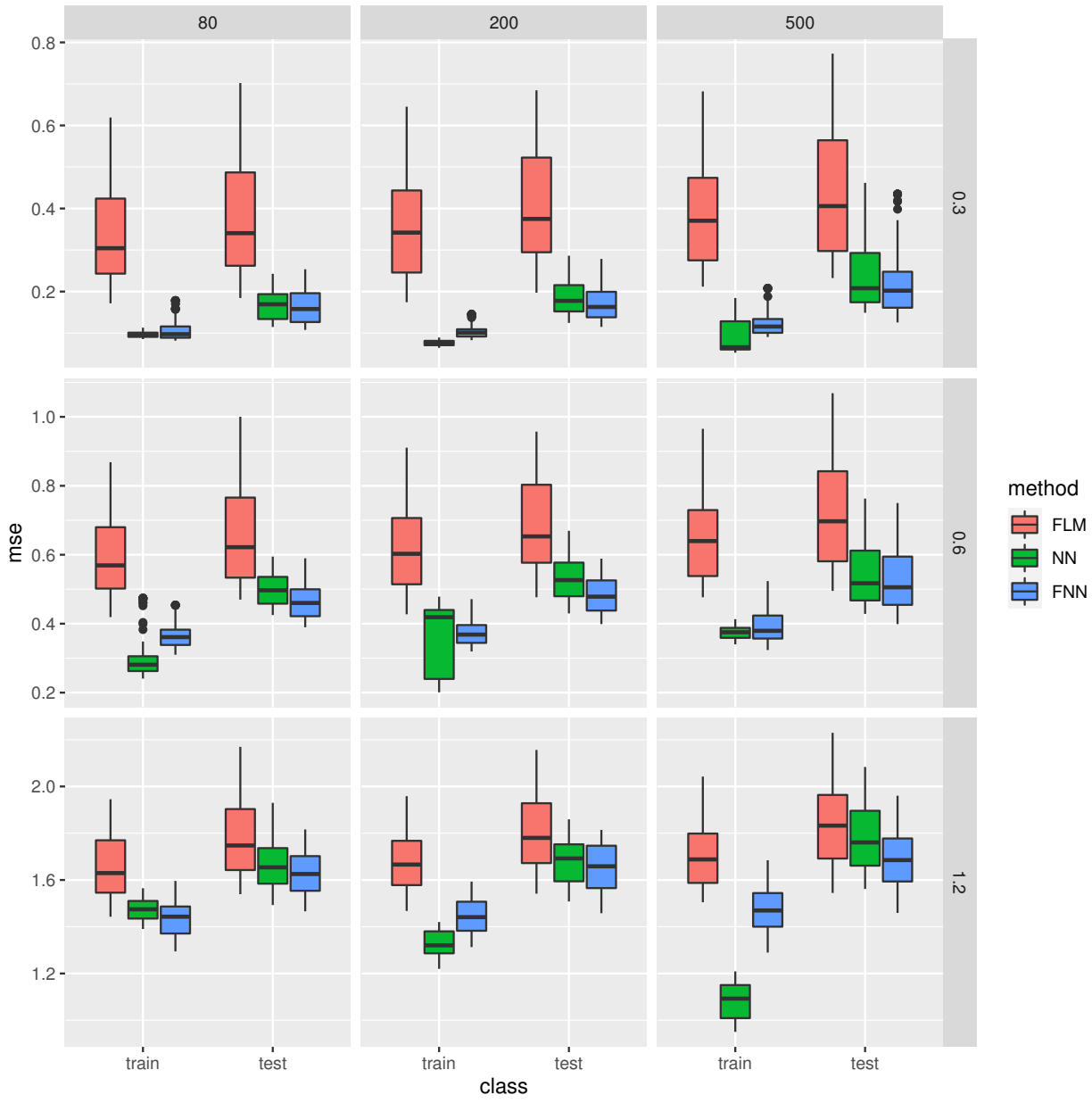

Figure 2.4: Accuracy comparison of three methods for different numbers of input variables (i.e., 80, 200, and 500)

2.4 Real Data Applications

2.4.1 The role of *CHRNA5*, race, gender, and age in predicting cigarette smoking

Cigarette smoking is one of the leading causes of preventable disease, contributing to 5 million deaths worldwide each year ([21]). During the last decade, a great deal of progress has been made in identifying genetic variants associated with smoking. Among those findings, the *CHRNA5* gene has been identified and confirmed in several large-scale studies ([19]). In this application, I will evaluate the role of *CHRNA5*, race, gender, and age in predicting cigarette smoking, measured by smoking frequency.

The genetic dataset to be analyzed is from a large-scale genetic study of addiction: genetics and environment (SAGE). The participants of the SAGE were unrelated individuals selected from three independent studies: COGEND, COGA, and FSCD. The SAGE included smoking measurements and personal characteristics (e.g., age). Prior to the analysis, I re-assessed the quality of the genetic data. Markers and samples with low quality and high missing value are removed from the analysis. The final data includes 2502 samples, among which 795 samples are African-American, 1707 samples are Caucasian, 1479 samples are female, and 1023 samples are male.

Since cigarette smoking is dependent on age, smoking quantity is considered as a function of age, and model its relationship with *CHRNA5*, race, gender, and age. To evaluate the prediction performance of the three methods, I randomly split the whole data into training (80%) and testing (20%) data. By applying the methods to the training data, I built models and obtained the training MSE. The models were then evaluated on the testing data by

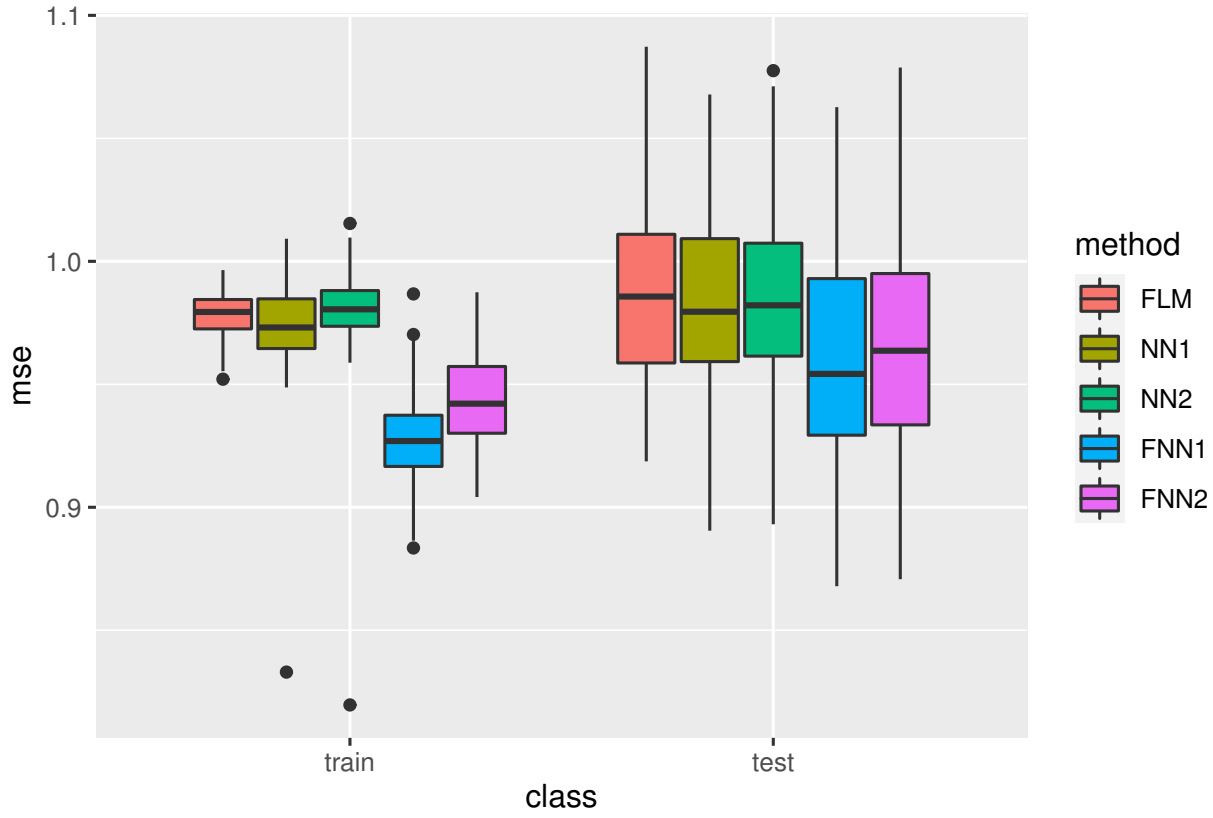

Figure 2.5: The role of *CHRNA5*, race, gender, and age in predicting smoking quantity . "NN" and "FNN" refer to the neural networks method and the functional neural networks method, while the number "1" and "2" indicate the number of hidden layers.

calculating the testing MSE. To avoid chance finding due to random splitting, I repeated the procedure 100 times and obtained the average MSE values.

As is shown from Figure 5, the FNN method attains higher prediction accuracy than FLM and NN in terms of testing MSE. In this analysis, I evaluated NN and FNN with one and two hidden layers. While adding additional layers could allow us to model more abstract and complex features from the data, the result suggests that the models with one hidden layer are sufficient to model the relationship between *CHRNA5*, race, gender, age and smoking quantity. Among all the five models, the best model is FNN with one hidden layer.

2.4.2 The role of *APOE* in predicting hippocampus volume change over time

Accurately identifying high-risk Alzheimer's Disease (AD) individuals at an early stage is important for early AD prevention, as treatments prior to the onset of dementia can ensure intervention occurs before irreversible neuronal death. In this application, I study the role of *APOE* in predicting hippocampus volume change over time, an important indicator for AD.

Data used in the application were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu)([28]). The ADNI was launched in 2003 as a public-private partnership. The primary goal of ADNI is to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease. It has three phases: ADNI1, ADNI GO, and ADNI2. ADNI includes standardized diagnostic assessments of AD (i.e., Case vs. Control). DNA samples were obtained from 808 ADNI participants and were sent to Illumina where non-CLIA whole-genome sequencing (WGS) was performed on each sample. MR image data (e.g., hippocampus volumes), and clinical data (e.g., biospecimen and cognitive tests) were also collected both at the baseline and through follow-up visits.

I downloaded and reformatted the analysis-ready ADNI dataset. Prior to the data analysis, I re-assessed the quality of the sequencing data. Markers and samples with a low proportion of successful genotype calls were removed from the analysis. After a careful quality assessment, I selected all SNVs in the *APOE* gene and evaluated its role in predicting hippocampus volume change over time (i.e., the phenotype is a longitudinal phenotype).

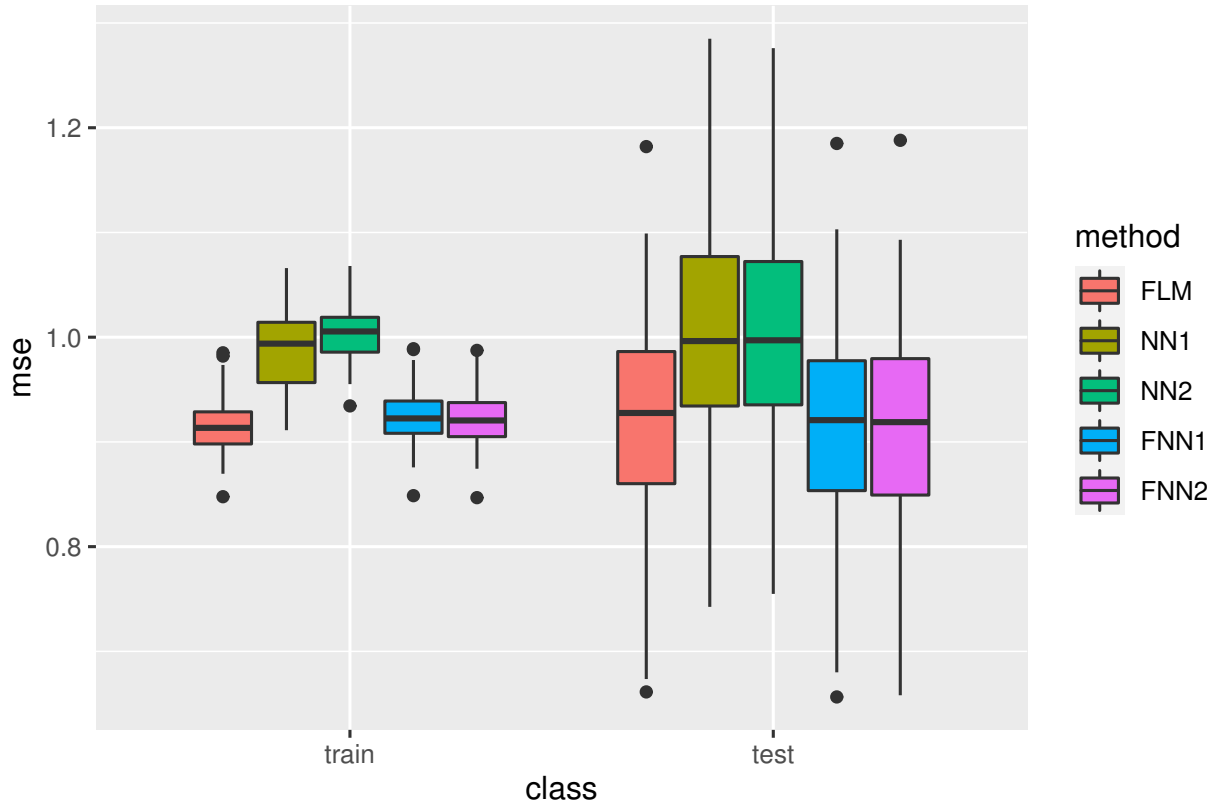

Figure 2.6: Prediction accuracy of hippocampus volume change over time based on the gene *APOE*. "NN" and "FNN" refer to the neural networks method and functional neural networks method, respectively. The number "1" or "2" indicates the number of hidden layers.

After removing samples with only one visit, I have 547 samples and 1434 measure of hippocampus volume over time. Our phenotype is the hippocampus volume change from the initial value. The initial and current year of investigation is known as the time points of interest. For FLM and NN, to consider time varying effects, I model the initial year and the current year as covariates in the models.

Figure 6 shows that FNN and FLM attain comparable performance, and both of them obtain higher accuracy than NN. Among all the models, FNN with 2 hidden layers has the best performance in terms of testing MSE. While there is no obvious evidence of overfitting for NN, the model built by NN could be too complicated to reflect the underlying relationship

between *APOE* and hippocampus volume change over time, leading to the sub-optimal performance.

2.5 Discussion

Ongoing GWAS and sequencing studies allow researchers to comprehensively investigate the role of a deep catalog of human genome variations in human complex diseases. Although these studies hold great promise for uncovering novel variants predisposing to human diseases, the high-dimensionality of genetic data and complex relationships between genotypes and disease phenotypes bring tremendous challenges to data analysis. To facilitate the gene discovery process, I develop an FNN model for high-dimensional genetic data analysis of complex phenotypes. FNN uses the hierarchy of the functional neural networks framework to learn complicated features from genetic data, making it feasible to capture non-linear genetic effects. As we observed from simulations and real data analyses, FNN attained better or at least comparable performance as compared to FLM and NN. When the underlying relationship between genotypes and disease phenotypes is nonlinear, both FNN and NN outperform FLM. Even when the underlying relation is linear, FNN has similar performance as FLM and attain better performance than NN.

In addition to its capacity to capture non-linear effects, FNN inherits advantages from FLM, which allows FNN to utilize the prior information of effects smoothness to improve accuracy and robustness. By modeling the effects of SNVs as a function in the form of a combination of basis functions, FNN is able to take LD information from nearby SNVs into account and reduces model complexity. Similarly, by modeling complex phenotypes as functions, FNN reduces the number of parameters, making it applicable to high-dimensional

phenotypes (e.g., matrices). Through simulations, the FNN method can have a close performance to FLM in the linear setting, where NN was subject to an overfitting issue. Even in the nonlinear cases, FNN has demonstrated its robustness against overfitting and attained better or comparable performance than NN.

FNN can be applied to different types of phenotypes (e.g., vectors and matrices), which make it useful for emerging genetic research (e.g., imaging genetics research or multi-omics research). Similar to FLM, FNN has a unique advantage for modeling complex phenotypes with temporal and spatial correlations (e.g., imaging or longitudinal phenotypes). Such an advantage has been shown in two real data applications. For instance, in the cigarette smoking application, I have also tried a traditional approach by treating age as a covariate in FNN. The result from such an approach is only slightly better than the result from NN. Therefore, utilizing the functional structure of phenotypes (i.e., considering smoking quantity as a function of age) can improve the performance of FNN significantly. Nonetheless, FNN is different from the classic functional model. Unlike FLM, FNN can handle the situation when the evaluation points are different among phenotypes (e.g., individuals' phenotype data are collected at different time points or locations), which is very common in real scenarios.

Methods have been developed to integrate the spline method into convolutional neural networks for image classification. In [8], B-splines are used for kernel construction in the convolutional layers to make the computation time independent from the kernel size, resulting in great performance in image classification. Different from the previous methods, FNN model is based on vanilla neural networks, which uses basis functions to model temporal and spatial correlations of the phenotype data (e.g., brain volumes measured over time) and LD structure of genetic data. The contribution of this chapter is thus to introduce a new tool, FNN, that addresses the ongoing challenges from high-dimensional genetic data analysis and

facilitates the gene discovery with complex phenotypes. FNN inherits advantages from both NN and FLM, which makes it capable of considering complex relationships (e.g., non-linear relationships) between genotypes and phenotypes while not substantially increasing model complexity. Therefore, compared to NN and FLM, it achieves a better bias-variance balance. These features make it ideal for analyzing high-dimensional genetic data that has a large number of SNVs, low signal-to-noise ratio, and complex relationships with phenotypes.

In this chapter, I mainly focus on introducing the FNN model and illustrating FNN with simulations and real data applications. Statistical properties of FNN (e.g., consistency and rate of convergence) are important topics that may be worth further investigation in the future work.

Chapter 3

Consistency of Functional Neural Networks

In the previous chapter, I introduced the FNN model and conducted simulations and real data analysis to evaluate its performance. While simulations and real data applications showed FNN had advantages over existing methods, it is also important to study the statistical property of FNN. In this chapter, the consistency of FNN will be proved.

In the process of proof, some lemmas and theorems are applied and make significant effects. I quote them and provide the citation when I introduce them before the application. Their proofs can be found from the provided sources. As for my own theorems, the proof is given below these theorems. The notations of the proposed model are changed compared with Chapter 2 for the consistency of quoted theorems and ease of proofs.

3.1 Introduction

Consistency is an important property of a statistical model. It determines whether the estimate can converge to the true function as the sample size n increases. As a new nonlinear function, it is also necessary to check the consistency of FNN.

In the proposed FNN, the underlying function can be any continuous function, which forms a large function space. In section 3, I prove that for any continuous function, there

exists one FNN model that is approximated to the continuous function at any given extend. However, it is uncertain that the approximate FNN model can be learnt from the given data even the FNN model exists. If the space of the candidate models is too large, it is hard to find the right one. To measure how large the function space of FNN is, I utilize the covering number to quantify the extend in section 4. In section 5, I prove that the covering number of our proposed model is not too large, and our estimated model only deviates from the best possible model at a controlled rate. Combined with all these results, we prove the consistency of FNN.

3.2 Model

First, some definitions needed in the proof are introduced.

Definition 3.2.1. *A function space $\mathcal{B} = \{B_i(t)|i \in \mathbf{N}^+\}$ is a basis function system in a compact space C, if for every $\epsilon > 0$ and every f as a continuous real-valued function defined in C, there exists a function $g(x) = \sum_{i=1}^{N} a_i B_i(t), a_i \in \mathbb{R}$ satisfying $|f(x) - g(x)| < \epsilon$ on C.*

In the following, the concept smoothness is used to restrict the function space of linear combinations of basis functions. Therefore, the basis functions must be differentiable. In chapter 2, B-spline basis functions are utilized as a basis function system, which are differentiable. Moreover, we can select one B-spline basis function, which is exactly a order differentiable. By Stone-Weierstrass theorem, monomial functions also compose a basis function system. Other common basis function systems, such as the monomial basis or Fourier basis, are also differentiable on a compact space.

Based on the basis function system, I can define a function-to-function operator.

Definition 3.2.2. *Let $x(t)$ be a function in $L^2([0,1])$. Define an operator $M(\cdot) : L^2([0,1]) \to L^2([0,1])$ as $M(x(t)) = \int x(t)\alpha(s,t)\mathrm{d}t + \beta(s)$, where $\alpha(s,t)$ and $\beta(s)$ are linear combinations of basis functions.*

To write in a recursive pattern, denote that $x(t) = x_0(t_0)$ and

$$\begin{aligned} y_i(t_i) &= M_i \circ x_{i-1}(t_{i-1}) = M_i(x_{i-1}(t_{i-1})) \\ &= \int x_{i-1}(t_{i-1})\alpha_i(t_i, t_{i-1})\mathrm{d}t_{i-1} + \beta_i(t_i), \end{aligned} \tag{3.1}$$

where

$$\alpha_i(t_i, t_{i-1}) = \sum_{j=1}^{J_i} \sum_{k=1}^{J_{i-1}} W_{i,j,k} B_{i-1,k}(t_{i-1}) B_{i,j}(t_i),$$

$$\beta_i(t_i) = \sum_{j=1}^{J_i} b_{i,j} B_{i,j}(t_i),$$

and $\{B_{i,j}(t) | j \in \mathbf{N}^+\}$ is a predetermined basis function system for each i, and $W_{i,j,k}$ and $b_{i,j}$ are parameters of the operator.

Definition 3.2.3. *A function $\Psi : \mathbf{R} \to [0,1]$ is called a squashing function if it is non-decreasing,* $\lim_{\lambda \to \infty} = 1$ *and* $\lim_{\lambda \to -\infty} = 0$.

If $x(t)$ is a function, $\Psi \circ x(t) = \Psi(x(t))$. To express in a recursive way, I have

$$x_i(t_i) = \Psi(y_i(t_i)).$$

The $x_i(t_i)$ and the $y_i(t_i)$ refers to the same object as in 3.1, therefore I can construct a

model by connecting these operator:

Definition 3.2.4. *A model* $F : L^2([0,1]) \times [0,1] \to \mathbf{R}$ *is called FNN if it satisfies the form,*

$$F(x_0(t_0), t_d) = y_d(t_d) = M_d \circ \Psi \circ M_{d-1} \circ \Psi \circ \cdots M_2 \circ \Psi \circ M_1 \circ x_0(t_0), \tag{3.2}$$

where d *is the number of hidden layers in FNN.*

The function space of all possible FNN satisfying the above form is denoted by $F_d(\Psi)$. In the following, I use H to denote the model domain $L^2([0,1]) \times [0,1]$.

Definition 3.2.5. *For functions defined on* $L^2([0,1])$, $x(t)$ *and* $y(t)$ *are considered as in the same equivalence class if* $\int_0^1 |x(t) - y(t)| \mathrm{d}t = 0$.

The distance of elements in space H is defined by

$$d(h_i, h_j) = \int_0^1 |x_i(t) - x_j(t)| \mathrm{d}t + |s_i - s_j|,$$

where $h_i = (x_i(t), s_i), h_j = (x_j(t), s_j) \in H$.

Remark 3.2.1. *Normally random variable are defined in a subset of Euclidean space, while the domain of the variable in the proposed FNN model involves a function space* $L^2([0,1])$. *There comes an issue on whether the definition of random variable is valid. From the section on convergence of sample distribution in [23], random variable is constructed on any separable metric space. Therefore the domain* H *is meaningful in terms of the space of random variable since* $L^2([0,1])$ *is separable.*

3.3 Universal Approximation Theorem

Hornik [12] proved that multi-layer forward neural networks are universal approximators in 1989. In this section, I prove that FNN is also a universal approximator based on approximations.

To outline the proof, I first construct another function space $G_d(\Psi)$, and then prove that $G_d(\Psi)$ is dense in all real continuous function defined on H following the idea in [12]. Finally, that $F_d(\Psi)$ is dense is proved by showing that any function in $G_d(\Psi)$ can be approximated by functions in $F_d(\Psi)$.

3.3.1 Definitions for New Construction

Definition 3.3.1. *Let $x(t)$ be a function on $L^2([0,1])$. Define an operator $L(\cdot)$ as $L(x(t)) = \int x(t)\alpha(s,t)\mathrm{d}t + \beta(s)$, where $\alpha(s,t)$ and $\beta(s)$ are step functions.*

To express in a recursive pattern, I have

$$\begin{aligned} y_i(t_i) &= L_i \circ x_{i-1}(t_{i-1}) = L_i(x_{i-1}(t_{i-1})) \\ &= \int x_{i-1}(t_{i-1})\alpha_i(t_i, t_{i-1})\mathrm{d}t_{i-1} + \beta_i(t_i), \end{aligned}$$

where

$$\begin{aligned} \alpha_i(t_i, t_{i-1}) &= \sum_{j=1}^{J_i} \sum_{k=1}^{J_{i-1}} W_{i,j,k}\chi_{E_{i-1,k}}(t_{i-1})\chi_{E_{i,j}}(t_i), \\ \beta_i(t_i) &= \sum_{j=1}^{J_i} b_{i,j}\chi_{E_{i,j}}(t_i). \end{aligned}$$

Among them, W_i is a matrix where the j-th row and k-th column element is $W_{i,j,k}$. b_i is

a vector where the element in the j-th position is $b_{i,j}$. $E(\cdot)$ are sub intervals of $[0,1]$, where $E_{i,j} \cap E_{i,j'} = \emptyset$ if $j \neq j'$. χ_E is a characteristic function satisfying:

$$\chi_E(t) = \begin{cases} 1 & \text{if } x \in E \\ \\ 0 & \text{if } x \notin E. \end{cases}$$

For every set E, denote that $aE + b = \{ax + b | x \in E\}$

Definition 3.3.2. *The function space $G_d(\Psi)$ is a set of all possible functions f of the the following form:*

$$f(x_0(t_0), t_d) = y_d(t_d) = L_d \circ \Psi \circ L_{d-1} \circ \Psi \circ \dots L_2 \circ \Psi \circ L_1 \circ x_0(t_0).$$

An operator $L(\cdot)$ whose domain is a set of measurable functions is bounded by λ if $|L(x(t)) - L(y(t))| \leq \lambda \int |x(t) - y(t)| \mathrm{d}t$.

3.3.2 Step One

Recall that a family A of real functions defined on a set E is an algebra if A is closed under addition, multiplication, and scalar multiplication.

A family A separates points on E if, for every x, y in E, $x \neq y$, there exists a function f in A such that $f(x) \neq f(y)$.

The family A vanishes at no point of E if, for each x in E, there exists f in A such that $f(x) \neq 0$.

Theorem 3.3.1. *(Stone-Weierstrass Theorem.) Let A be an algebra of real continuous functions on a compact set K. If A separates points on K and A vanishes at no point of K,*

then the uniform closure B of A consists of all real continuous functions on K.

In this chapter, the term $C(H)$ is used to denote all real continuous functions defined on H.

Definition 3.3.3. *A subset S of a metric space (X, ρ) is ρ - dense in a subset T if, for every $\epsilon > 0$ and every $t \in T$, there is an $s \in S$ such that $\rho(s, t) < \epsilon$.*

Definition 3.3.4. *A subset A of C is said to be uniformly dense on compacta in C if, for every compact subset K which is subset of the domain of C, A is ρ_K - dense in C, where $\rho_K(f, g) = \sup_{x \in K} |f(x) - g(x)|$.*

By the definition, A is uniformly dense on compacta in C has the same meaning with that the uniform closure B of A consists of all real continuous functions on K.

In order to apply the Stone-Weierstrass theorem on $G_d(\Psi)$, I have to show that $G_d(\Psi)$ is an algebra.

Lemma 3.3.1. *$G_d(\Psi)$ is an algebra on H if Ψ is the cosine function.*

Proof. It is necessary to show that $G_d(\Psi)$ is closed under addition, multiplication, and scalar multiplication on C if Ψ is the cosine function.

(i) $G_d(\Psi)$ is closed under the scalar multiplication:

$\forall f \in G_d(\Psi)$, λf can be represented by f in which W_d and b_d are replaced by λW_d and λb_d without changing any other parameters.

(ii) $G_d(\Psi)$ is closed under addition:

It is clear that $G_1(\Psi)$ is closed under addition since it is a simple functional linear regression.

$\forall f, \tilde{f} \in G_d(\Psi)$, I need to prove that $f + \tilde{f} \in G_d(\Psi)$.

For $\tilde{f}$, the corresponding functions $\alpha_i(t_i, t_{i-1})$ and $\beta_i(t_i)$ are denoted by $\tilde{\alpha}_i(t_i, t_{i-1})$ and $\tilde{\beta}_i(t_i)$.

The function which equals to $f + \tilde{f}$ can be constructed with the following coefficient functions including $\widehat{\alpha}_i$ and $\widehat{\beta}_i$ For $i = 1$,

$$\widehat{\alpha}_1\ (t_1, t_0) := \alpha_1(2t_1, t_0) + \tilde{\alpha}_1(2t_1 - 1, t_0),$$

$$\widehat{\beta}_1\ (t_1) := \beta_1(2t_1) + \tilde{\beta}_1(2t_1 - 1).$$

For $1 < i < d$,

$$\widehat{\alpha}_i\ (t_i, t_{i-1}) := 2\alpha_i(2t_i, 2t_{i-1}) + 2\tilde{\alpha}_i(2t_i - 1, 2t_{i-1} - 1),$$

$$\widehat{\beta}_i\ (t_i) := \beta_i(2t_i) + \tilde{\beta}_i(2t_i - 1).$$

For $i = d$,

$$\widehat{\alpha}_d\ (t_d, t_{d-1}) := \alpha_i(t_d, 2t_{d-1}) + \tilde{\alpha}_d(t_d, 2t_{d-1} - 1),$$

$$\widehat{\beta}_d\ (t_d) := \beta_d(2t_d) + \tilde{\beta}_d(2t_d - 1).$$

(iii) $G_d(\Psi)$ is closed under multiplication:

$\forall f, \tilde{f} \in G_d(\Psi)$, $y_{d-1}(t_{d-1}), \tilde{y}_{d-1}(t_{d-1})$ are the corresponding object in the previous layer, i.e.,

$$f(x(t_0), t_d) = \int \alpha_d(t_d, t_{d-1})\Psi(y_{d-1}(t_{d-1}))\mathrm{d}t_{d-1} + \beta_d(t_d)$$

$$\tilde{f}(\tilde{x}(t_0), t_d) = \int \tilde{\alpha}_d(t_d, t_{d-1})\Psi(\tilde{y}_{d-1}(t_{d-1}))\mathrm{d}t_{d-1} + \tilde{\beta}_d(t_d).$$

For $\tilde{f}$, the corresponding parameters $W_{i,j,k}, b_{i,j}$ and $E_{i,j}$ are denoted by $\tilde{W}_{i,j,k}, \tilde{b}_{i,j}$ and $\tilde{E}_{i,j}$. Since $G_d(\Psi)$ is closed in addition, it satisfies to check each component of $f \cdot \tilde{f}$ in the following:

$$\begin{aligned}
&\beta_d(t_d) \cdot \tilde{\beta}_d(t_d) \\
=&\sum_{j=1}^{J_d} b_{d,j}\chi_{E_{d,j}}(t_d) \cdot \sum_{j=1}^{\tilde{J}_d} \tilde{b}_{d,j}\chi_{\tilde{E}_{d,j}}(t_d) \\
=&\sum_{j=1}^{J_d}\sum_{j'=1}^{\tilde{J}_d} b_{d,j}\tilde{b}_{d,j'}\chi_{E_{d,j}\cap\tilde{E}_{d,j'}}(t_d),
\end{aligned}$$

satisfies the form of $\beta_d(t_d)$.

$$\begin{aligned}
&\tilde{\beta}_d(t_d) \cdot \int \alpha_d(t_d, t_{d-1})\Psi(y_{d-1}(t_{d-1}))\mathrm{d}t_{d-1} \\
=&\int \alpha_d(t_d t_{d-1})\tilde{\beta}_d(t_d)\Psi(y_{d-1}(t_{d-1}))\mathrm{d}t_{d-1} \\
=&\int \sum_{j=1}^{J_d}\sum_{k=1}^{J_{d-1}} W_{d,j,k}\chi_{E_{d-1,k}}(t_{d-1})\chi_{E_{d,j}}(t_d) \cdot \sum_{j=1}^{\tilde{J}_d} \tilde{b}_{d,j}\chi_{\tilde{E}_{d,j}}(t_d)\Psi(y_{d-1}(t_{d-1}))\mathrm{d}t_{d-1} \\
=&\int \sum_{k=1}^{J_{d-1}}\sum_{j=1}^{J_d}\sum_{j'=1}^{\tilde{J}_d} W_{d,j,k}\tilde{b}_{d,j'}\chi_{E_{d-1,j}}(t_{d-1})\chi_{E_{d,j}\cap\tilde{E}_{d,j'}}(t_d)\Psi(y_{d-1}(t_{d-1}))\mathrm{d}t_{d-1},
\end{aligned}$$

satisfies the form of $\int \alpha_d(t_d, t_{d-1})\Psi(y_{d-1}(t_{d-1}))\mathrm{d}t_{d-1}$.

In the same manner, it is easy to show that $\beta_d(t_d) \cdot \int \tilde{\alpha}_d(t_d, t_{d-1})\Psi(\tilde{y}_{d-1}(t_{d-1}))\mathrm{d}t_{d-1}$ satisfies the form of $\int \tilde{\alpha}_d(t_d, t_{d-1})\Psi(\tilde{y}_{d-1}(t_{d-1}))\mathrm{d}t_{d-1}$.

By using the step function decomposition, I have,

$$y_{d-1}(t_{d-1}) = \sum_{k=1}^{J_{d-1}} c_k \chi_{E_{d-1,k}}(t_{d-1}),$$

$$\tilde{y}_{d-1}(t_{d-1}) = \sum_{k=1}^{\tilde{J}_{d-1}} \tilde{c}_k \chi_{\tilde{E}_{d-1,k}}(t_{d-1}),$$

where

$$\begin{aligned}
&y_{d-1}(t_{d-1}) \\
&= \int x_{d-2}(t_{d-2}) \sum_{k=1}^{J_{d-2}} \sum_{j=1}^{J_{d-1}} \chi_{E_{d-1,j}}(t_{d-1}) \chi_{E_{d-2,k}}(t_{d-2}) \mathrm{d}t_{d-2} + \sum_{j=1}^{J_{d-1}} b_{d-1,j} \chi_{E_{d-1,j}}(t_{d-1}) \\
&= \sum_{j=1}^{J_{d-1}} (\sum_{k=1}^{J_{d-2}} \int x_{d-2}(t_{d-2}) \chi_{E_{d-2,k}}(t_{d-2}) \mathrm{d}t_{d-2} + b_{d-1,j}) \chi_{E_{d-1,j}}(t_{d-1}).
\end{aligned}$$

Define $|E| = \int \chi_E(s) \mathrm{d}s$. By changing the parameter $E_{d-1,j}$ to be independent with $\tilde{E}_{d-1,j}$ without changing its length, the condition $|E_{d-1,j}||E_{d-1,j'}| = |E_{d-1,j} \cap E_{d-1,j'}|$ is satisfied. The transformed $y_{d-1}(t_{d-1})$ with the new $E_{d-1,j}$ still satisfies the form of $G_{d-1}(\Psi)$. I used the transformed $y_{d-1}(t_{d-1})$ in the following proofs without altering its form.

$$\begin{aligned}
&\int \alpha_d(t_d, t_{d-1}) \Psi(x(t_{d-1})) \mathrm{d}t_{d-1} \\
=& \int \sum_{j=1}^{J_d} \sum_{k=1}^{J_{d-1}} W_{d,j,k} \chi_{E_{d-1,k}}(t_{d-1}) \chi_{E_{d,j}}(t_d) \cdot \Psi(\sum_{k=1}^{J_{d-1}} c_k \chi_{E_{d-1,k}}(t_{d-1})) \mathrm{d}t_{d-1} \\
=& \sum_{j=1}^{J_d} \sum_{k=1}^{J_{d-1}} \Psi(c_k) W_{d,j,k} |E_{d-1,k}| \chi_{E_{d,j}}(t_d),
\end{aligned}$$

where Ψ is the cos function that satisfies $\Psi(x)\Psi(y) = (\Psi(x+y) + \Psi(x-y))/2$.

$$
\begin{aligned}
&\int \alpha_d(t_d, t_{d-1})\Psi(x(t_{d-1}))\mathrm{d}t_{d-1}] \cdot \int \tilde{\alpha}_2(t_d, t_{d-1})\Psi(\tilde{x}(t_{d-1}))\mathrm{d}t_{d-1} \\
=&\sum_{j=1}^{J_d}\sum_{k=1}^{J_{d-1}} \Psi(c_k)W_{d,j,k}|E_{d-1,k}|\chi_{E_{d,j}}(t_d)\cdot \\
&\sum_{j'=1}^{\tilde{J}_d}\sum_{k'=1}^{\tilde{J}_d} \Psi(\tilde{c}_{k'})W_{d,j',k'}|E_{d-1,k'}|\chi_{E_{d,j'}}(t_d) \\
=&\sum_{j=1}^{J_d}\sum_{k=1}^{J_{d-1}}\sum_{j'=1}^{\tilde{J}_d}\sum_{k'=1}^{\tilde{J}_{d-1}} W_{d,j,k}W_{d,j',k'}|E_{d-1,k}||E_{d-1,k'}|\Psi(c_k)\Psi(\tilde{c}_{k'})\chi_{E_{d,j}\cap\tilde{E}_{d,j'}}(t_d) \\
=&\sum_{j=1}^{J_d}\sum_{k=1}^{J_{d-1}}\sum_{j'=1}^{\tilde{J}_d}\sum_{k'=1}^{\tilde{J}_{d-1}} \frac{1}{2}W_{d,j,k}W_{d,j',k'}|E_{d-1,k}\cap E_{d-1,k'}| \\
&(\Psi(c_k+\tilde{c}_{k'}) + \Psi(c_k-\tilde{c}_{k'}))\chi_{E_{d,j}\cap\tilde{E}_{d,j'}}(t_d) \\
=&\sum_{j=1}^{J_d}\sum_{k=1}^{J_{d-1}}\sum_{j'=1}^{\tilde{J}_d}\sum_{k'=1}^{\tilde{J}_{d-1}} \frac{1}{2}W_{d,j,k}W_{d,j',k'}|E_{d-1,k}\cap E_{d-1,k'}|\Psi(c_k+\tilde{c}_{k'})\chi_{E_{d,j}\cap\tilde{E}_{d,j'}}(t_d) \\
+&\sum_{j=1}^{J_d}\sum_{k=1}^{J_{d-1}}\sum_{j'=1}^{\tilde{J}_d}\sum_{k'=1}^{\tilde{J}_{d-1}} \frac{1}{2}W_{d,j,k}W_{d,j',k'}|E_{d-1,k}\cap E_{d-1,k'}|\Psi(c_k-\tilde{c}_{k'})\chi_{E_{d,j}\cap\tilde{E}_{d,j'}}(t_d).
\end{aligned}
$$

Since $x(t_{d-1}), \tilde{x}(t_{d-1}) \in G_{d-1}(\Psi)$ and $G_{d-1}(\Psi)$ is closed in addition and scalar multiplication, $(x(t_{d-1}) + \tilde{x}(t_{d-1})), (x(t_{d-1}) - \tilde{x}(t_{d-1})) \in G_{d-1}(\Psi)$.

$$
\begin{aligned}
&\int \widehat{\alpha}_d\,(t_d, t_{d-1})\Psi(x(t_{d-1}) + \tilde{x}(t_{d-1}))\mathrm{d}t_{d-1} \\
=&\sum_{j=1}^{\widehat{J}_d}\sum_{k=1}^{J_{d-1}}\sum_{k'=1}^{\tilde{J}_{d-1}} \Psi(c_k+\tilde{c}_{k'})\,\widehat{W}_{d,j,k}\,|E_{d-1,k}\cap\tilde{E}_{d-1,k'}|\chi_{\widehat{E}_{d,j}}(t_d).
\end{aligned}
$$

By choosing $\widehat{E}_{d,j} := E_{d,j}\cap\tilde{E}_{d,j'}$, $\widehat{W}_{d,j,K} := \frac{1}{2}W_{d,j,k}W_{d,j',k'}$, the first term satisfies

the form of $\int \widehat{\alpha}_d\ (t_d, t_{d-1})\Psi(\widehat{y}_{d-1}\ (t_{d-1}))\mathrm{d}t_{d-1}$. The same condition applies to $x(t_{d-1}) - \tilde{x}(t_{d-1})$. Since $G_d(\Psi)$ is closed under addition, the whole component belongs to $G_d(\Psi)$.

□

By using the Stone-Weierstrass theorem, if $G_d(\Psi)$ separates points on K for any compact set $K \subseteq H$ and vanishes at no point, I can have the following theorem.

Theorem 3.3.2. *$G_d(\Psi)$ is uniformly dense on compacta in $C(H)$ if G is the cosine function.*

Proof. It is easy to show that $G_d(\Psi)$ vanishes at no point by using $\beta_d(t_d) = 1$ while keeping other functions as 0.

If $\hat{t}_d \neq \tilde{t}_d$, we can assume that $\hat{t}_d < \tilde{t}_d$ without loss of generality. The points can be separated by using $\beta_d(t_d) := \chi_{[\frac{\hat{t}_d + \tilde{t}_d}{2}, 1]}(t_d)$, while keeping other functions as 0.

It is sufficient to show that $\forall x(t_0), \tilde{x}(t_0) \in C, \exists f \in G_d(\Psi)$ s.t. $f(x(t_0), t_d) \neq f(\tilde{x}(t_0), t_d)$ if $\int_0^1 |x(t_0) - \tilde{x}(t_0)| > 0$. Choose

$$\alpha_1(t_1, t_0) := \frac{1}{c}(\chi_{x(t_0) > \tilde{x}(t_0)} - \chi_{x(t_0) < \tilde{x}(t_0)})\chi_{[0,1]}(t_1),$$

$$\beta_1(t_1) := -\frac{1}{c}(\int_{x(t_0) > \tilde{x}(t_0)} x(t_0)\mathrm{d}t_0 - \int_{x(t_0) < \tilde{x}(t_0)} x(t_0)\mathrm{d}t_0)\chi_{[0,1]}(t_1),$$

where $c = \int_{x(t_0) > \tilde{x}(t_0)} x(t_0) - \tilde{x}(t_0)\mathrm{d}t_0 - \int_{x(t_0) < \tilde{x}(t_0)} x(t_0) - \tilde{x}(t_0)\mathrm{d}t_0$.

Therefore $L_1(x(t_0)) = 0$ and $L_1(\tilde{x}(t_0)) = -\chi_{[0,1]}(t_1)$. For $i > 1$, choose

$$\alpha_i(t_i, t_{i-1}) := \frac{2 + 2e}{e - 1}\chi_{[0,1]}(t_i)\chi_{[0,1]}(t_{i-1}),$$

$$\beta_i(t_i) := \frac{e + 1}{e - 1}\chi_{[0,1]}(t_i).$$

Therefore $f(x(t_0), t_d) = 0$ and $f(\tilde{x}(t_0), t_d) = -1$, i.e., $G_d(\Psi)$ separates points on compact function space C. □

The following lemma is quoted from the Lemma A.3 in [12].

Lemma 3.3.2. *For every squashing function Ψ, $\epsilon > 0$, and $M > 0$, there is a function $\sigma(x)$ that satisfies $\sigma(x) = \sum_{j=1}^{q} b_j \Psi(a_j x + c_j); b_j, a_j, c_j \in \mathbb{R}, q \in \mathbb{N}^+$ such that*

$$\sup_{x \in [-M, M]} |\sigma(x) - \cos(x)| < \epsilon.$$

The above lemma shows that the cosine function can be approximated by any squashing function with some linear transformations. Based on this lemma, the squashing function Ψ is not limited to the cosine function.

Theorem 3.3.3. *$G_d(\Psi)$ is uniformly dense on compacta in $C(H)$ if Ψ is a squashing function.*

Proof. For each given $f \in G_d(\Psi)$, a common bound $B > 1$ can be found for each operator $L_i(\cdot)$.

For any given compact set $K \subseteq H$, there exists M s.t. the values inside the cosine function are within $[-M, M]$. I will compare the squashing function $\sigma(\cdot)$ defined in the previous lemma and the cosine function.

In each step, there exists the difference between the inside functions, which we distinguish by $\tilde{y}_i(t)$ and $\widehat{y}_i\,(t)$.

For each $\delta > 0$, choose $\epsilon = \delta(B-1)/B^d$,

$$
\begin{aligned}
&|\sigma(\tilde{y}_i(t_i)) - \cos(\widehat{y}_i\ (t_i))| \\
\leq &|\sigma(\tilde{y}_i(t_i)) - \cos(\tilde{y}_i(t_i))| + |\cos(\tilde{y}_i(t_i)) - \cos(\widehat{y}_i\ (t_i))| \\
\leq &\delta(B-1)/B^d + |\tilde{y}_i(t_i) - \widehat{y}_i\ (t_i)| \\
\leq &\delta(B-1)/B^d + B \max_{0 \leq t_{i-1} \leq 1} |\tilde{x}_{i-1}(t_{i-1}) - \widehat{x}_{i-1}\ (t_{i-1})|.
\end{aligned}
$$

Suppose that $\tilde{f}$ uses the squashing function σ and $\widehat{f}$ uses the cosine function, we have

$$
\begin{aligned}
&|\tilde{f}(x_0(t_0), t_d) - \widehat{f}\ (x_0(t_0), t_d)| \\
= &|L_d(\sigma(\tilde{y}_{d-1}(t_{d-1}))) - L_d(\cos(\widehat{y}_{d-1}\ (t_{d-1}))|) \\
\leq &B \max_{0 \leq t_{d-1} \leq 1} |\sigma(\tilde{y}_{d-1}(t_{d-1})) - \cos(\widehat{y}_{d-1}\ (t_{d-1}))| \\
\leq &\delta(B-1)/B^{d-1} + B \max_{0 \leq t_{d-2} \leq 1} |L_{d-1}(\sigma(\tilde{y}_{d-2}(t_{d-2}))) - L_{d-1}(\cos(\widehat{y}_{d-2}\ (t_{d-2}))| \\
\leq &\delta(B-1) \sum_{i=1}^{d-1} 1/B^i \leq \delta.
\end{aligned}
$$

It satisfies to verify that $\tilde{f} \in G_d(\Psi)$.

Since $a_j \tilde{L_1} + c_j$ is still an operator of $L(\cdot)$, it is sufficient to show that

$\tilde{L}_{i+1}(\sum_{j=1}^{q} b_j \Psi(\tilde{L}_{i,j}(x_{i-1}(t_{i-1}))))$ can be rewritten in the form $L_{i+1}(\Psi(L_i(x_{i-1}(t_{i-1}))))$.

Let

$$
\alpha_i(t_i, t_{i-1}) := \sum_{j=1}^{q} \tilde{\alpha}_i(qt_i + (1-j), t_{i-1}),
$$

$$
\beta_i(t_i) := \sum_{j=1}^{q} \tilde{\beta}_i(qt_i + (1-j)),
$$

$$\alpha_{i+1}(t_{i+1}, t_i) := \sum_{j=1}^{q} q b_j \tilde{\alpha}_{i+1}(t_{i+1}, qt_i + (1-j)),$$

$$\beta_i(t_{i+1}) := \tilde{\beta}_{i+1}(t_{i+1}).$$

we have $f \in G_d(\Psi)$ that satisfies $f = \tilde{f}$, i.e.,

$$|\tilde{f}(x_0(t_0), t_d) - \widehat{f}\ (x_0(t_0), t_d)| \leq \delta.$$

$G_d(\Psi)$ is therefore uniformly dense on compacta in $C(H)$. □

3.3.3 Step Two

I want to replace the step functions in $G_d(\Psi)$ with the linear combinations of basis functions in $F_d(\Psi)$. With the knowledge of real analysis, every measurable function is almost a continuous function. By the definition of a basis function system, every continuous function can be approximated by linear combinations of basis functions. The conclusions are proceeded by these approximations.

I quote the Lusin theorem below. The proof of theorem can be found in book [29].

Theorem 3.3.4. *(Lusin Theorem.) Suppose f is measurable and finite valued on E with E of finite measure, for every $\epsilon > 0$, there exists a closed set F_ϵ with*

$$F_\epsilon \subseteq E \quad \text{and} \quad m(E - F_\epsilon) \leq \epsilon$$

such that $f|_{F_\epsilon}$ is continuous.

Denote $\widehat{G}_d(\Psi)$ is the same as $G_d(\Psi)$ except that the step functions in $G_d(\Psi)$ are replaced by continuous functions. If $\widehat{G}_d(\Psi)$ is ρ - dense in $G_d(\Psi)$, I have the following theorem.

Theorem 3.3.5. *$\widehat{G}_d(\Psi)$ is uniformly dense on compacta in $C(H)$ if Ψ is a Lipschitz continuous squashing function.*

Proof. It is sufficient to prove that for every $\delta > 0$, any compact set $K \subseteq H$ and any $f \in G_d(\Psi)$,

$$\exists \widehat{f} \in \widehat{G}_d(\Psi), \quad \text{s.t.} \quad |\widehat{f}(x_0(t_0), t_d) - f(x_0(t_0), t_d)| < \delta,$$

for all $(x_0(t_0), t_d) \in K$. Since K is compact, there exists $E > 1$ and $|x_0(t_0)| < E$.

A common bound $B > 1$ can be found for each operator $L_i(\cdot)$ in function f.

By the Lusin Theorem, $\forall \epsilon > 0$, there exists $F_{i,\epsilon}$ and $F'_{i,\epsilon}$ s.t.

$$\widehat{\alpha}_i(t_i, t_{i-1})\chi_{F_{i,\epsilon}} = \alpha_i(t_i, t_{i-1})\chi_{F_{i,\epsilon}},$$

$$\widehat{\beta}_i(t_i)\chi_{F'_{i,\epsilon}} = \beta_i(t_i)\chi_{F'_{i,\epsilon}}.$$

Find a common bound D for $\alpha_i(t_i, t_{i-1})$ and $\beta_i(t_i)$.

The squashing function Ψ is Lipschitz continuous, therefore there exists $L > 1$ s.t.

$$|\Psi(x) - \Psi(y)| \leq L|x - y|.$$

And the difference between each functional linear transformation can also be bounded:

$$\begin{aligned}
&|L_{i+1}(x_i(t_i)) - \widehat{L}_{i+1}(\widehat{x}_i(t_i))| \\
\leq &|L_{i+1}(x_i(t_i)) - L_{i+1}(\widehat{x}_i(t_i))| + |L_{i+1}(\widehat{x}_i(t_i)) - \widehat{L}_{i+1}(\widehat{x}_i(t_i))| \\
\leq &B \int_0^1 |x_i(t_i) - \widehat{x}_i(t_i)| \mathrm{d}t_i + 4D(E+1)\epsilon \\
\leq &B \max_{0 \leq t_i \leq 1} |x_i(t_i) - \widehat{x}_i(t_i)| + 4D(E+1)\epsilon,
\end{aligned}$$

Combined the two inequalities and recursive representation of the two functions,

$$
\begin{aligned}
&|f(x_0(t_0), t_d) - \widehat{f}(x_0(t_0), t_d)| \\
=&|L_d(x_{d-1}(t_{d-1})) - \widehat{L}_d(\widehat{x}_{d-1}(t_{d-1}))|) \\
\leq& 4D(E+1)\epsilon + BL \max_{0 \leq t_{d-1} \leq 1} |L_{d-1}(x_{d-2}(t_{d-2})) - \widehat{L}_{d-1}((\widehat{x}_{d-2}(t_{d-2}))| \\
\leq& 4D(E+1)\epsilon \sum_{i=0}^{d-1} (BL)^i \leq 4D\epsilon(BL)^d/(BL-1).
\end{aligned}
$$

$|\widehat{f}(x(t)) - f(x(t))| < \delta$ is satisfied if $\epsilon < \delta(BL-1)/4D(E+1)(BL)^d$. □

Following the same strategy of the last proof, I can show that $F_d(\Psi)$ is ρ - dense in $\widehat{G}_d(\Psi)$ and therefore $F_d(\Psi)$ is a universal approximator.

Theorem 3.3.6. *$F_d(\Psi)$ is uniformly dense on compacta in $C(H)$ if Ψ is Lipschitz continuous squashing function.*

Proof. It is sufficient to show that for every $\delta > 0$, any compact set $K \subseteq H$ and any $f \in G_d(\Psi)$, $\widehat{f} \in \widehat{G}_d(\Psi)$,

$$\exists f \in F_d(\Psi), \quad s.t. \quad |f(x_0(t_0), t_d) - \widehat{f}(x_0(t_0), t_d)| < \delta,$$

for all $(x_0(t_0), t_d) \in K$. Since K is compact, there exists $E > 1$ and $|x_0(t_0)| < E$.
Based on the property of basis function, $\forall \epsilon > 0$, we have

$$\alpha_i(t_i, t_{i-1}) - \widehat{\alpha}_i(t_i, t_{i-1}) \leq \epsilon,$$

$$\beta_i(t_i) - \widehat{\beta}_i(t_i) \leq \epsilon.$$

A common bound $B > 1$ can be found for each operator $L_i(\cdot)$ in function f.

$$
\begin{aligned}
&|M_{i+1}(x_i(t_i)) - \widehat{L}_{i+1}(\widehat{x}_i(t_i))| \\
\leq &|M_{i+1}(x_i(t_i)) - \widehat{L}_{i+1}(x_i(t_i))| + |\widehat{L}_{i+1}(x_i(t_i)) - \widehat{L}_{i+1}(\widehat{x}_i(t_i))| \\
\leq &\epsilon(1+E) + B \int |x_i(t_i) - \widehat{x}_i(t_i)| \mathrm{d}t_i \\
\leq &\epsilon(1+E) + B \max_{0 \leq t_i \leq 1} |x_i(t_i) - \widehat{x}_i(t_i)|.
\end{aligned}
$$

Suppose that the squashing function Ψ is L-Lipschitz continuous,

$$
\begin{aligned}
&|f(x_0(t_0), t_d) - \widehat{f}(x_0(t_0), t_d)| \\
= &|M_d(x_{d-1}(t_{d-1})) - \widehat{L}_d(\widehat{x}_{d-1}(t_{d-1}))|) \\
\leq &\epsilon(1+E) + LB \max_{0 \leq t_{d-1} \leq 1} |M_{d-1}(x_{d-2}(t_{d-2})) - \widehat{L}_{d-1}((\widehat{x}_{d-2}(t_{d-2}))| \\
\leq &\epsilon(1+E) \sum_{i=0}^{d-1} (LB)^i \leq \epsilon(1+E)(LB)^d/(LB-1).
\end{aligned}
$$

$|\widetilde{f}(x(t)) - \widehat{f}((x(t)))| < \delta$ is satisfied if $\epsilon < \delta(LB-1)/(1+E)(LB)^d$. □

3.4 Covering Number

3.4.1 Smooth Functions

I introduce a class of functions on a bounded set χ_d in $\mathbb{R}^d$ quoted from [31]. A differential operator D^k for any vector $k = (k_1, \cdots, k_d)$ is defined as:

$$D^k = \frac{\partial^{k.}}{\partial x_1^{k_1} \cdots \partial x_d^{k_d}},$$

where $k. = \sum_{i=1}^{d} k_i$. Then, for a function $f : \chi_d \to \mathbb{R}$,

$$\|f\|_a = \max_{k. \leq \underline{a}} \sup_x |D^k f(x)| + \max_{k. = \underline{a}} \sup_{x,y} \frac{|D^k f(x) - D^k f(y)|}{\|x - y\|^{a - \underline{a}}},$$

where the suprema taken over all x, y in the interior of χ_d with $x \neq y$. a measures how smooth the function class is. $\underline{\alpha}$ is the greatest integer smaller than a.

Let $C_M^{a,d}$ be the set of all continuous functions $f : \chi_d \to \mathbb{R}$ with $\|f\|_a \leq M$. To obtain the covering number of the function space, I quote the theorem 2.7.1 from [31]

Theorem 3.4.1. *Let χ_d be a bounded, convex subset of $\mathbb{R}^d$ with nonempty interior. There exists a constant K depending only on a and d such that*

$$\log \mathcal{N}(\epsilon, C_1^{a,d}(\chi_d), \|\cdot\|_\infty) \leq K \lambda(\chi_d^1) (\frac{1}{\epsilon})^{d/a},$$

for every $\epsilon > 0$, where $\lambda(\chi_d^1)$ is the Lebesgue measure of the set $\{x : \|x - \chi_d\| < 1\}$.

It is easy to show that the value of $\lambda(\chi_d^1)$ is 3 and $5 + \pi$ when χ_d is $[0, 1]$ and $[0, 1] \times [0, 1]$.

Since the two cases are the only cases that are considered in the proof, I have

$$\log \mathcal{N}(\epsilon, C_1^a(\chi_d), \|\cdot\|_\infty) \leq K(\frac{1}{\epsilon})^{d/a},$$

by replacing K with $9K$.

Since $C_M^a(\chi_d) = \{f | f/M \in C_1^a(\chi_d)\}$, we have

$$\mathcal{N}(M\epsilon, C_M^a(\chi_d), \|\cdot\|_\infty) = \mathcal{N}(\epsilon, C_1^a(\chi_d), \|\cdot\|_\infty).$$

Finally, the covering number of function space $\mathcal{N}(M\epsilon, C_M^a(\chi_d), \|\cdot\|_\infty)$ satisfies

$$\log \mathcal{N}(\epsilon, C_M^a(\chi_d), \|\cdot\|_\infty) \leq K(\frac{M}{\epsilon})^{d/a}.$$

To calculate the covering number of functions space within functional neural networks space $F_d(\Psi)$, I have to define the distance of the space in the following:

$$d(f, \tilde{f}) = \max\{\sup|\alpha_i - \tilde{\alpha}_i|, \sup|\beta_i - \tilde{\beta}_i|; i = 1, \cdots, d\},$$

where $f, \tilde{f} \in F_d(\Psi)$ and their corresponding coefficient functions are $\alpha_i, \beta_i)$ and $\tilde{\alpha}_i, \tilde{\beta}_i)$.

3.4.2 Deductions

Since the domain H of our model is a compact set, the input function $x_0(t_0)$ should have a common boundary, which is denoted by E. The term X_i is used to denote the set consisting of all possible $x_i(t_i)$. I use the term S_i to denote the set

$$\left\{\int_0^1 \alpha(t_i,t_{i-1})x_i(t_{i-1})\mathrm{d}t_{i-1}+\beta_i(t_i)|\alpha(t_i,t_{i-1})\in C_M^{a,2},\beta_i(t_i)\in C_M^{a/2,1}\right\}.$$

Lemma 3.4.1.

$$\log\mathcal{N}_\infty(M\epsilon+(E_i+1)\delta,S_i)\leq\log\mathcal{N}_\infty(\epsilon,X_{i-1})+2K(\frac{M}{\delta})^{2/a},$$

where the functions in X_{i-1} is bounded by E_i.

Proof. For each $x_{i-1}(t_i)\in X_{i-1},\alpha_i(t_i,t_{i-1})\in C_M^{a,2},\beta_i(t_i)\in C_M^{a/2,1}$, we can find their representatives $\tilde{x}_{i-1}(t_{i-1}),\tilde{\alpha}_i(t_i,t_{i-1}),\tilde{\beta}_i(t_i)$ from their corresponding ϵ-cover, δ-cover and δ-cover.

$$\begin{aligned}
&\int_0^1 \alpha_i(t_i,t_{i-1})x_{i-1}(t_{i-1})\mathrm{d}t_{i-1}+\beta_i(t_i)-\int_0^1\tilde{\alpha}_i(t_i,t_{i-1})\tilde{x}_{i-1}(t_{i-1})\mathrm{d}t_{i-1}-\tilde{\beta}_i(t_i)\\
=&\int_0^1(\alpha_i(t_i,t_{i-1})-\tilde{\alpha}_i(t_i,t_{i-1}))x_{i-1}(t_{i-1})\mathrm{d}t_{i-1}+\\
&\int_0^1\tilde{\alpha}_i(t_i,t_{i-1})(x_{i-1}(t_{i-1})-\tilde{x}_{i-1}(t_{i-1}))\mathrm{d}t_{i-1}+\beta_i(t_i)-\tilde{\beta}_i(t_i)\\
\leq&E_i\delta+M\epsilon+\delta.
\end{aligned}$$

Therefore, the Cartesian product of the two covering is the $(M\epsilon+(E_i+1)\delta)$-covering of S_i, i.e.,

$$\begin{aligned}
&\log\mathcal{N}_\infty(M\epsilon+(E_i+1)\delta,S_i)\\
\leq&\log\mathcal{N}_\infty(\epsilon,X_{i-1})+\log\mathcal{N}_\infty(\delta,C_M^{a,2})+\log\mathcal{N}_\infty(\delta,C_M^{a/2,1})\\
\leq&\log\mathcal{N}_\infty(\epsilon,X_{i-1})+2K(\frac{M}{\delta})^{2/a}.
\end{aligned}$$

□

$E_1 = E$ in the first layer. When the layer $i > 1$, the value is within the range of a squashing function, therefore $E_i = 1$.

In the first layer, since $X_0(t_0)$ only consists of identity function without any parameters, $\log \mathcal{N}_\infty(\epsilon, X_0) = 0$, $\log \mathcal{N}_\infty(\delta, S_1) \leq 2K(\frac{(E+1)M}{\delta})^{2/a}$.

To illustrate the effect of nonlinear activation, I quote lemma 14.13 in [1].

Lemma 3.4.2. *Suppose F is a class of real-valued functions in the vector space from domain X to $\mathbf{R}^p$ and $\sigma : \mathbf{R} \to \mathbf{R}$ is a element-wise function satisfying Lipschitz condition: $|\sigma(x) - \sigma(y)| \leq L|x - y|$. Let $\sigma(F)$ denote the class $\{\sigma \circ f : f \in F\}$, then*

$$\mathcal{N}_\infty(\epsilon, \sigma \circ F, m) \leq \mathcal{N}_\infty(\epsilon/L, F, m).$$

Suppose the squashing function Ψ is L-Lipschitz, I have

$$\log \mathcal{N}_\infty(\epsilon, X_i) \leq \log \mathcal{N}_\infty(\epsilon/L, S_i).$$

Based on the results of the two lemmas, if $i > 1$, therefore

$$\log \mathcal{N}_\infty(ML\epsilon + 2\delta, S_i) \leq \log \mathcal{N}_\infty(\epsilon, S_{i-1}) + 2K(\frac{M}{\delta})^{2/a}.$$

We have known that $\log \mathcal{N}_\infty(\delta, S_1) = 2K(\frac{(E+1)M}{\delta})^{2/a}$. Suppose that $\log \mathcal{N}_\infty(\delta, S_{i-1}) = c_{i-1}K(\frac{Mb_{i-1}}{\delta})^{2/a}$, then

$$\log \mathcal{N}_\infty(ML\epsilon + 2\delta, S_i) \leq c_{i-1}K(\frac{Mb_{i-1}}{\epsilon})^{2/a} + 2K(\frac{M}{\delta})^{2/a}.$$

Take $\epsilon = \delta b_{i-1}$,

$$\log \mathcal{N}_\infty((MLb_{i-1}+2)\delta, S_i) \leq (c_{i-1}+2)K(\frac{M}{\delta})^{2/a}.$$

Furthermore, by scaling the distance, we have

$$\log \mathcal{N}_\infty(\delta, S_i) \leq (c_{i-1}+2)K(\frac{M(MLb_{i-1}+2)}{\delta})^{2/a}.$$

i.e. $c_i = c_{i-1}+2$ and $b_i = MLb_{i-1}+2$. Considering that $c_1 = 2$ and $b_1 = E+1$, we can have that $c_d = 2d$ and $b_d = (E + \frac{ML+1}{ML-1})(ML)^{d-1} - \frac{2}{ML-1}$. Therefore it can be concluded that

$$\log \mathcal{N}_\infty(\delta, S_d) \leq 2dK(\frac{M(E + \frac{ML+1}{ML-1})(ML)^{d-1} - \frac{2}{ML-1}}{\delta})^{2/a}.$$

3.5 Consistency

When it comes to consistency, the error can be divided into the approximation error and the estimation error, which is expressed by Lemma 10.1 in the book on non-parametric regression [11].

Lemma 3.5.1. *Let $\mathcal{F}_n = \mathcal{F}_n(D_n)$ be a class of functions f: $\mathbb{R}^d \to \mathbb{R}$ depending on the data $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. If m_n satisfies*

$$m_n(\cdot) = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{j=1}^{n} |f(X_j) - Y_j|^2.$$

then

$$\int |m_n(x) - m(x)|^2 \mu(\mathrm{d}x) \tag{3.3}$$

$$\leq 2 \sup_{f \in \mathcal{F}_n} \left| \frac{1}{n} \sum_{j=1}^{n} |f(X_j) - Y_j|^2 - \mathbf{E}\{f(X) - Y)^2\} \right| \tag{3.4}$$

$$+ \inf_{f \in \mathcal{F}_n} \int |f(x) - m(x)|^2 \mu(\mathrm{d}x). \tag{3.5}$$

As is seen in the decomposition, the estimation error (3.4) refers to the the supremum difference between the error of the trained data and the error expectation over all function space, while the approximation error (3.5) refers to the infimum difference between the true function and the model function space. As model the function space becomes larger, the approximation error is decreasing while the approximation error is increasing. To make the total error as small as possible, I have to make a trade off in the selection of function space $\mathcal{F}_n$.

I quote theorem 10.2 in [11] to clarify what is required in the consistency problem.

Theorem 3.5.1. *Let $\mathcal{F}_n = \mathcal{F}_n(D_n)$ be a class of functions $f : \mathbb{R}^d \to \mathbb{R}$ and the estimator m_n satisfies*

$$\tilde{m}_n \in \mathcal{F}_n \quad \text{and} \quad \frac{1}{n} \sum_{j=1}^{n} |\tilde{m}_n(X_j) - Y_j|^2 = \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{j=1}^{n} |f(X_j) - Y_j|^2,$$

$$m_n(x) = T_{\beta_n} \tilde{m}_n(x).$$

(a) If

$$\lim_{n \to \infty} \beta_n = \infty,$$

$$\lim_{n\to\infty} \inf_{f\in\mathcal{F}_n, \|f\|_\infty \le \beta_n} \int |f(x) - m(x)|^2 \mu(\mathrm{d}x) = 0 \quad a.s.,$$

$$\lim_{n\to\infty} \inf_{f\in T_{\beta_n}\mathcal{F}_n} \left| \frac{1}{n}\sum_{j=1}^{n} |f(X_j) - Y_{j,L}|^2 - \mathbb{E}\{(f(X) - Y_L)^2\} \right| = 0,$$

a.s. for all $L > 0$, then

$$\lim_{n\to\infty} \int |m_n(x) - m(x)|^2 \mu(\mathrm{d}x) = 0 \quad a.s..$$

(b) If

$$\lim_{n\to\infty} \beta_n = \infty,$$

$$\lim_{n\to\infty} \mathbb{E}\left\{ \inf_{f\in\mathcal{F}_n, \|f\|_\infty \le \beta_n} \int |f(x) - m(x)|^2 \mu(\mathrm{d}x) \right\} = 0,$$

$$\lim_{n\to\infty} \mathbb{E}\left\{ \inf_{f\in T_{\beta_n}\mathcal{F}_n} \left| \frac{1}{n}\sum_{j=1}^{n} |f(X_j) - Y_{j,L}|^2 - \mathbb{E}\{(f(X) - Y_L)^2\} \right| \right\} = 0,$$

for all $L > 0$, then

$$\lim_{n\to\infty} \mathbb{E}\left\{ \int |m_n(x) - m(x)|^2 \mu(\mathrm{d}x) \right\} = 0.$$

I quote theorem 9.1 in [11] to bound the estimation error.

Theorem 3.5.2. *Let $\mathcal{G}$ be a set of functions $g : \mathbb{R}^d \to [0, B]$. For any n and $\epsilon > 0$,*

$$\mathbf{P}\left\{ \sup_{g\in\mathcal{G}} |\frac{1}{n}\sum_{i=1}^{n} g(Z_i) - \mathbf{E}\{g(Z)\}| > \epsilon \right\} \le 8\mathbf{E}\{\mathcal{N}_1(\epsilon/8, \mathcal{G}, Z_1^n)\} \exp(-n\epsilon^2/128B^2).$$

The domain of functions in above theorems are defines on $\mathbb{R}^d$ for neural networks model. I checked the proofs of them and make sure that their validity has no issues with the explicit form of its domain. Therefore I can apply these theorems in our FNN model whose domain

is defined in H.

The consistency of FNN model $F_d(\Psi)$ is given below.

Theorem 3.5.3. *Let $\mathcal{F}_n$ be a class of functions satisfying*

$$\mathcal{F}_n = \left\{ f \in F_d(\Psi) | \alpha_i(t_i, t_{i-1}) \in C^{a,2}_{M_n}, \beta_i(t_i) \in C^{a,1}_{M_n}, J_i \leq k_n \right\},$$

where Ψ is L-Lipschitz continuous squashing function. Let

$$m_n = \arg \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{j=1}^{n} |f(X_j) - Y_j|^2.$$

If M_n and k_n satisfy

$$\frac{M_n^{4+\frac{2d+2}{a}}}{n} \to 0, \beta_n \to \infty, M_n \to \infty,$$

then $\int_K (m_n(x) - m(x))^2 \mu(\mathrm{d}x) \to 0$ almost surely with $\mathbf{E}|Y|^2 < \infty$ and any compact set $K \subseteq H$.

Proof. For the approximation error, universal approximation theorem implies $\lim_{n\to\infty} M_n \to \infty$ and $\lim_{n\to\infty} k_n \to \infty$.

For the estimation error, theorem 3.5.1 implies that I can assume $|Y| \leq L$ almost surely, for some L, and then it is required to show that

$$\sup_{f \in \mathcal{F}_n} \left| \mathbf{E}|f(X) - Y|^2 - \frac{1}{n} \sum_{j=1}^{n} |f(X_j) - Y_j|^2 \right| \to 0 \quad a.s..$$

Let $H = L^2 \times [0, 1]$ and $f : H \to \mathbf{R}$. Define

$$Z = (X, Y), Z_1 = (X_1, Y_1), \ldots, Z_n = (X_n, Y_n),$$

where $X_i \in H$, $Y_i \in \mathbb{R}$ and

$$\mathcal{H}_n = \{h : H \times \mathbb{R} \to \mathbb{R} : \exists f \in \mathcal{F}_n \quad \text{s.t.} \quad h(x,y) = |f(x(t), t_d) - y|^2\}.$$

Assume $M_n \geq L$ so that functions in $\mathcal{H}_n$ satisfy

$$0 \leq h(x,y) \leq 2M_n^2 + 2L^2 \leq 4M_n^2.$$

Using the boundary of theorem 3.5.2, we have, for arbitrary $\epsilon > 0$,

$$\begin{aligned}
&\mathbf{P}\left\{\sup_{f\in\mathcal{F}_n} |\frac{1}{n}\sum_{i=1}^n |f(X_i) - Y_i|^2 - \mathbf{E}\{|f(X) - Y|^2\}| > \epsilon\right\} \\
=&\mathbf{P}\left\{\sup_{h\in\mathcal{H}_n} |\frac{1}{n}\sum_{i=1}^n h(Z_i) - \mathbf{E}\{h(Z)\}| > \epsilon\right\} \\
\leq& 8\mathbf{E}\mathcal{N}_1(\frac{\epsilon}{8}, \mathcal{H}_n, Z_1^n) e^{-\frac{n\epsilon^2}{128(4M_n^2)^2}}.
\end{aligned}$$

Next, I bound the covering number and get

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^n |h_1(Z_i) - h_2(Z_i)| \\
=&\frac{1}{n}\sum_{i=1}^n ||f_1(X_i) - Y_i|^2 - |f_2(X_i) - Y_i|^2| \\
=&\frac{1}{n}\sum_{i=1}^n |f_1(X_i) - f_2(X_i)||f_1(X_i) - Y_i + f_2(X_2) - Y_i| \\
\leq& 4M_n \frac{1}{n}\sum_{i=1}^n |f_1(X_i) - f_2(X_i)|.
\end{aligned}$$

Thus,

$$\mathcal{N}_1(\frac{\epsilon}{8}, \mathcal{H}_n, Z_1^n) \leq \mathcal{N}_1(\frac{\epsilon}{32M_n}, \mathcal{F}_n, X_1^n).$$

$$\begin{aligned}
&\mathbf{P}\left\{\sup_{f\in\mathcal{F}_n} |\frac{1}{n}\sum_{i=1}^{n} |f(X_i)-Y_i|^2 - \mathbf{E}\{|f(X)-Y|^2\}| > \epsilon\right\} \\
\leq& 8\mathcal{N}_1(\frac{\epsilon}{32M_n}, \mathcal{F}_n, X_1^n) e^{-\frac{n\epsilon^2}{128(4M_n^2)^2}} \\
\leq& 8\mathcal{N}_\infty(\frac{\epsilon}{32M_n}, \mathcal{F}_n, X_1^n) e^{-\frac{n\epsilon^2}{128(4M_n^2)^2}} \\
\leq& 8\exp\left(2dK(\frac{32M_n^2(E+\frac{M_nL+1}{M_nL-1})(M_nL)^{d-1}-\frac{2}{M_nL-1}}{\epsilon})^{2/a} - \frac{n\epsilon^2}{128(4M_n^2)^2}\right) \\
\leq& 8\exp\left(-n^\delta \frac{n^{1-\delta}}{M_n^4}(\frac{\epsilon^2}{2048} - \frac{2dKM_n^4}{n}(\frac{32M_n^2(E+\frac{M_nL+1}{M_nL-1})(M_nL)^{d-1}-\frac{2}{M_nL-1}}{\epsilon})^{2/a})\right),
\end{aligned}$$

if $\delta > 0$ exists. To satisfy the condition that

$$\sum_{n=1}^{\infty} \mathbf{P}\left\{\sup_{f\in\mathcal{F}_n} |\frac{1}{n}\sum_{i=1}^{n} |f(X_i)-Y_i|^2 - \mathbf{E}\{|f(X)-Y|^2\}| > \epsilon\right\} < \infty,$$

The following conditions are required:

$$\lim_{n\to\infty} \frac{n^{1-\delta}}{M_n^4} \to \infty,$$

and

$$\frac{2dKM_n^4}{n}(\frac{32M_n^2(E+\frac{M_nL+1}{M_nL-1})(M_nL)^{d-1}-\frac{2}{M_nL-1}}{\epsilon})^{2/a} \to 0.$$

The latter condition can be simplified as

$$\frac{M_n^{4+\frac{2d+2}{a}}}{n} \to 0$$

which implies the former one, therefore the former one is not necessary. □

In terms of consistency, approximation error and estimation error should be bounded together. The approximation error encourages the functional space to be larger, i.e. $k_n \to \infty$ and $M_n \to \infty$. The estimation error controls the increasing rate of M_n. The above theorem requires that the increasing rate of $M_n^{4+\frac{2d+2}{a}}$ is slower than n.

3.6 Discussion

In this chapter, the consistency of our proposed FNN model is proved, which implies that the error between the FNN model and the true model can be as small as possible if the sample size is large enough. In the future, I will investigate other statistical properties, such as the convergence rate and asymptotic normality of FNN.

When it comes to proof of the rate of convergence rate of FNN model, the idea is normally similar with that of consistency. I have to find the convergence rate of estimation error and approximation error and find a trade off. The convergence rate of estimation error is not hard to get given the covering number of the model, while that of approximation error is a mathematical problem, which is still under investigation.

Chapter 4

Transfer Learning for Genetic Studies

In the previous chapter,a new statistical model FNN is proposed and its consistency is proved. In this chapter, I will introduce transfer learning and study whether transfer learning can be used to improve the performance of genetic analyses.

4.1 Introduction

With the vast amounts of genetic data collected from biobank projects and from the Caucasian population, an interesting scientific question is whether these resources can be used to enhance the genetic analysis in small-scale studies or in minority populations (e.g., African American population). With the increased amount of genetic data becoming publicly available for genetic research, challenges remain in how to efficiently utilize these enriched resources in the studies of small-scale studies and in minority populations. A common assumption made by existing approaches is that two studies should be similar (e.g., the same population). However, this assumption could fail in reality. The study design and study population could be different between the two studies (e.g., Caucasian vs. African American). Transfer learning does not require data from two studies drawn from exactly the same feature space or the same distribution. It learns possibly useful feature from mature studies and applies learned features based on focused problem. Therefore it holds great promise to use the enriched resources from large-scale studies for uncovering novel genetic variants in

small-scale studies or in minority populations [30].

Transfer learning has become increasingly popular in AI and is expected to become a key driver of machine learning success in the future. Various models have been tested and therefore showed the method and strength of transfer learning ([18]). Transfer learning is a widely used method in image classification and natural language processing[13], which transfer the knowledge from a source study/population to a target study/population.

While transfer learning has achieved success in areas such as natural language processing, it has rarely been used in genetic data analysis. In this chapter, I integrate the idea of transfer learning into neural networks and use the knowledge learned from the large-scale UK Biobank dataset to facilitate genetic analysis in small-scale studies or in minority populations. By integrating transfer learning into neural networks, I am able to transfer knowledge regarding complex genotype-phenotype relationships (e.g., gene-gene interactions) between two studies.

The remaining chapter is organized as follows: The details of our models and two candidate regularization methods are introduced in section 2. Real data analyses are given in section 3, which provides the comparison and selection between the two regularization methods and the performance of transfer learning. Summary and discussion are provided in section 4.

4.2 Method

Transfer learning applies knowledge gained from one research problem to a different but related problem, and has been widely used in text and image recognition. The basic idea of transfer learning is illustrated in Figure 1. In this chapter, the model I used to illustrate transfer learning is the vanilla neural networks (NN) model. In this section, I briefly introduce

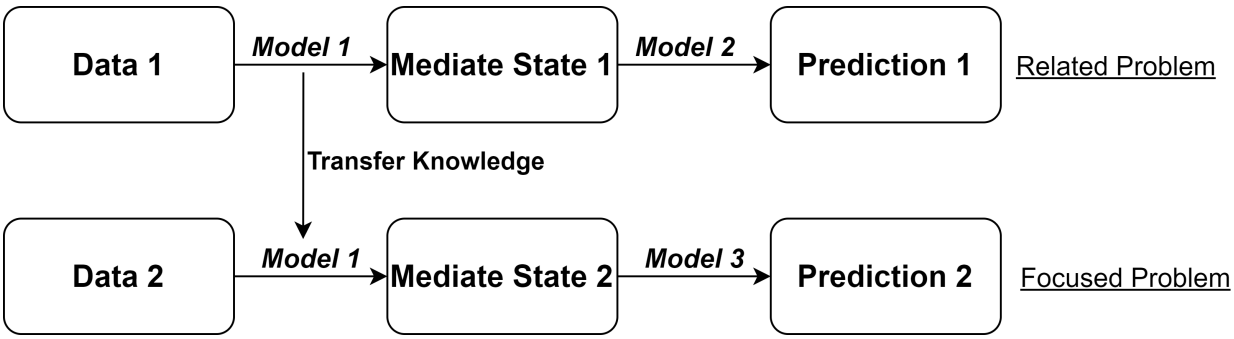

Figure 4.1: Transfer Learning

the NN model, and then integrate the idea transfer learning into NN (TL-NN) to transfer the knowledge between two datasets. Finally, the technical details of the TL-NN will be provided.

To distinguish the different types of data types in Chapter 4, I use lower case letters, bold lower case letters and upper case letters to denote scalar, vector and matrix, respectively. I also use lower case letters denote the output function.

In this section, the vanilla neural networks is introduced at first. The transfer learning model is then constructed based on neural networks model. The details of technical issues are discussed in the end.

4.2.1 Neural Networks

Suppose our research interest is to find a predictive function f that models the response variable y and predictor variable $\boldsymbol{x} = (x_1, \ldots, x_q)$, where q is the dimension of input. The true model is written as:

$$y = f(\boldsymbol{x}) + \epsilon, \tag{4.1}$$

where ϵ is the noise in this model.

I use the fully connected NN model to demonstrate transfer learning:

$$\boldsymbol{h}_1 = \sigma(W_1\boldsymbol{x} + \boldsymbol{b}_1), \tag{4.2}$$

$$\boldsymbol{h}_d = \sigma(W_d\boldsymbol{h}_{d-1} + \boldsymbol{b}_d), \quad d = 2, \ldots, l-1, \tag{4.3}$$

$$\hat{y} = \hat{f}(\boldsymbol{x}) = \boldsymbol{w}_l\boldsymbol{h}_{l-1}, \tag{4.4}$$

where $\boldsymbol{h}_d$ is the feature which is learned from the source study and has a dimension of m_d. σ is a nonlinear activation function, such as the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ used this chapter. l is the number of layers of the NN model. W_d and $\boldsymbol{b}_d$ are weights and biases, respectively. I denote all the parameters by $\boldsymbol{S} = \{W_d, \boldsymbol{b}_d, \boldsymbol{w}_l | d = 1, \ldots, l-1\}$.

In NN, I can normalize the response variable y. Suppose that $\bar{y}$ and $\bar{v}$ are the estimated mean value and standard deviation of y, the final model is realized by:

$$\hat{y} = \hat{f}(\boldsymbol{x}) = \bar{v}\boldsymbol{w}_D\boldsymbol{h}_{D-1} + \bar{y}. \tag{4.5}$$

NN can be applied to both regression and classification problems. For simplicity, I here focus on the regression problem, and use L_2 loss and mean square error (MSE) to estimate parameters and evaluate the model's performance. The model can also be applied to classification problem by using different loss functions (e.g., cross entropy) and measurements (e.g., misclassification error)

Based on the L_2 loss, the parameters can be estimated,

$$\widehat{\boldsymbol{S}} = \arg\min_{\boldsymbol{S}} \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\boldsymbol{x}_i))^2,$$

where n is the sample size. $\widehat{\boldsymbol{S}}$ is the set of all estimated parameters. y_i and $\boldsymbol{x}_i$ are the

response and predictor variable of the i-th sample.

Directly fitting NN to a large number of genetic covariates could bring an overfitting issue. To avoid the overfitting issue, various regularization methods, such as penalty regularization, drop out method and early stopping method can be used. In this chapter, I impose parameter norm penalty $p = p(\boldsymbol{S}; \lambda)$ over our function space as a regularization method. where λ is a hyperparameter controling the solution space.

In a typical NN, a L_2 penalty is usually applied on the weight term,

$$p(\boldsymbol{S}; \lambda) = \lambda(\sum_{d=1}^{l-1} \|W^d\|_2^2 + \|\boldsymbol{w}_l\|_2^2).$$

.

In genetic studies, genetic effects are usually smaller than the noise, and therefore heavy penalties are required to avoid overfitting.

There are two regularization methods regarding the parameter norm penalty, parameter regularization and constraint regularization. Their solutions are:

$$\widehat{\boldsymbol{S}} = \arg\min_{\boldsymbol{S}} \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 + p(\boldsymbol{S}; \lambda) \right), \tag{4.6}$$

$$\widehat{\boldsymbol{S}} = \arg\min_{\boldsymbol{S}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 \quad \text{s.t.} \quad p(\boldsymbol{S}; \lambda) \leq 1. \tag{4.7}$$

The backpropagation method is typically used to obtain the solution. Since $p(\boldsymbol{S}; \lambda)$ is a norm penalty, the larger λ is, the smaller the parameter space is. Parameter regularization is a widely used regularization method to avoid overfitting. It adds a penalty term to the objective function, which encourages the weight of NN to be small in each epoch of

backpropagation. Nonetheless, large weights are still possible for parameter regularization. Constraint regularization can be used to avoid large weights by forcing the weights to be small, but the penalty term would not affect the backpropagation.

In the section 4.3.2, I compare the performance between parameter regularization and constraint regularization. Based on the result, I recommend constraint regularization for genetic studies. The selection of hyperparameter λ is also discussed in the next section.

4.2.2 Transfer Learning

If we have a source data (e.g., UK biobank), in which we model the response variable y' with predictor variable $\boldsymbol{x'} = (x'_1, \ldots, x'_q)$, where $\boldsymbol{x}$ and $\boldsymbol{x'}$ shares similar data structure and effects. Usually, $\boldsymbol{x}$ and $\boldsymbol{x'}$ are the same properties of different subjects. We have the similar model:

$$\boldsymbol{h}'_1 = \sigma(W'_1\boldsymbol{x'} + \boldsymbol{b}'_1), \tag{4.8}$$

$$\boldsymbol{h}'_d = \sigma(W'_d\boldsymbol{h}'_{d-1} + \boldsymbol{b}'_d), \quad d = 2, \ldots, l-1, \tag{4.9}$$

$$\hat{y}' = \hat{f}(\boldsymbol{x'}) = \bar{v'}\boldsymbol{w}'_l\boldsymbol{h}'_{l-1} + \bar{y'}, \tag{4.10}$$

where $\bar{y'}$ and $\bar{v'}$ are the estimated mean value and standard deviation of y'. The parameters space of this model is denoted by $\boldsymbol{S'}$. The set of their solutions are denoted by $\widehat{\boldsymbol{S'}}$.

Since the solutions are derived from source studies, it is reasonable to suppose that $\boldsymbol{h}'_d$ is a good representation of features in $\boldsymbol{x'}$. The relation between $\boldsymbol{x'}$ and $\boldsymbol{h}'_d$ are realized through $\{\hat{W}'_d, \hat{\boldsymbol{b}}'_d | d = 1, \ldots, l-1\}$, which is denoted by $\widehat{\boldsymbol{S}'_1}$. The other part $\{\hat{\boldsymbol{w}}'_l\}$ is represented by $\widehat{\boldsymbol{S}'_2}$. Furthermore, we use f_1 and f_2 to denote the relationship, i.e.,

$$\boldsymbol{h}'_d = f'_1(\boldsymbol{x}|\widehat{\boldsymbol{S}'_1}), \tag{4.11}$$

$$\hat{y}' = f'_2(\boldsymbol{h}'_d|\widehat{\boldsymbol{S}'_2}). \tag{4.12}$$

Similarly, the parameter set $\widehat{\boldsymbol{S}}$ from the focused problem can also be divided to $\widehat{\boldsymbol{S}_1}$ and $\widehat{\boldsymbol{S}_2}$. The corresponding relationship is denoted by f_1 and f_2. The penalty term $p(\boldsymbol{S};\lambda)$ can also be written as $p(\boldsymbol{S}_1, \boldsymbol{S}_2; \lambda)$.

Based on the idea of transfer learning, I use $\widehat{\boldsymbol{S}'_1}$ as the solution of the focused problem rather than $\widehat{\boldsymbol{S}_1}$. When it comes to the solution of $\boldsymbol{S}_2$, it is derived from the dataset of focused problem given $\boldsymbol{S}_1 = \widehat{\boldsymbol{S}'_1}$

Our final solution in transfer learning is:

$$\widetilde{\boldsymbol{S}_1} = \widehat{\boldsymbol{S}_1} = \arg\min_{\boldsymbol{S}_1, \boldsymbol{S}_2} \frac{1}{n}\sum_{i=1}^{n}(y'_i - f'(\boldsymbol{x}'_i))^2, \quad \text{s.t.} \quad p(\boldsymbol{S}_1, \boldsymbol{S}_2; \lambda) \leq 1, \tag{4.13}$$

$$\widetilde{\boldsymbol{S}_2} = \arg\min_{\boldsymbol{S}_2} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i|\boldsymbol{S}_1 = \widetilde{\boldsymbol{S}_1}))^2, \quad \text{s.t.} \quad p(\widetilde{\boldsymbol{S}_1}, \boldsymbol{S}_2; \lambda) \leq 1. \tag{4.14}$$

The difference between neural networks and transfer learning methods is illustrated in Figure 2.

4.2.3 Technical Issue

The optimization algorithm in this chapter is the Adam algorithm [16]. To satisfy the restriction condition $p(\boldsymbol{S}_1, \boldsymbol{S}_2; \lambda) \leq 1$, the back propagation procedure is realized as:

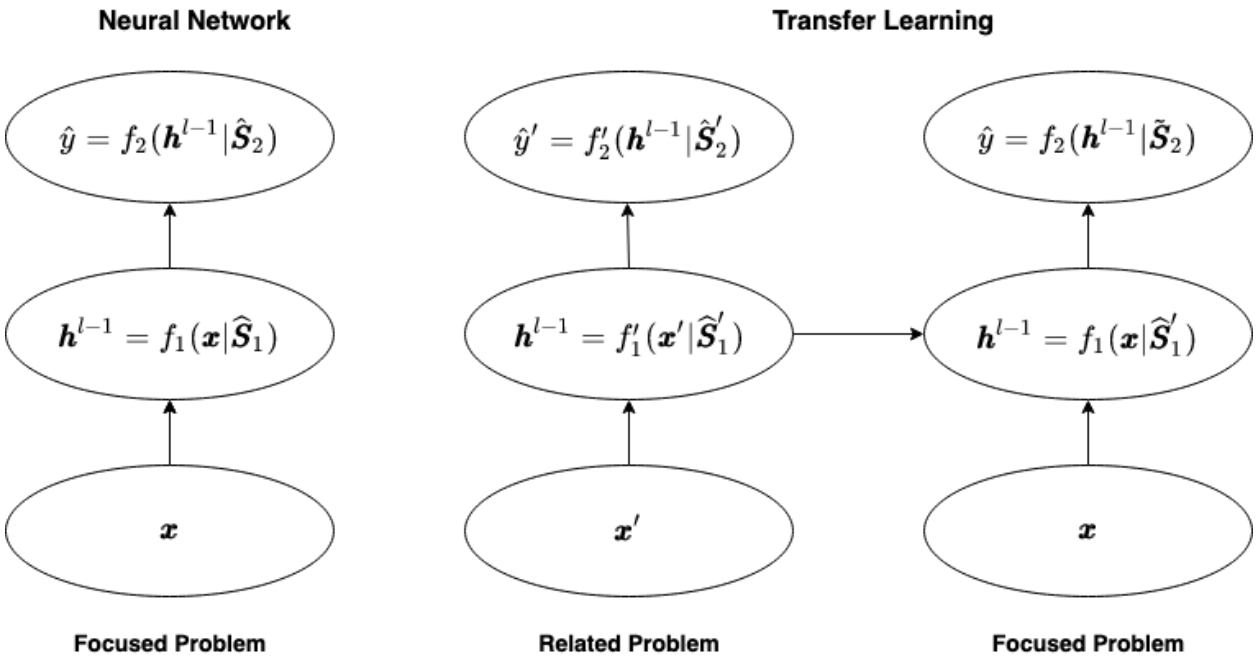

Figure 4.2: Transfer Learning in neural networks

$$c(\boldsymbol{S}) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2, & \text{if } p(\boldsymbol{S}_1, \boldsymbol{S}_2; \lambda) \leq 1 \\ p(\boldsymbol{S}_1, \boldsymbol{S}_2; \lambda_1, \lambda_2), & \text{if } p(\boldsymbol{S}_1, \boldsymbol{S}_2; \lambda) > 1 \end{cases}, \tag{4.15}$$

$$s \leftarrow s - r_1 \frac{\partial c(\boldsymbol{S})}{\partial s}, \quad s \in \boldsymbol{S}_1, \tag{4.16}$$

$$s \leftarrow s - r_2 \frac{\partial c(\boldsymbol{S})}{\partial s}, \quad s \in \boldsymbol{S}_2, \tag{4.17}$$

where r_1, r_2 is the learning rate of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively.

The Adam algorithm is an adaptive optimization method where the learning rate is determined element wisely. While the default setting of the Adam algorithm works well in most cases, it is not suitable for genetic studies. Due to the heavy penalty in the last layer, the solution of $\boldsymbol{S}$ can be too small compared with the default learning rate of the Adam algorithm. Therefore, I set the parameter of the learning rate based on λ while keeping

other parameters as the default value. The selection of the parameter is described in detail in the next section. For transfer learning, I keep the same settings as the one used in NN, i.e., $r_1 = 0$ while r_2 .

The iteration process stops when MSE does not decrease for 3000 epochs. The formed NN model has 2 hidden layers (i.e., $l = 3$). The numbers of hidden units of each layer is 16 and 4.

For the comparison purpose, I also added a baseline model, which is essentially the mean value of predictor variable $\bar{y}$. The measurement comparing two models is therefore:

$$m(f) = 1 - \frac{\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

The larger the value is, the better the formed model as compared to the baseline model.

To illustrate the prediction ability, I split investigated dataset into the training set (80%) and test set (20%). I tuned the parameters of our models on the training set, and compare their performance on the test set. In order to remove the random effect of splitting, I repeat our experiment for 100 times by random splitting the sample. The results are summarized by the boxplots, where the left part shows the value of $m(f)$ in the training set and the right show that of the test set.

4.3 Real Data Analysis

4.3.1 Data Description

Cigarette smoking is one of the leading causes of preventable disease, contributing to 5 million deaths worldwide each year[21]. During the last decade, a great deal of progress has

been made in identifying genetic variants associated with smoking. Among those findings, the nAChRs subunit genes (e.g., *CHRNA5*) have been identified and confirmed in several large-scale studies[19]. In this application, I use transfer learning to study the complex relationships nAChRs subunit genes with nicotine dependence.

The datasets to be analyzed are the large-scale UK Biobank (UKB) dataset and the relatively small-scale dataset from the Study of Addiction: Genetics and Environment (SAGE). [2] UKB is a population-based prospective cohort of nearly $500,000$ individuals recruited in the United Kingdom who were aged 40-69 years. UKB contains a wealth of data on detailed clinical information, genome-wide genotype data and whole-exome sequencing data. SAGE is one of the most comprehensive studies conducted to date, aimed at discovering genetic contributions to substance use disorders. Samples from SAGE were selected from three studies: COGEND, COGA, and FSCD. Multiple phenotypes were measured in the SAGE studies. For the focus of our gene-based analysis, I used cigarettes per day (CPD) that is available in both SAGE and UKB, and only considered genes from two clusters, CHRNA5-CHRNA3-CHRNB4 and CHRNB3-CHRNA6. SAGE comprises 1445 female and 1272 male samples, among which 807 samples are African-American and 1910 samples are Caucasian.

Prior to the analysis, I re-assessed the quality of the data (e.g., check for successful genotype calls, missing rate, deviation from the Hardy-Weinberg equilibrium, unexpected relationships67). After a careful quality assessment, I first use the Irish sample to find the hyperparameters.

Two studies of transfer learning are investigated in this chapter. One is a cross-project study and the other is a cross-ethnicity study. In the first study, I am trying to investigate whether the genetic finding from the British population can be transferred to the Black and the Irish populations. In the second study, I try to utilize the data from UKBiobank to help

the genetic research in SAGE.

To illustrate the effect of transfer learning strategy, I split investigated dataset into the training set (80%) and the test set (20%). I model our original NN model and transfer learning model respectively on the train set, and compare their performance on the test set. In order to remove the random effect of splitting, I repeat our experiment for 100 times with random splitting.

A validation process is used to determine the value of λ. In the validation process, I split our training set into the subtrain and validation sets with the ratio of $4:1$. I evaluate a range of possible values of λ, such as $\{1, 10, 100, 1000\}$, in the subtrain set, and then select the optimal λ based on the validation set.

4.3.2 Regularization

In this section, two regularization methods, parameter regularization and constraint regularization methods are compared under different possible values of hyperparameter λ. The Constraint regularization method and suitable hyperparameter selection are then used in the next section.

4.3.2.1 Constraint regularization

Various values of hyperparameters for the constraint regularization is illustrated here. The candidates of λ are $\{10^2, 10^{1.5}.10^1, 10^{0.5}, 10^0\}$.

As is seen from Figure 3, the optimal parameter for the penalty term λ is between $10^{0.5}$ and 10^0.

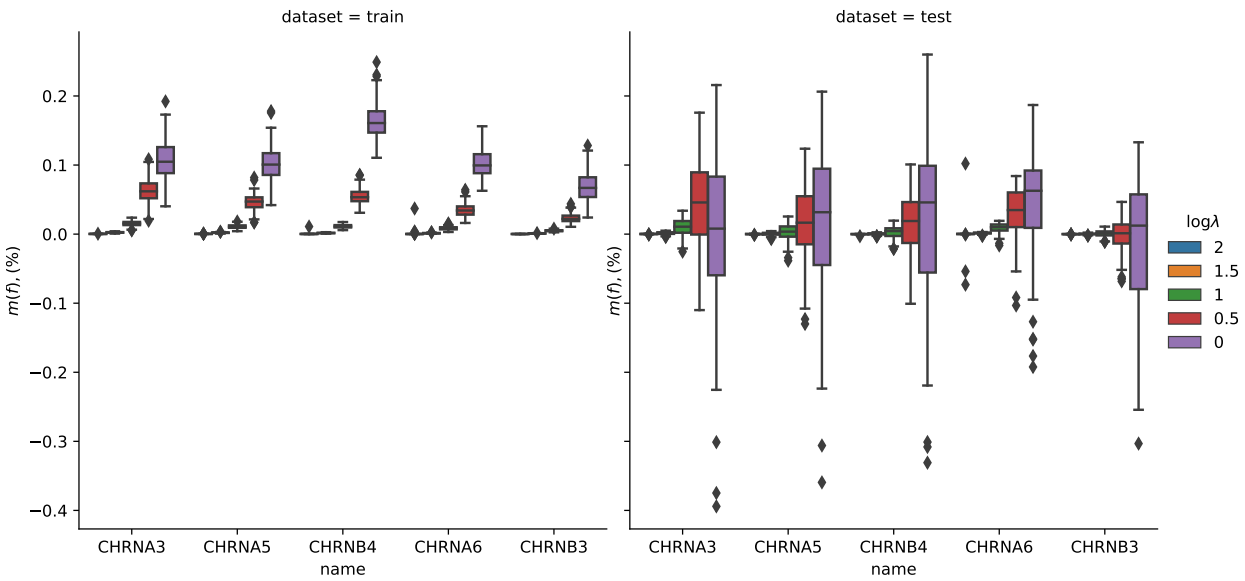

Figure 4.3: Constraint Regularization

4.3.2.2 Parameter Regularization

Possible values of λ for the constraint regularization are $\{10^0, 10^{-0.5}.10^{-1}, 10^{-1.5}, 10^{-2}\}$. The corresponding parameter of learning rate in the Adam method is $10^{-2.5} \cdot \lambda^{-0.5}$. The optimal parameter for the penalty term λ is between $10^{-1.5}$ and 10^{-2} according to Figure 4.

4.3.2.3 Comparison

Compared with parameter regularization, constraint regularization encourages a higher learning rate, which enables NN to jump out of local minimum and attain better performance. The result is more stable with various λ values in the penalty constraint. Therefore, in the following analysis, I will use the constraint regularization method with the selected parameter setting.

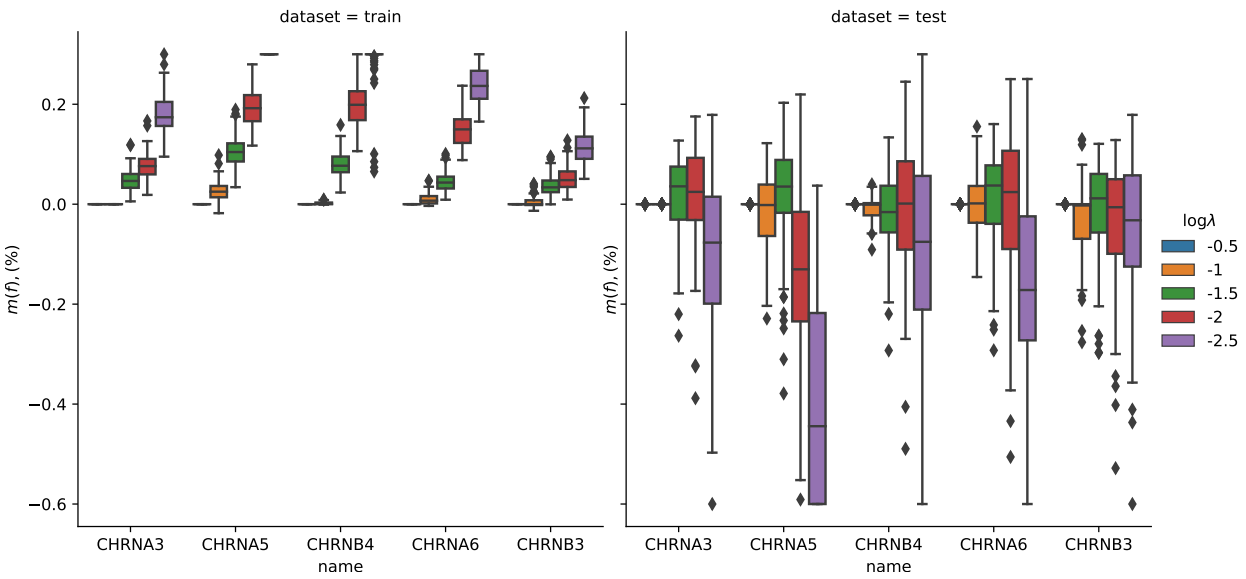

Figure 4.4: Parameter Regularization

4.3.3 Cross-ethnicity

The vast amount of genetic data collected from the Caucasian population provides us with great resource for genetic research in other populations, especially minority populations. In the UK Biobank dataset, there are around $300,000$ British people, $10,000$ Irish people and $10,000$ Black people. In this project, I transfer the knowledge learned from the British population to the Irish and the Black to enhance the genetic study of these populations.

By applying the NN model and transfer learning model, I obtain the following result in Figure 5 for the Black population.

The performance comparison in the Irish population is given in Figure 6.

As is seen in the figure, transfer learning helps improve the performance for *CHRNA3*, *CHRNA5* and *CHRNB4*, while has limited gain for *CHRNA6* and *CHRNB3*.

[38] found genetic heterogeneity of *CHRNA6* and *CHRNB3* in the studies of nicotine dependence, while the heterogeneity is not significant in the other three genes, which is consistent with our finding.

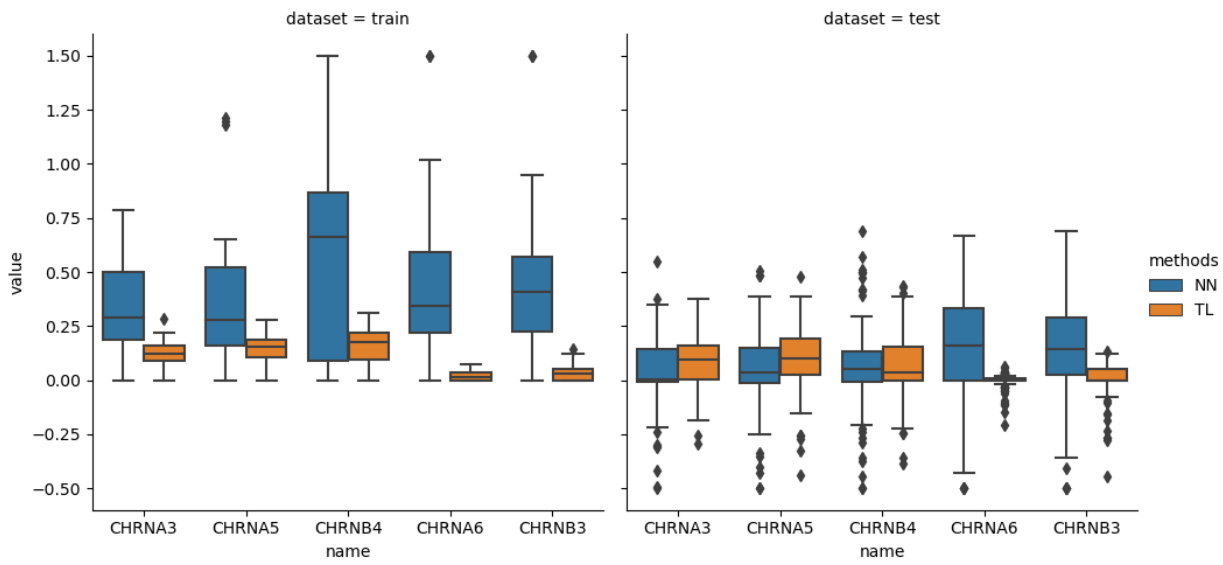

Figure 4.5: Transfer the knowledge from the British population to the Black population

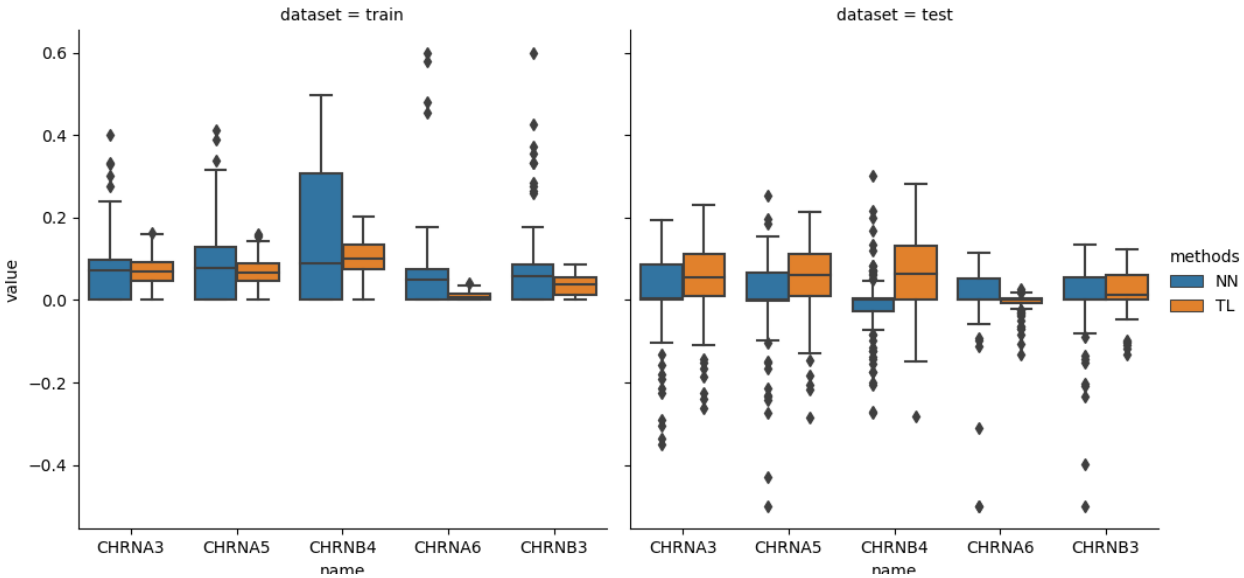

Figure 4.6: Transfer the knowledge from the British population to the Irish population

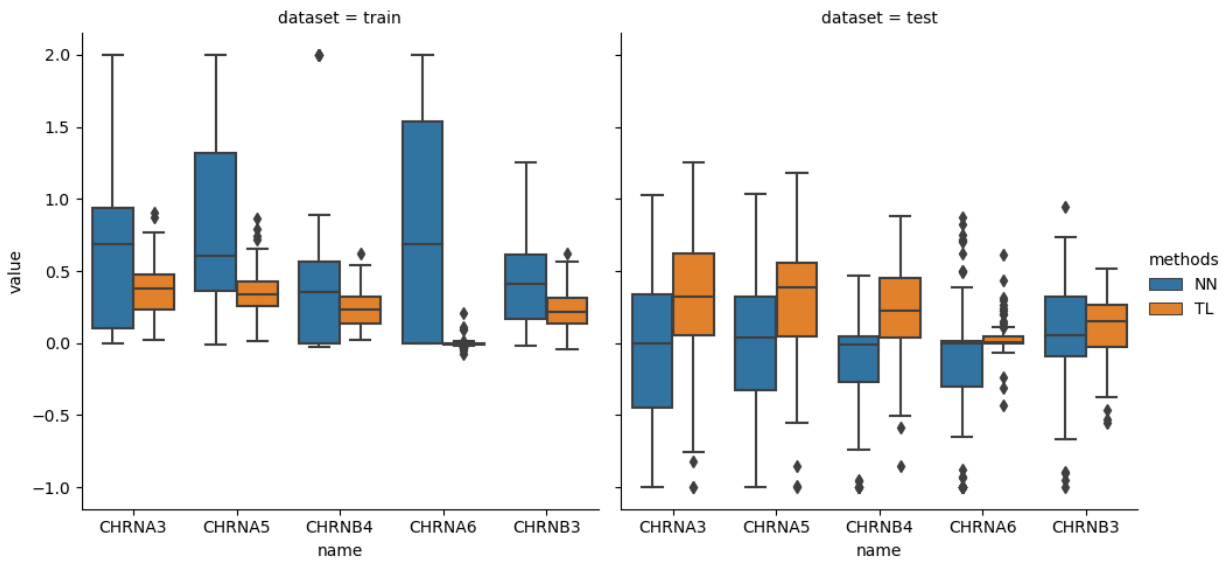

Figure 4.7: Transfer the knowledge from UK Biobank to SAGE

4.3.4 Cross-Projects

In this scenario, I am transferring the knowledge from the UK Biobank data to the SAGE data. For this analysis, I focus on the Caucasian population in both datasets.

By applying the NN model and transfer learning model to the datasets, I have the result which is illustrated in Figure 7.

As we can see from the figure, the NN model performs poorly due to the small sample size (i.e., around 1000) in the SAGE dataset, and the genetic effect can only explain less than 1 percent of the variation. Compared to the NN, transfer learning improves the performance by using the information from the UK biobank. The improvement is observed for all genes as we focus on the Caucasian population in this application and genetic heterogeneity has little influence on the performance.

4.4 Discussion

In this chapter, I have shown the advantages of transfer learning for genetic research. Transfer learning can improve the prediction performance across populations, but we need to be cautious about the heterogeneity of data or genetic mechanisms, such as genetic heterogeneity. While I illustrate the method by transferring the knowledge to a small-scale study and minority population, transfer learning can be potentially used for other purposes, such as transferring the genetic findings between different traits (e.g., smoking and alcohol drinking) and different species (e.g., rats and mice).

Chapter 5

Epilogue

The main goal of this dissertation is to investigate neural networks based models with their application to genetics. In Chapter 2, I proposed a new neural networks model, FNN, which combines functional linear model and nonlinear activation in a recursive way. By treating genetic and phenotype data as functional data, The underlying smooth property is utilized to improve the models' performance in simulations and real data analyses. The consistency of the proposed FNN model is proved in Chapter 3 by providing the conditions to bound the approximation error and estimations error. In Chapter 4, I show how transfer learning can help transfer knowledge between different populations and studies.

In the FNN model discussed in Chapter 2, I mainly focused on the continuous phenotype. In the future, I will extend the FNN model to discrete variables and henceforth solve the classification problem. This can be done by changing the loss function and the corresponding link function. Another extension of the FNN model is the application to multivariate functions. In the real data analyses, the input function is SNPs, whose indexes are their positions on the chromosome. The output function is phenotypes over age. These functions have only one index. In the simulations, I have also shown that FNN outperformed existing methods when the output function is a bivariate function. Therefore, it will also be interesting to apply FNN to the brain surface data.

The consistency of the proposed FNN model is proved in Chapter 3. While consistency

has been proved for the vanilla NN model, the result is not directly applicable to the FNN model. The domain of variables in the NN model is defined on the Euclidean space, whose dimension is finite. Nevertheless, the domain of the FNN model contains the integrable function $L^2([0,1])$, whose dimension is infinite. Therefore, I have to prove the universal approximation theorem and calculate its covering number from scratch.

Chapter 3 proves that the estimated FNN model can approximate to the underlying model to any extend if enough samples are provided, but it cannot quantify the speed of the approximation, i.e., the rate of convergence. The difficulty lies in a mathematical problem, i.e., the distance of the FNN model function space and continuous function space defined on the domain of the FNN model. The is an interesting problem for future research.

Transfer learning is applied to genetic studies under the framework of vanilla NN model in Chapter 4. An obvious extension is to change the framework to our proposed FNN model. It can be realized by replacing NN with FNN. The real data analyses in this dissertation are limited to human genetic data, where I transfer knowledge from the British population to the Irish and Black populations. The formed model can also be used for cross-species studies, where the understanding of the genetic effects on human subjects can benefit from the knowledge learned from animals. Animal studies can be more powerful and designed in a way that is not allowed in human studies. I am cooperating with other groups on transfer learning between rats and mice. One of the challenges of cross-species studies is SNPs from different species are not aligned. Efforts are required to find the positions of the corresponding SNPs in other spices. Currently, I am trying to overcome the problems and using FNN in cross-species studies.

APPENDICES

APPENDIX A

Solutions of FLM

At first, the solution to equation (2.2) whose phenotype is a scalar value is discussed. In practice, I do not have the true functions but have the observations at discrete points $t_j, j = 1, \ldots, p$. Using the genetic data as an example, we observe genotypes G_{ij} for i-th samples at locus j. By considering the effects of genotypes as a function, I obtain the following beta-smooth function([7]):

$$\hat{y}_i = \theta_0 + Z_i\boldsymbol{\theta} + \sum_{j=1}^{p} G_{ij}\beta(t_j).$$

By using pre-specified basis functions $\beta_k(t), k = 1, \ldots, K$ (e.g., B-spline basis functions), I can further expand $\beta(t)$ as $\beta(t) = \sum_{k=1}^{K} w_k\beta_k(t)$ and rewrite FLM as,

$$\begin{aligned}
\hat{y}_i &= \theta_0 + Z_i\theta + \sum_{j=1}^{p} G_{ij}(\sum_{k=1}^{k} w_k\beta_k(t_j)) \\
&= \theta_0 + Z_i\theta + \sum_{k=1}^{k} w_k(\sum_{j=1}^{p} G_{ij}\beta_k(t_j)) \\
&= \theta_0 + Z_i\theta + \sum_{k=1}^{k} w_k d_{ik} \\
&= \theta_0 + Z_i\theta + D_i\boldsymbol{w},
\end{aligned}$$

where $d_{ik} = \sum_{j=1}^{p} G_{ij}\beta_k(t_j)$, $\boldsymbol{w} = (w_1, \ldots, w_K)^T$ and $D_i = (d_{i1}, \ldots, d_{iK})^T$. FLM (equation

(2.2)) is then transformed to a linear model:

$$\hat{Y} = X\mathbf{\Theta} + D\boldsymbol{w},$$

where $X = (1_n, Z)$, $Z = (Z_1^T, \ldots, Z_n^T)^T$, $\mathbf{\Theta} = (\theta_0, \boldsymbol{\theta}^T)^T$. 1_n refers to a column vector with all 1s.

The penalized loss function to solve the model is

$$\begin{aligned} R(\boldsymbol{w}, \mathbf{\Theta}) &= (\hat{Y} - Y)^T(\hat{Y} - Y) + \lambda \int (\beta''(t))^2 \mathrm{d}t \\ &= (\hat{Y} - Y)^T(\hat{Y} - Y) + \lambda \boldsymbol{w}^T P \boldsymbol{w}. \end{aligned}$$

By minimizing the loss function, I have the solution:

$$\begin{aligned} \hat{\boldsymbol{w}} &= (D^T(I - X(X^TX)^{-1}X^T)D + \lambda P)^{-1} D^T(I - X(X^TX)^{-1})Y \\ \hat{\mathbf{\Theta}} &= (X^TX)^{-1}X^T(Y - D\hat{\boldsymbol{w}}). \end{aligned}$$

Now I would provide the solution to (2.3) whose phenotype is dispersed in a one dimensional space. To our best knowledge, no existing analytical method is available to solve the model described in equation (2.3) without imposing restrictions on the model. I adapt the commonly used restriction ([14]) $s_{ij} = s_{i'j}, \forall i, i'$. Without loss of generality, I use s_j, ε_{ij} to denote s_{ij} and $\varepsilon_i(s_{ij})$. The model can then be rewritten as,

$$Y_{ij} = \mathbf{Y}_i(s_j) = Z_i\boldsymbol{\theta} + \alpha_0(s_j) + \int \alpha(s_j, t)\mathbf{G}_i(t)\mathbf{d}t + \varepsilon_{ij}.$$

I further generalize the above model by assuming that the coefficients for covariates $\boldsymbol{\theta}$ is

a function,

$$Y_{ij} = Z_i\boldsymbol{\theta}(s_j) + \alpha_0(s_j) + \int \alpha(s_j,t)\mathbf{G}_i(t)\mathbf{d}t + \varepsilon_{ij},$$

where $\boldsymbol{\theta}(s) = (\theta_1(s), \ldots, \theta_m(s))$.

By adding another basis functions $\alpha_l(s), l = 1, \ldots, L$, I replace $\alpha_0(s)$ and $\alpha(s,t)$ with $\sum_{l=1}^{L} c_l\alpha_l(s)$ and $\sum_{k=1}^{K}\sum_{l=1}^{L} W_{kl}\alpha_l(s)\beta_k(t)$, respectively.

For the bivariate function $\alpha(s,t)$, I use two basis function systems $\{\beta_{k(d)}^{(d)}, d \in \{0,1\}, k_d \in \{1, \ldots, l_d\}\}$,

$$\alpha_0(s) = \sum_{k^{(1)}=1}^{l^{(1)}} \boldsymbol{b}_{k^{(1)}}\beta_{k^{(1)}}^{(1)}(s),$$
$$\alpha(s,t) = \sum_{k^{(1)}=1}^{l^{(1)}}\sum_{k^{(0)}=1}^{l^{(0)}} W_{k^{(0)}k^{(1)}}\beta_{k^{(0)}}^{(0)}(t)\beta_{k^{(1)}}^{(1)}(s).$$

By using the package provided by [14], which is based on the REML method, I can obtain the solution of W, $\boldsymbol{b}$, and θ.

APPENDIX B

Functional Neural Networks

The technical details of the functional neural networks model will be discussed in this appendix. At first, the explicit form of the proposed model will be given step by step.

In practice, integration can be approximated by numeric integration in the form of summation. For simplicity, I assume that all functions is scaled in the interval of $[0, 1]$. If I choose evenly distributed points $t_1, t_2, \ldots, t_m$, a naive integration for $\mathbf{X}(t)$ is

$$\int \mathbf{X}(t)\mathrm{d}t = \sum_{j=1}^{m} \mathbf{X}(t_j)/m.$$

I can rewrite functional notation in a matrix form, i.e.

$$B^{(d)} = [\beta_{k(d)}^{(d)}(t_j^{(d)})]_{j=1,k(d)=1}^{m(d),l(d)},$$
$$X^{(d)} = [X_i^{(d)}(t_j^{(d)})]_{i=1,j=1}^{n,m(d)}.$$

Therefore, the forward propagation algorithm can be written as the following:

$$\begin{aligned} D^{(d)} &= X^{(d-1)}B^{(d-1)}/m^{(d-1)}, \\ C^{(1)} &= D^{(1)}W^{(1)} + 1_{n,1}\boldsymbol{b}^{(1)} + Z\theta, \\ C^{(d)} &= D^{(d)}W^{(d)} + 1_{n,1}\boldsymbol{b}^{(d)}, d > 1, \\ X^{(d)} &= \sigma(C^{(d)}B^{(d)T}). \end{aligned} \tag{B.1}$$

As is seen from the above equations, FNN reduces to a traditional neural networks when each matrix of basis B is a diagonal matrix. In other words, NN can be seen as a special case of FNN.

When it comes to the solution of the FNN model, the process of back propagation should be explained. A gradient decent method is used to solve the weight functions. The recursive process stops when $\mathbf{O}^{(r)}$ converges. The weights and biases are updated as follows:

$$\begin{aligned} \mathbf{O}^{(r)} &= \mathbf{O}(\alpha^{(d)}, \alpha_0^{(d)}), \\ W^{(r+1)} &= W^{(r)} - \gamma\frac{\partial \mathbf{O}^{(r)}}{\partial W^{(r)}}, \\ \boldsymbol{b}^{(r+1)} &= \boldsymbol{b}^{(r)} - \gamma\frac{\partial \mathbf{O}^{(r)}}{\partial \boldsymbol{b}^{(r)}}, \end{aligned}$$

where γ is determined by using the adadelta method [37].

The deductions of derivatives involve matrices, functions, and trace. Two lemmas are given below to simplify the calculation of derivatives.

Lemma B.0.1. *For any function $\sigma : \mathbb{R} \to \mathbb{R}$ applied to matrix elementwisely, $\mathbf{F} : \mathbb{R}^{m\times n} \to \mathbb{R}$ that maps matrix to scalar. X, A, B are matrices with proper dimensions and $A = \sigma(B)$. If $\frac{\partial \mathbf{F}(X)}{\partial A} = D$, then $\frac{\partial \mathbf{F}(X)}{\partial B} = D \circ \sigma'(B)$.*

Proof.

$$\frac{\mathbf{F}(X)}{B_{ij}} = \frac{\mathbf{F}(X)}{\sigma(A_{ij})}\frac{A_{ij}}{B_{ij}} = D_{ij}\sigma'(B_{ij}) = (D \circ \sigma'(B))_{ij}.$$

□

Lemma B.0.2. *For any function* $\mathbf{F} : \mathbb{R}^{m\times n} \to \mathbb{R}$ *that maps matrix to scalar.* A, C *are square matrices and* $Y = ABC$. *If* $\frac{\partial \mathbf{F}(X)}{\partial Y} = D$, *then* $\frac{\partial \mathbf{F}(X)}{\partial B} = A^T D C^T$.

Proof.

$$\begin{aligned}
\frac{\partial \mathbf{F}(X)}{\partial B_{ij}} &= \sum_{k,l} (\frac{\partial \mathbf{F}(X)}{\partial ABC})_{k,l} \frac{\partial (ABC)_{k,l}}{\partial B_{ij}} \\
&= \sum_{k,l} D_{kl} A_{ki} C_{jl} \\
&= \sum_{k,l} A^T_{ik} D_{kl} C^T_{lj} \\
&= (A^T D C^T)_{ij}.
\end{aligned}$$

□

Let $M^{(d)} = \frac{\partial \mathbf{R}(W,\boldsymbol{b})}{\partial C^{(d)}}$. By using B.1 and lemma B, I have

$$\begin{aligned}
\frac{\partial \mathbf{R}(W,\boldsymbol{b})}{\partial \boldsymbol{b}^{(d)}} &= 1_{1,n} M^{(d)}, \\
\frac{\partial \mathbf{R}(W,\boldsymbol{b})}{\partial W^{(d)}} &= (X^{(d-1)} B^{(d-1)})^T M^{(d)} / m^{(d-1)}.
\end{aligned} \tag{B.2}$$

For the last layer, $\sigma^{(D)}$ is a linear function, i.e. $\hat{Y} = C^{(D)}B^{(D)T}$. I then have

$$\frac{\partial \mathbf{R}(W, \boldsymbol{b})}{\partial \hat{Y}} = \frac{\partial \sum_{i,j}(\hat{y}_{ij} - y_{ij})^2}{\partial \hat{Y}} == 2(\hat{Y} - Y),$$
$$M^{(D)} = 2(\hat{Y} - Y)B^{(D)}. \tag{B.3}$$

For other layers, by using B.1, I have

$$\begin{aligned} C^{(d+1)} &= (X^{(d)}B^{(d)}/m^{(d)})W^{(d+1)} + 1_{n,1}\boldsymbol{b}^{(d+1)} \\ &= \sigma^{(d)}(C^{(d)}B^{(d)T})B^{(d)}W^{(d+1)}/m^{(d)} + 1_{n,1}c^{(d+1)}. \end{aligned}$$

Based on lemma B.0.1 and B, I have

$$\begin{aligned} &\frac{\partial \mathbf{R}(W, \boldsymbol{b})}{\partial \sigma^{(d)}(C^{(d)}B^{(d)T})} = M^{(d+1)}W^{(d+1)T}B^{(d)T}/m^{(d)}, \\ &\frac{\partial \mathbf{R}(W, \boldsymbol{b})}{\partial (C^{(d)}B^{(d)T})} = M^{(d+1)}W^{(d+1)T}B^{(d)T} \circ \sigma'(C^{(d)}B^{(d)T})/m^{(d)}, \\ &M^{(d)} = [(M^{(d+1)}W^{(d+1)T}B^{(d)T}) \circ \sigma'(C^{(d)}B^{(d)T})]B^{(d)}/m^{(d)}. \end{aligned} \tag{B.4}$$

The formula equation (B.2), equation (B.3) and equation (B.4) are then used for calculating back propagation on cost function $\mathbf{R}(W, \boldsymbol{b})$. As for the penalty term $J(W, \boldsymbol{b})$, I have equation (B.5), which is illustrated below,

$$\begin{aligned} P_0^{(d)} &= [\int \beta_i^{(d)}(t^{(d)})\beta_j^{(d)}(t^{(d)})\mathrm{d}t^{(d)}]_{i=1,j=1}^{l^{(d)},l^{(d)}}, \\ P_2^{(d)} &= [\int \beta_i^{(d)''}(t^{(d)})\beta_j^{(d)''}(t^{(d)})\mathrm{d}t^{(d)}]_{i=1,j=1}^{l^{(d)},l^{(d)}} \end{aligned}$$

$$
\begin{aligned}
J^{(d)} =& \int\int \left[\left(\frac{\partial^2 \alpha^{(d)}}{\partial t^{(d)2}} \right)^2 + \left(\frac{\partial^2 \alpha^{(d)}}{\partial t^{(d-1)2}} \right)^2 \right] \mathrm{d}t^{(d)} \mathrm{d}t^{(d-1)} + \int \left(\frac{\partial^2 \alpha_0^{(d)}}{\partial t^{(d)2}} \right)^2 \mathrm{d}t^{(d)} \\
=& \mathrm{tr} \left(P_0^{(d-1)} W^{(d)} P_2^{(d)} W^{(d)\prime} + P_0^{(d)} W^{(d)\prime} P_2^{(d-1)} W^{(d)} \right) + \mathrm{tr} \left(\boldsymbol{b}^{(d)} P_2^{(d)} \boldsymbol{b}^{(d)\prime} \right)
\end{aligned}
$$

Therefore, the final derivatives for back propagation are:

$$
\begin{aligned}
\frac{\partial J^{(d)}}{\partial W^{(d)}} =& 2 P_0^{(d-1)} W^{(d)} P_2^{(d)} + 2 P_2^{(d-1)} W^{(d)} P_0^{(d)}, \\
\frac{\partial J^{(d)}}{\partial b^{(d)}} =& 2 b^{(d)} P_2^{(d)}.
\end{aligned}
\tag{B.5}
$$

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[2] Clare Bycroft, Colin Freeman, Desislava Petkova, Gavin Band, Lloyd T Elliott, Kevin Sharp, Allan Motyer, Damjan Vukcevic, Olivier Delaneau, Jared O'Connell, et al. The uk biobank resource with deep phenotyping and genomic data. *Nature*, 562(7726):203–209, 2018.

[3] 1000 Genomes Project Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061, 2010.

[4] Heather J Cordell. Detecting gene–gene interactions that underlie human diseases. *Nature Reviews Genetics*, 10(6):392–404, 2009.

[5] Murat Dikmen and Catherine M Burns. Autonomous driving in the real world: Experiences with tesla autopilot and summon. In *Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications*, pages 225–228, 2016.

[6] Gökcen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

[7] Ruzong Fan, Yifan Wang, James L Mills, Alexander F Wilson, Joan E Bailey-Wilson, and Momiao Xiong. Functional linear models for association analysis of quantitative traits. *Genetic epidemiology*, 37(7):726–742, 2013.

[8] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[10] Sara Goodwin, John D McPherson, and W Richard McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333, 2016.

[11] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[12] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[13] Mahbub Hussain, Jordan J Bird, and Diego R Faria. A study on cnn transfer learning for image classification. In *UK Workshop on Computational Intelligence*, pages 191–202. Springer, 2018.

[14] Andrada E Ivanescu, Ana-Maria Staicu, Fabian Scheipl, and Sonja Greven. Penalized function-on-function regression. *Computational Statistics*, 30(2):539–568, 2015.

[15] Adam Kiezun, Kiran Garimella, Ron Do, Nathan O Stitziel, Benjamin M Neale, Paul J McLaren, Namrata Gupta, Pamela Sklar, Patrick F Sullivan, Jennifer L Moran, et al. Exome sequencing and the genetic basis of complex traits. *Nature genetics*, 44(6):623, 2012.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Vasundhara Rathod, and Shreya Bisen. Implementation of a chatbot system using ai and nlp. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST) Volume-6, Issue-3*, 2018.

[18] Torrey Lisa and S Jude. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends*, pages 242–264. IGI Global, 2010.

[19] Jason Z Liu, Federica Tozzi, Dawn M Waterworth, Sreekumar G Pillai, Pierandrea Muglia, Lefkos Middleton, Wade Berrettini, Christopher W Knouff, Xin Yuan, Gérard Waeber, et al. Meta-analysis and imputation refines the association of 15q25 with smoking quantity. *Nature genetics*, 42(5):436–440, 2010.

[20] Teri A Manolio, Francis S Collins, Nancy J Cox, David B Goldstein, Lucia A Hindorff, David J Hunter, Mark I McCarthy, Erin M Ramos, Lon R Cardon, Aravinda Chakravarti, et al. Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753, 2009.

[21] Colin D Mathers and Dejan Loncar. Projections of global mortality and burden of disease from 2002 to 2030. *PLoS medicine*, 3(11):e442, 2006.

[22] Melinda C Mills and Charles Rahal. A scientometric review of genome-wide association studies. *Communications biology*, 2(1):1–11, 2019.

[23] Kalyanapuram Rangachari Parthasarathy. *Probability measures on metric spaces*, volume 352. American Mathematical Soc., 2005.

[24] James O Ramsay and CJ Dalzell. Some tools for functional data analysis. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):539–561, 1991.

[25] JO Ramsay. When the data are functions. *Psychometrika*, 47(4):379–396, 1982.

[26] JO Ramsay and BW Silverman. Functional data analysis-methods and case studies, 2002.

[27] Fabrice Rossi, Nicolas Delannay, Brieuc Conan-Guez, and Michel Verleysen. Representation of functional data in neural networks. *Neurocomputing*, 64:183–210, 2005.

[28] Andrew J Saykin, Li Shen, Tatiana M Foroud, Steven G Potkin, Shanker Swaminathan, Sungeun Kim, Shannon L Risacher, Kwangsik Nho, Matthew J Huentelman, David W Craig, et al. Alzheimer's disease neuroimaging initiative biomarkers as quantitative phenotypes: Genetics core aims, progress, and plans. *Alzheimer's & dementia*, 6(3):265–273, 2010.

[29] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.

[30] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[31] Aad W Van Der Vaart and Jon A Wellner. Weak convergence. In *Weak convergence and empirical processes*, pages 16–28. Springer, 1996.

[32] Peter M Visscher, Naomi R Wray, Qian Zhang, Pamela Sklar, Mark I McCarthy, Matthew A Brown, and Jian Yang. 10 years of gwas discovery: biology, function, and translation. *The American Journal of Human Genetics*, 101(1):5–22, 2017.

[33] Olga A Vsevolozhskaya, Dmitri V Zaykin, David A Barondess, Xiaoren Tong, Sneha Jadhav, and Qing Lu. Uncovering local trends in genetic effects of multiple phenotypes via functional linear models. *Genetic epidemiology*, 40(3):210–221, 2016.

[34] Olga A Vsevolozhskaya, Dmitri V Zaykin, Mark C Greenwood, Changshuai Wei, and Qing Lu. Functional analysis of variance for association studies. *PLoS One*, 9(9):e105074, 2014.

[35] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

[36] Paul Yushkevich, Stephen M Pizer, Sarang Joshi, and James Stephen Marron. Intuitive, localized analysis of shape variability. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 402–408. Springer, 2001.

[37] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[38] Xuefen Zhang, Tongtong Lan, Tong Wang, Wei Xue, Xiaoran Tong, Tengfei Ma, Guifen Liu, and Qing Lu. Considering genetic heterogeneity in the association analysis finds genes associated with nicotine dependence. *Frontiers in genetics*, 10:448, 2019.

[39] James Zou, Mikael Huss, Abubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti. A primer on deep learning in genomics. *Nature genetics*, 51(1):12–18, 2019.