UNIVERSITY OF CALIFORNIA
RIVERSIDE

Topics in Artificial Neural Networks: Causal Inference and Functional Derivative Estimation

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Applied Statistics

by

Ying Liu

September 2022

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Shujie Ma, who guided me throughout my Ph.D. study. Without her guidance, patience, encouragement and support, I would not be able to complete this dissertation. Her guidance not only helped me in all the time of research and writing of this thesis, but also taught me how to become as an independent statistician.

Besides my advisor, I would like to thank my oral exam and committee members: Dr. Subir Ghosh, Dr. Daniel Jeske, Dr. Esra Kurum, Dr. Yehua Li, Dr. Gloria Gonzalez-Rivera, for their constructive suggestions and insightful comments, but also for the questions which encouraged me to dive deeper into my research from various perspectives. Additionally, I would like to sincerely thank all professors who have been my instructors during my study at UCR for helping me to develop my background in statistics.

I have to thank my collaborators, Dr. Xiaohong Chen and Dr. Zheng Zhang for their significant contribution and immersive insights during my research. I am also grateful to the staff and fellow students in the Department of Statistics during my time at Riverside. Their kind guidance helped me overcome the challenges of going back to school after working for several years.

Finally, I am thankful to both friends and family whose support helped me get through the hard times.

In dedication to my family.

ABSTRACT OF THE DISSERTATION

Topics in Artificial Neural Networks: Causal Inference and Functional Derivative Estimation

by

Ying Liu

Doctor of Philosophy, Graduate Program in Applied Statistics
University of California, Riverside, September 2022
Dr. Shujie Ma, Chairperson

Advances in computer science technologies have shed light on artificial neural networks (ANN). ANN shows its power and efficiency in various classification and regression problems and is gaining more and more popularity in various fields. In this dissertation, we investigate treatment effects estimation using fully connected shallow network, and explore sparse deep neural network regression and functional derivative estimation. The first part of this dissertation provides a unified framework for efficient estimation of various types of treatment effects (TE) in observational data with a diverging number of covariates through a generalized optimization. We show that the number of confounders is allowed to increase with the sample size, and further investigate how fast it can grow with the sample size to ensure root-n consistency of the resulting TE estimator. Moreover, we establish asymptotic normality and semiparametric efficiency of the TE estimator. The second part of this dissertation proposes a penalized deep ReQU network estimator (PDRN) obtained from empirical risk minimization framework. The proposed neural network bases on Jacobi polynomial approximation on the hyperbolic cross/sparse grid and alleviates the "curse of dimensionality". Our PDRN estimator also provides smooth functional derivative estimation. Our estimators are illustrated through simulation studies and multiple real data examples.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid growth in the volume of data has brought new opportunities, as well as tough challenges to data analysis. To study the unknown pattern between the predictors and the target response variables, many technologies are developed and widely applied. For instance, parametric models such as linear regression models are very convenient and easy to interpret, however, in the sacrifice of flexibility. The mis-specified models will lead to large bias and false conclusions. Classical nonparametric methods such as kernels or splines are flexible for recovering unknown patterns, but they are not comfort with a large number of predictor variables. In recent days, the advances in computer science technologies has accelerated the emerge of new machine learning methods, as well as shed light on some pre-existing methods like artificial neural networks (ANN).

ANN shows its power and efficiency in various classification and regression problems and is gaining more and more popularity in various fields such as computer vision, image recognition, speech recognition and bioinformatics [4]. The theoretical properties of

ANN have also been an increased focus. ANNs are universal approximators of a wide variety of functions [122, 65, 35, 126, 15, 102] , so they are robust to mis-specification. It is shown in [35] that their approximation rate to a smooth function can be smaller than $n^{-1/4}$, where n is the sample size, no matter how large the dimension of covariates is, indicating that ANNs have the potential to overcome the "curse of dimensionality" that typically arises in classical nonparametric estimation approaches. ANNs are shown to be particularly useful for classification and prediction from large datasets [7]. A review on the basic set up of ANN and estimation algorithm is given in Chapter 2.

Theoretical properties of ANN based estimators have also received increasing attention, for instance, how to conduct causal inference using ANNs? Chapter 3 proposes a new unified approach for efficient estimation of treatment effects using ANNs when the number of covariates is allowed to increase with the sample size. We consider a general optimization framework that includes the average, quantile and asymmetric least squares treatment effects as special cases. Under this unified setup, we develop a generalized optimization estimator for the treatment effect with the nuisance function estimated by neural networks. We further establish the consistency and asymptotic normality of the proposed estimator and show that it attains the semiparametric efficiency bound. The proposed methods are illustrated via simulation studies and a real data application.

Though it is known that using only one hidden layer, ANN can provide an optimal order of approximation for functions which satisfy certain smoothness conditions ([35], [89]), it is also proved that deep ReLU networks have better approximation than shallow ReLU networks [125]. In the meantime, sparse grids approximation is gaining more popularity as an approximation tool due to the rapid growth in the volume of high-dimensional data. The connection between sparse grids and deep neural networks has been built in [90] and [77]. While ANN is widely applied in regression problems, derivative estimation has received less attention. Derivative estimation is essential in many situations including comparing regression curves [97], analyzing significant trends [99] and identifying change points in longitudinal data [107]. For certain nonparametric methods and machine learning methods applied in regression problems, there is no analytical form solution to the estimated regression function and considered as "black boxes", thus we can't estimate the functional derivatives. Chapter 4 proposes an ANN approach for both regression and functional derivative estimation. Our estimator of the target function is built upon a network architecture of sparsely-connected deep neural networks with the rectified power unit (RePU) activation function. The proposed methods are illustrated via simulation studies and two real data applications.

Chapter 5 summarizes this dissertation with concluding remarks. The related technical proofs are included in the Appendix.

Chapter 2

Neural Networks, Estimation Algorithm and High-dimensional Regression

2.1 Neural Networks

The history of artificial neural network may date back to 1940 [88]. This machine learning method, inspired by interactions among neurons within the brain, is gaining more and more polarities due to the rapid advance in computation technologies. "An artificial neural network consists of a collection of simulated neurons. Each neuron is a node which is connected to other nodes via links that correspond to biological axon-synapse-dendrite connections. Each link has a weight, which determines the strength of one node's influence on another [123]". There are changing size of network and number of neurons, various type of“links (activation functions)”, as well as flexibility in the node connection structure and in the mechanism of one node’s influence on another, resulting in multiple types of neural networks, including feedforward neural network, convolutional neural network, recurrent neural network and etc. First, we introduce the basic setup of feedforward neural network which would be the main focus of this dissertation.

2.1.1 Basic Setup

The feedforward neural networks move in one direction from the input variables of dimension d, through the hidden nodes, and to the ouput node. The feedforward neural network with L hidden layers is given as

$$\Phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{N_{L+1}};\ \Phi(\boldsymbol{x}) = \boldsymbol{z}_{L+1},$$

where $\Phi(\boldsymbol{x}; \eta)$ is defined as $\boldsymbol{z}_0 = \boldsymbol{x}$, $\boldsymbol{z}_\ell = \sigma(\boldsymbol{A}_\ell \boldsymbol{z}_{\ell-1} + \boldsymbol{b}_\ell)$ for $\ell = 1, ..., L$, and $\boldsymbol{z}_{L+1} = \boldsymbol{A}_{L+1}\boldsymbol{z}_L + \boldsymbol{b}_L$, $\sigma(\cdot)$ is the activation function, and $\boldsymbol{A}_\ell$ are $N_\ell \times N_{\ell-1}$ matrices and $\boldsymbol{b}_\ell \in \mathbb{R}^{N_\ell}$ with $N_0 = d$ and $N_\ell \in \mathbb{N}$ for $\ell = 1, ..., L{+}1$. Moreover, for any vector $\boldsymbol{v} = (v_1, ..., v_N)^\top \in \mathbb{R}^N$, $\sigma(\boldsymbol{v}) = \{\sigma(v_1), ..., \sigma_2(v_N)\}^\top$.

2.1.2 Activation Functions

The choice of activation function is essential for neural networks since it has a large impact on how data is transformed in each layer and further on the properties of neural networks. Four types of activation function, which can be categorized as sigmoid-like functions and rectified functions, are introduced in this section and there are more to meet the need in different situation.

1. Sigmoid function: Since the range of sigmoid function is (0, 1), it is especially used for models where we have to predict the probability as an output. The S-shaped function is monotonic and differentiable, however it has the "vanishing gradient" problem, which means the gradient is almost zero when x is very large or very small. The maximum value of the derivative of sigmoid is 0.25 and the minimum is 0, so that the neural network is difficult to learn from data.

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

2. Tahn (hyperbolic tangent) function: it is very similar to sigmoid function as it is also S-shaped with the range being (-1, 1). It is monotonic and differentiable but still suffering from "vanishing gradient".

$$\sigma(x) = \frac{e^{2x} - 1}{e^{2x} + 1}.$$

3. ReLU (Rectified Linear Unit) function: It is rectified since it would be zero if x is negative. The gradient would not vanish since it would always be 1 for positive x. It is monotonic and differentiable except for $x = 0$. In Chapter 3 we will mainly use ReLU as activation function.

$$\sigma(x) = xI(x \geq 0).$$

4. ReQU (Rectified Quadratic Unit) function: Similar to ReLU, it is rectified since it would be zero if x is negative. It is monotonic and differentiable everywhere. In Chapter 4 we will build our neural network with ReQU function.

$$\sigma(x) = x^2 I(x \geq 0).$$

ReQU also belongs to the family of RePU (Rectified Power Unit) functions, whose degree of non-zero part is $s \geq 2$.

$$\sigma(x) = x^s I(x \geq 0).$$

2.2 Estimation Algorithm

2.2.1 Parameter Initialization

The weight $\boldsymbol{A}_\ell$ and bias $\boldsymbol{b}_\ell$ for $\ell = 1, ..., L+1$ of a neural network are estimated through an optimization framework and a good choice of initial values of parameters is necessary for efficient training. While a too-large initialization may result in too-large gradient and the learning process would be unstable, a too-small initialization may result in vanishing gradient and the learning process would be too slow. Here we introduce two initialization methods.

1. Xavier Initialization [50]: the goal of Xavier Initialization is to initialize the weights such that the variance of the activations are the same across every layer. This constant variance helps prevent the gradient from exploding or vanishing. The Xavier Initialization method is calculated as

   $$A_{\ell,ij} \sim \mathcal{N}(0, \frac{1}{n^{[l-1]}}),$$

   where $n^{[\ell-1]}$ is the number of nodes of layer $\ell - 1$, $1 \leq i \leq N_\ell, 1 \leq j \leq N_\ell$. Xavier Initialization is widly used when the activation function is sigmoid function or tahn function. The bias are initialized with zero.

2. He Initialization [58]: weights are initialized in the same way as the Xavier initialization except that variance is by multiplied by 2, and it is commonly used for rectifier, for instance, ReLU function.

2.2.2 ADAM Algorithm

The weight $\boldsymbol{A}_\ell$ and bias $\boldsymbol{b}_\ell$ for $\ell = 1, ..., L+1$ of a neural network are estimated using first-order gradients based algorithm, here we introduce the widely used ADAM algorithm [71]. Denote the vector of parameters of interest by θ and let $f(\theta)$ be a scalar function that is differentiable with respect to the parameter θ. The detailed algorithm is given in Algorithm 1.

2.3 High-dimensional Regression

The increasing volume of large-scale data has challenged the application of neural networks. As the network grows bigger, the number of parameters involved also grows rapidly, which results in over-fitting problem with small bias but large variance in the estimate. One remedy for high-dimensional cases is to use penalized regression. Two popular penalized regression methods are ridge [63] and lasso regression [112]. They both shrink the parameters to zero. Another remedy is to reduce the dimensionality of the original large-scale data, for instance, Principal Component Analysis (PCA) which preserve most of the variability in the original data.

```
**Algorithm 1** ADAM
**Require:** $\theta_{ini}$: Initial value for $\boldsymbol{\theta}$
**Require:** $\epsilon_0$: converge criterion
**Require:** $\tilde{\alpha}$: step size with default value of 0.001
**Require:** $\tilde{\beta}_1$: decay rate with default value of 0.9
**Require:** $\tilde{\beta}_2$: decay rate with default value of 0.999
**Require:** $\tilde{\epsilon}$: stabilizer with default value of $10^{-8}$
  $t \leftarrow 0$ (Initialize timestep)
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $\theta_0 \leftarrow \theta_{ini}$
  **while** $\theta_t$ not converge **do**
      $t \leftarrow t + 1$
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Obtain gradient for $\theta$ at timestep $t$)
      $m_t \leftarrow \tilde{\beta}_1 m_{t-1} + (1 - \tilde{\beta}_1) g_t$ (Update biased first moment estimate)
      $v_t \leftarrow \tilde{\beta}_2 v_{t-1} + (1 - \tilde{\beta}_2) g_t \odot g_t$ (Update biased second raw moment estimate))
      $\widehat{m}_t \leftarrow m_t / (1 - \tilde{\beta}_1^t)$ (Compute bias-corrected first moment estimate)
      $\widehat{v}_t \leftarrow v_t / (1 - \tilde{\beta}_2^t)$ (Compute bias-corrected second raw moment estimate)
      $\Delta(\theta_t) \leftarrow \widehat{m}_t / (\sqrt{\widehat{v}_t} + \tilde{\epsilon})$
      $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} - \tilde{\alpha} \Delta(\boldsymbol{\alpha}_t)$ (Update $\theta$ at timestep $t$)
  **end while**
      **return** $\theta_t$
```

2.3.1 Penalizaed Regression

In a penalized regression setting, suppose we want to obtain the estimate for parameter $\theta \in \mathbb{R}^k$ through an optimizing problem:

$$L(\theta) = f(\theta) + \sum_{j=1}^{k} p(\lambda, \theta_j),$$

where $f(\theta)$ is the loss function, $p(\lambda, \theta_j)$ is the penalty function and λ is the tuning parameter controlling the strength of the penalty term. In ridge regression, the penalty function $\sum_{j=1}^{k} p(\lambda, \theta_j) = \lambda \sum_{j=1}^{k} \theta_j^2$. As λ increases, the θ_j are moving toward zero, the bias in the estimator increases but the variance decreases. Another type of penalty is the lasso penalty, which penalizes the absolute value of θ: $\sum_{j=1}^{k} p(\lambda, \theta_j) = \lambda \sum_{j=1}^{k} |\theta_j|$. Different with ridge penalty which will only cause θ to decrease but never be zero, lasso penalty can shrink some θ_j to be exact zero. Thus lasso penalty is applied in the situation that we want to perform variable selection. The penalty function $\sum_{j=1}^{k} p(\lambda, \theta_j)$ changes as to meet the need of different situations. In Chapter 4 we use a ridge-like penalty term to control the complexity of our estimator.

Chapter 3

Efficient Estimation of General Treatment Effects Using Neural Networks

3.1 Introduction

The estimation of causal effects is a primary goal of behavioral, social, economic and biomedical sciences. Recent technological advances have created numerous large-scale observational studies, which bring unprecedented opportunities for evaluating the treatment effectiveness. Examples of such data include patient registries, electronic health records, pharmacy and health insurance claims and user-generated social media platforms, all of which are increasingly available in large volumes. The increase occurs not only in the number of sample observations, but also in the number of variables measured for each subject. Moreover, the usage of a large number of covariates in casual inference is justified to increase the plausibility of the no unmeasured confounding assumption.

A major difficulty in causal inference from observational studies is how to control the bias caused by the confounding variables that influence both the outcome and treatment assignment. To overcome this difficulty, under the unconfounded treatment assignment condition [100], one often needs an intermediate estimate of unknown nuisance functions that relate outcome and/or treatment to confounders [60, 62, 29, 3, 40, 55] . The mis-specification of the parametric approaches can introduce serious bias into casual effect estimation [69, 49], which is a big concern in the context of large-scale data. Although the classical nonparametric methods such as kernels or splines are flexible for recovering unknown functions, they suffer from the "curse of dimensionality" [16]. On the other hand, the unconfounded treatment assignment requires that all observed confounders be included in the analysis, as we often have no prior knowledge of which variables are important confounders. Thus, there is a pressing need to apply a data-driven method that can provide effective protection against mis-specification bias as well as achieving dimension reduction. Some proposals have made initial attempts to solve this problem using the sufficient dimension reduction [67, 82, 85]. This technique requires the dependence of treatment assignment on confounders through a few linear combinations of them.

Thanks to the rapid development of scalable computing and optimization techniques in recent years [71, 1], it becomes appealing to use artificial neural networks (ANNs) to approximate the nuisance functions. Similar as splines, ANNs are also a class of approximation bases, but they can contain multilayers. ANNs are universal approximators of a wide variety of functions [122, 65, 35, 126, 15, 102] , so they are robust to mis-specification. It is shown in [35] that their approximation rate to a smooth function can be smaller than $n^{-1/4}$, where n is the sample size, no matter how large the dimension of covariates is, indicating that ANNs have the potential to overcome the "curse of dimensionality" that typically arises in classical nonparametric estimation approaches. ANNs are shown to be particularly useful for classification and prediction from large datasets [7]. However, how do we go one step further to conduct causal inference using ANNs? It needs careful thought, and research on this topic is still in its infancy.

In this dissertation, we propose a new ANN-based estimator of general TEs. Our TE estimator is directly obtained through optimizing a generalized objective function that only involves the propensity score (PS) function [100], which is approximated by ANNs. As a result, it can be naturally used to estimate general TEs, including the average, quantile and asymmetric least squares TEs. Our estimation procedure enables us to easily adopt a convenient weighted bootstrap procedure for conducting inference without the need of estimating the asymptotic variance, which is nontrivial in general.

Theoretically, we derive a new convergence rate of the ANN estimator for the nuisance function under mild conditions when the number of confounders is allowed to grow with the sample size. It has been shown in the literature [12, 65, 35, 72] that when the target function admits a Fourier representation with a bounded moment, its ANNs approximator enjoys a fast approximation rate, making ANNs a promising tool to potentially break the notorious "curse of dimensionality" in multivariate nonparametric regression. However, this Fourier function class is less commonly used than the smoothness spaces such as Hölder and

Sobolev spaces [109, 120, 66, 84] in the nonparametric regression literature. Moreover, it is still unclear how the moment of the Fourier transform appeared in the approximation error bounds depends on the data dimension. This moment is simply treated as a constant in the existing works [12, 35, 72] as they consider fixed dimensions. In this dissertation, we introduce a mixed Sobolev space. We show that when the target function belongs to this mixed Sobolev space, it is also in the Fourier function class. We further derive an upper bound for the moment of the Fourier transform, in terms of the data dimension. Functions in this mixed Sobolev space need to be at least one order smoother in each coordinate than those in the conventional Sobolev space, over which minimax optimal rates for estimation have been established and no nonparametric estimator can avoid the "curse of dimensionality" [102]. We build a connection between the Fourier function class used for ANNs and a mixed Sobolev space. Neither this connection nor the upper bound of the Fourier transform moment has not been investigated in the literature. We also show that the conventional linear sieve approximators still have the dimensionality problem when the targe function belongs to the mixed Sobolev space. Moreover, different from [35], our ANNs class no longer requires the L_1 norm constraint on the weights. This greatly facilitate the computation of the ANNs estimates. Our new theoretical results play a crucial role in helping researchers better understand the required conditions for ANNs and figure out how exactly the convergence rate depends on the data dimension.

We develop asymptotic normality of our proposed TE estimator without the need of assuming that the PS function is bounded below by a constant. This is a strong assumption, especially in the settings with a large number of confounders, albeit still used in the literature, see for example [10]. We find that the number of the confounders is allowed to grow with the sample size with a rate no greater than $\{\log(n)\}^{1/2}$ to ensure the desirable asymptotic property of the ANNs-based TE estimator for conducting inference. Different from the existing works [98, 30, 31, 3], our semiparametric inferential theory is built upon the settings with diverging dimensions. To the best of our knowledge, our work is the first one that considers a growing dimension of the covariates with respect to the sample size for conducting casual inference when the nuisance function is approximated by ANNs under mild smoothness conditions, and proposes a generalized optimization approach that can estimate different types of TEs without estimating the efficient influence function in the presence of a large number of covariates. While the development of credible inferential theories for the ANN-based estimator of TEs is essential to test the significance of the treatment effects, it is also a daunting task because of the complex nonlinear structure of the ANNs. It is worth noting that if our interest focuses on the average TE specifically, we also propose an ANN-based estimator obtained from the outcome regression (OR) function. This estimator can be more robust than the PS based estimator in case that the estimated PS function has very small values. However, it is difficult to apply this OR based estimator to other types of TEs such as quantile TEs. In the context of average TE, our proposed PS and OR estimators have the same asymptotic distribution. To better illustrate our TE estimation procedure, we focus on using the ANNs with one-hidden layer to construct the TE estimator, and discuss the extension of our method to ANNs with multiple hidden layers and its statistical properties in the online Supplemental Materials [33].

It is worth mentioning that the doubly robust (DR)-based machine learning methods for estimation of TEs [20, 19, 36] have received considerable attention. One recent work [46] proposed to estimate the average TEs using the DR method with the nuisance functions approximated by neural networks, and provides a sound theoretical justification for their method. They assume that the nuisance functions belong to the regular Hölder space, so their proposed framework still requires the dimension be fixed. The DR estimators of TEs are constructed based on efficient influence functions, so they arise naturally for pursuing asymptotic normality, which are of critical importance for conducting casual inference [e.g. 115, 11, 26, 110, 116, 101, 28, 70, 111, 94]. The DR method is mainly applied for average TE estimation because that estimation of the efficient influence functions for the average TE is straightforward, but it can be complicated for other types of TEs, such as quantile TEs. Moreover, random forests are another attractive machine learning tool that enjoys flexibility for unknown function approximation, and have been applied by pioneer works [118, 9] for causality analysis. They propose random forest estimators for conditional TEs localizing around given values of confounders and their estimators enjoy local asymptotic normality, while we focus on efficient estimation of population TEs. Moreover, it enjoys computational convenience, especially in complicated settings such as quantile TEs, which is of essential importance when the nuisance functions are trained by machine learning algorithms.

This chapter is organized as follows. Section 3.2 sets up the basic framework, Section 3.4 describes the artificial neural networks, our proposed inverse probability weighting estimator for TEs, and establishes the large sample properties of the proposed estimator, Section 3.5 describes the outcome regression estimator for TEs, Section 3.6 reports the numerical results of simulation studies, and Section 3.7 illustrates the proposed method using a data example, followed by some concluding remarks in Section 3.8. All the technical proofs are provided in the online Supplemental Materials [33].

3.2 Basic Framework

Let D denote a treatment variable taking value in $\mathcal{D} = \{0, 1, ..., J\}$, where $J \geq 1$ is a positive integer. Let $Y^*(d)$ denote the potential outcome when the treatment status $D = d$ is assigned. The probability density of $Y^*(d)$ exists, denoted by $f_{Y^*(d)}$, is continuously differentiable. Let $\mathcal{L}(\cdot)$ denote a nonnegative and strictly convex loss function satisfying $\mathcal{L}(0) = 0$ and $\mathcal{L}(v) \geq 0$ for all $v \in \mathbb{R}$. The derivative of $\mathcal{L}(\cdot)$, denoted by $\mathcal{L}'(\cdot)$, exists almost everywhere and non-constant. Let $\boldsymbol{\beta}^* = (\beta_0^*, \beta_1^*, \ldots, \beta_J^*)^\top \in \mathbb{R}^{J+1}$ be the parameter of interest which is uniquely identified through the following optimization problem:

$$\boldsymbol{\beta}^* := \arg\min_{\boldsymbol{\beta}} \sum_{d=0}^{J} \mathbb{E}\left[\mathcal{L}\left(Y^*(d) - \beta_d\right)\right], \tag{3.1}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_J)^\top \in \mathbb{R}^{J+1}$. The formulation (3.1) permits the following important already considered models and much more:

- $\mathcal{L}(v) = v^2$ and $J = 1$, then $\beta_0^* = \mathbb{E}[Y^*(0)]$ and $\beta_1^* = \mathbb{E}[Y^*(1)]$, and $\beta_1^* - \beta_0^*$ is the average treatment effects (ATE) studied by [54], [62] and [29]. When $J \geq 2$, then $\beta_d^* = \mathbb{E}[Y^*(d)]$ is the multi-valued treatment effects studied by [27].

- $\mathcal{L}(v) = v \cdot \{\tau - \mathbb{1}(v \leq 0)\}$ for some $\tau \in (0, 1)$ and $J = 1$, then $\beta_0^* = F_{Y^*(0)}^{-1}(\tau)$ and $\beta_1^* = F_{Y^*(1)}^{-1}(\tau)$, and $\beta_1^* - \beta_0^*$ is the quantile treatment effects [QTE 47, 56].

- $\mathcal{L}(v) = v^2 \cdot |\tau - \mathbb{1}(v \leq 0)|$ is the asymmetric least square effects [124].

The problem with (3.1) is that the potential outcomes $(Y^*(0), Y^*(1), ..., Y^*(J))$ cannot all be observed. The observed outcome is denoted by $Y := Y^*(D) = \sum_{d=0}^{J+1} \mathbb{1}(D = d)Y^*(d)$. One may attempt to solve the problem:

$$\min_{\boldsymbol{\beta}} \sum_{d=0}^{J} \mathbb{E}\left[\mathcal{L}\left(Y - \beta_d\right)\right].$$

However, due to the selection in treatment, the true value $\boldsymbol{\beta}^*$ is not the solution of the above problems. To address this problem, most literature imposes *unconfoundedness* condition [100, 27]. This work considers high-dimensional confounders with p going to infinity as the sample size increases. Specifically, we work with triangular array data $\{((D_{i,n}, \boldsymbol{X}_{i,n}, Y_{i,n}), i = 1, ..., n), n = 1, 2, ...\}$ defined on some common probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Each $\boldsymbol{X}_{i,n}$ is a vector whose dimension p_n may grow with n. For each given n, these vectors are independent across i, but not necessarily identically distributed. The law $\mathbb{P}_n$ of $\{(D_{i,n}, \boldsymbol{X}_{i,n}, Y_{i,n}), i = 1, ..., n\}$ can change with n, though we do not make explicit use of $\mathbb{P}_n$. Thus, all parameters (including p_n) that characterize the distribution of $\{(D_{i,n}, \boldsymbol{X}_{i,n}, Y_{i,n}), i = 1, ..., n\}$ are implicitly indexed by the sample size n, but we omit the index n in what follows to simplify notation. The following condition shall be maintained through this chapter:

Assumption 1. *For each $d \in \mathcal{D}$, $Y_i^*(d) \perp D_i | \boldsymbol{X}_i$.*

Under Assumption 1, the causal parameters $\boldsymbol{\beta}^*$ can be identified by the minimizer of the following optimization problem:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \sum_{d=0}^{J} \mathbb{E}\left[\frac{\mathbb{1}(D_i = d)}{\pi_d^*(\boldsymbol{X}_i)} \mathcal{L}\left(Y_i - \beta_d\right)\right], \tag{3.2}$$

where $\pi_d^*(\boldsymbol{X}_i) := P(D_i = d|\boldsymbol{X}_i)$ is the PS function which is unknown in practice.

Based on (3.2), existing approaches rely on parametric or nonparametric estimation of the PS function $\pi_d^*(\cdot)$. Parametric methods suffer from model misspecification problem, while conventional nonparametric methods, such as linear sieve or kernel regression, fail to work if the dimension of covariates p is large which is known as the *curse of dimensionality* [78]. The goal of this chapter is to efficiently estimate $\boldsymbol{\beta}^*$ under this general framework when the dimension of covariates p is large, and it possibly increases as the sample size n grows. We propose to estimate the PS function $\pi_d^*(\cdot)$ using feedforward ANNs with one hidden layer described below.

3.3 New Approximation Error Bounds for ANNs with Diverging Dimension

Feedforward ANNs are effective tools for solving the classification and prediction problems for "big data". The basic idea is to extract linear combinations of the inputs as features, and then model the target as a nonlinear function of these features. This section presents ANNs with one hidden layer and the related results which are used in this chapter.

Let $\mathcal{X}$ denote the support of $\boldsymbol{X}_i$ which is compact in $\mathbb{R}^p$. Without loss of generality, we assume $\mathcal{X} = [0,1]^p$. Let F_X be the cumulative distribution function (CDF) of $\boldsymbol{X}_i$. Denote the $L^2(dF_X)$-norm of any function $f(x)$ by $\|f\|_{L^2(dF_X)} := \left\{\int_{\mathcal{X}} |f(x)|^2 dF_X(x)\right\}^{1/2}$.

[12], [65], [35], and [72] considered the target function belonging to the function class $\mathcal{F}_p$:

$$\mathcal{F}_p := \left\{ f : \mathcal{X} \to \mathbb{R} : f(x) = \int_{\mathbb{R}^p} \exp\left(ia^\top x\right) \widetilde{f}(a) da \right\}, \tag{3.3}$$

where $\widetilde{f}(a)$ is the Fourier transform of $f(x)$ defined by

$$\widetilde{f}(a) := \frac{1}{(2\pi)^p} \int_{\mathbb{R}^p} \exp\left(-ia^\top x\right) f(x) dx.$$

$\mathcal{F}_p$ contains a class of functions of p dimension that admit Fourier representations. The input variables of functions in $\mathcal{F}_p$ have dimension p, which is allowed to grow to infinity as the sample size n increases. We define the m^{th} moment of the Fourier transform of $f(x)$ as $v_{f,m} := \int_{\mathbb{R}^p} |a|_1^m \left|\widetilde{f}(a)\right| da$, where $|a|_1 := \sum_{i=1}^p |a_i|$. It is worth noting that $v_{f,m}$ depends on the dimension p, and its value can increase with p. In the nonparametric regression literature, spaces with certain smoothness constraints such as Hölder and Sobolev spaces are instead more commonly used [109, 120, 66, 84].

We consider the mixed Sobolev space $\mathcal{W}^{s,\delta_0,\infty}(\mathcal{X})$ for $s, \delta_0 \geq 0$ defined by

$$\mathcal{W}^{s,\delta_0,\infty}(\mathcal{X}) := \left\{ f : \mathcal{X} \to \mathbb{R} : \max_{|\boldsymbol{\delta}|_\infty \leq \delta_0} \max_{|\boldsymbol{\alpha}|_1 \leq s} \sup_{x \in \mathcal{X}} \left| D^{\boldsymbol{\alpha}+\boldsymbol{\delta}} f(x) \right| \leq 1 \right\},$$

where $\boldsymbol{\alpha} := (\alpha_1, ..., \alpha_p)$, $\boldsymbol{\delta} := (\delta_1, ..., \delta_p)$, $|\boldsymbol{\alpha}|_1 := \sum_{j=1}^p \alpha_j$, $|\boldsymbol{\delta}|_\infty := \max\{\delta_1, ..., \delta_p\}$, and

$$D^{\boldsymbol{\alpha}+\boldsymbol{\delta}} f(x) := \frac{\partial^{|\boldsymbol{\alpha}+\boldsymbol{\delta}|_1}}{\partial^{\alpha_1+\delta_1} x_1 \cdots \partial^{\alpha_p+\delta_p} x_p} f(x).$$

We will build a connection between the function class $\mathcal{F}_p^m$ given in (3.3) and a mixed Sobolev space, and will establish an upper bound for $v_{f,m}$ in terms of the dimension p in Theorem 1 in our online Supplemental Materials [33].

Consider to approximate a target function $f \in \mathcal{F}_p$ using the ANNs, belonging to the class

$$\begin{aligned} \mathcal{G}(\psi, B, r, p) = \Bigg\{ & g : g(x) = g_0(x; \boldsymbol{\gamma}_0) + \frac{B}{r} \sum_{j=1}^r \gamma_j \psi(a_j^\top x + a_{j0}),\ a_j = (a_{j1}, ..., a_{jp})^\top \in \mathbb{R}^p, \\ & \|a_j\|_2 \leq 1,\ |a_{j0}| \leq 1,\ |\gamma_j| \leq 1,\ j \in \{1, ..., r\},\ B \in \mathbb{R}^+ \Bigg\}, \end{aligned} \tag{3.4}$$

where $g_0(x;\boldsymbol{\gamma}_0)$ is a parametric function indexed by an unknown parameter vector $\boldsymbol{\gamma}_0$, and $\|a_j\|_2 := \{|a_{j1}|^2 + ... + |a_{jp}|^2\}^{1/2}$. The structure of $g_0(x;\boldsymbol{\gamma}_0)$ depends on the type of the activation function that is used. For example, if the ReLU activation function is used, then $g_0(x;\boldsymbol{\gamma}_0) = \gamma_{00} + \gamma_{01}^\top x$, where $\boldsymbol{\gamma}_0 := (\gamma_{00}, \gamma_{01}^\top)^\top$. $\mathcal{G}(\psi, B, r, p)$ is the collection of output functions for neural networks with p-dimensional input feature x, a single hidden layer with r hidden units and an activation function ψ, real-valued input-to-hidden unit weights (a_j), bias (a_{j0}), and hidden-to-output weights (γ_j).

The approximation error for a target function depends on the smoothness of the approximand, the dimension of the covariates, and the type of approximation basis. We first present the approximation results based on some popularly used neural networks, which have been established in the existing literature:

- (Sigmoid type activation function) Suppose that the function $f \in \mathcal{F}_p$, $g_0(x;\boldsymbol{\gamma}_0) \equiv 0$, the activation function $\psi(\cdot)$ is compactly supported, bounded, and uniformly Lipschitz continuous. If $B \leq 2v_{f,1} < \infty$, then [35] show that the $L^2(dF_X)$-approximation rate of f based on ANN is

$$\inf_{g \in \mathcal{G}(\psi,B,r,p)} \left\{ \int_{\mathcal{X}} |f(x) - g(x)|^2 dF_X(x) \right\}^{1/2} \leq \text{const} \times v_{f,1} \cdot r^{-\frac{1}{2}-\frac{1}{p}}. \qquad (3.5)$$

  The activation functions ψ includes the Heaviside, logistic, tanh, cosine squasher, and other sigmoid functions [65], but does not include the ReLU and squared ReLU ridge functions stated below.

- (ReLU activation function) Suppose that the function $f \in \mathcal{F}_p$, $g_0(x;\boldsymbol{\gamma}_0) = \gamma_{00} + \gamma_{01}^\top x$ for $\boldsymbol{\gamma}_0 = \{\gamma_{00}, \gamma_{01}\} \in \mathbb{R} \times \mathbb{R}^p$, and $\psi(a^\top x + a_0) = (a^\top x + a_0)_+$. If $B \leq 2v_{f,2} < \infty$, then

[72] show that the $L^2(dF_X)$-approximation rate based on ReLU ridge functions is

$$\inf_{g\in\mathcal{G}(\psi,B,r,p)} \left\{ \int_{\mathcal{X}} |f(x)-g(x)|^2 dF_X(x) \right\}^{1/2} \leq \text{const} \times v_{f,2} \cdot r^{-\frac{1}{2}-\frac{1}{p}}. \tag{3.6}$$

- (Squared ReLU activation function) Suppose that the function $f \in \mathcal{F}_p$, $g_0(x;\boldsymbol{\gamma}_0) = \gamma_{00} + \gamma_{01}^\top x + x^\top \gamma_{02} \cdot x$ for $\boldsymbol{\gamma}_0 = \{\gamma_{00}, \gamma_{01}, \gamma_{02}\} \in \mathbb{R} \times \mathbb{R}^p \times \mathbb{R}^{p\times p}$, and $\psi(a^\top x + a_0) = (a^\top x + a_0)_+^2$. If $B \leq 2v_{f,3} < \infty$, then [72] show that the $L^2(dF_X)$-approximation rate based on squared ReLU ridge functions is

$$\inf_{g\in\mathcal{G}(\psi,B,r,p)} \left\{ \int_{\mathcal{X}} |f(x)-g(x)|^2 dF_X(x) \right\}^{1/2} \leq \text{const} \times v_{f,3} \cdot r^{-\frac{1}{2}-\frac{1}{p}}. \tag{3.7}$$

If the target function $f(x)$ is in a Fourier functional class with a bounded moment given in (3.3), then (3.5), (3.6) and (3.7) show that the $L^2(dF_X)$-approximation rates of ANNs are $O(v_{f,m} \cdot r^{-1/2-1/p}) = o(v_{f,m} \cdot r^{-1/2})$ for $m = 1, 2, 3$, in which $r^{-1/2}$ no longer depends on the dimension p. Thus, the resulting ANNs estimator can break the "curse of dimensionality"that typically arises in nonparametric kernel and linear sieve estimation, see Appendix A.1 for a detailed discussion.

3.4 Inverse Probability Weighted (IPW) Estimators

All three ANNs described in Section 3.3 can be applied to estimate the PS function $\pi_d^*(x)$, and the resulting TE estimators have the same asymptotic properties based on the three ANNs. For convenience of presentation, we use the ANNs given in Condition A to present the theoretical results in this section. To facilitate our subsequent statistical applications, we allow $r = r_n$ and $B = B_n$ to depend on sample size n. We denote the resulting ANN sieve space as

$$\mathcal{G}_n := \mathcal{G}(\psi, B_n, r_n, p).$$

Denote $D_{di} := \mathbb{1}(D_i = d)$ for brevity. Let $L(a) := \exp(a)/(1+\exp(a))$, for $a \in \mathbb{R}$, be the logistic function. The inverse logistic transform of the true PS is defined by

$$g_d^*(x) := L^{-1}\left(\pi_d^*(x)\right) = \log\left\{\pi_d^*(x)/(1-\pi_d^*(x))\right\},$$

and it satisfies $\mathbb{E}[\ell_d(D_{di}, \boldsymbol{X}_i; g_d^*)] \geq \mathbb{E}[\ell_d(D_{di}, \boldsymbol{X}_i; g_d)]$ for all $g_d \in \mathcal{G}_n$, where

$$\begin{aligned}\ell_d(D_{di}, \boldsymbol{X}_i; g_d) :=& D_{di} \log L\left(g_d(\boldsymbol{X}_i)\right) + \{1 - D_{di}\} \log\left(1 - L\left(g_d(\boldsymbol{X}_i)\right)\right) \\ =& D_{di} g_d(\boldsymbol{X}_i) - \log\left[1 + \exp(g_d(\boldsymbol{X}_i))\right].\end{aligned}$$

The directional derivative of $\ell_d(D_{di}, \boldsymbol{X}_i; g_d)$ is given by

$$\begin{aligned}\frac{d}{d\pi_d}\ell_d(D_{di}, \boldsymbol{X}_i; g_d)[u] :=& \lim_{t\to 0} \frac{\ell_d(D_{di}, \boldsymbol{X}_i; g_d + t\cdot u) - \ell_d(D_{di}, \boldsymbol{X}_i; g_d)}{t} \\ =& \left\{D_{di} - L(g_d(\boldsymbol{X}_i))\right\} u(\boldsymbol{X}_i)\end{aligned}$$

for $u \in L^2(dF_X)$. To estimate g_d^*, we consider the following ANN estimator in the space $\mathcal{G}_n$.

Assumption 2. *We assume the ANN estimator $\widehat{g}_d$ of g_d^* satisfies*

$$L_{d,n}(\widehat{g}_d) \geq \sup_{g_d \in \mathcal{G}_n} L_{d,n}(g_d) - O(\epsilon_n^2),$$

where

$$L_{d,n}(g_d) := \frac{1}{n}\sum_{i=1}^{n} \ell_d(D_{di}, \boldsymbol{X}_i; g_d(\cdot))$$

is the empirical criterion, and $\epsilon_n = o(n^{-1/2})$.

The ANN estimator of g_d^* depends on the sample size n. For notational simplicity, we write it as $\widehat{g}_d$. Assumption 2 states that the ANN estimator $\widehat{g}_d$ of g_d^* does not need to be the global maximizer of the objective function $L_{d,n}(g_d)$, which may not be obtained in practice. It can be any local solutions satisfying the condition given in Assumption 2, i.e., the values of the objective function evaluated at the local solutions and at the global maximizer cannot be far away from each other, and their difference needs to satisfy the order $O(\epsilon_n^2)$. This assumption is also imposed for sieve extreme estimation; see [105], [34] and [35]. The estimator of π_d^* is defined by $\widehat{\pi}_d := L(\widehat{g}_d)$, then we use the empirical version of (3.2) to construct the estimators of $\boldsymbol{\beta}^*$, denoted by $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, ..., \widehat{\beta}_J)^\top$ where

$$\widehat{\beta}_d := \arg\min_{\beta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)} \mathcal{L}\left(Y_i - \beta\right), \tag{3.8}$$

for every $d \in \mathcal{D} = \{0, 1, ..., J\}$.

3.4.1 Large Sample Properties

Assumption 3. *The dimension of $\boldsymbol{X}_i$ is denoted by $p \in \mathbb{N}$ and the number of hidden units is denoted by $r_n \in \mathbb{N}$. They are allowed to grow to infinity as the sample size n increases, with the rates*

$$p \leq C_0 \cdot (\log n)^{\frac{1}{2}} \text{ and } C_1 \cdot n^{\frac{p+1}{2(p+2)}} \leq r_n \leq C_2 \cdot n^{\frac{1}{2}-\nu},$$

where C_1 and C_2 are two positive constants, and

$$0 \leq C_0 < \sqrt{\frac{1}{4 \log M}}, \; C_3 \cdot \frac{\log\log n}{\log n} < \nu < \frac{1}{2(p+2)} \text{ and } \frac{3}{2} < C_3 < \infty.$$

The bound of the hidden-to-output weights, B_n, specified in (3.4) satisfies $B_n \leq 2v_{\pi_d^*, m}$.

Assumption 4. *For every $d \in \mathcal{D} = \{0, 1, ..., J\}$, $g_d^*(\cdot) \in \mathcal{W}^{m,\delta_0,\infty}(\mathcal{X})$, where $m \geq 1$ and $\delta_0 > 1$.*

Assumption 3 allows the dimension of covariates goes to infinity as the sample size grows, while it imposes restrictions on the growth rate of the dimension of covariates and that of the number of hidden units to ensure that the $L^2(dF_X)$-convergence rate of estimated PS attains $o_P(n^{-1/4})$, which is needed to establish the $\sqrt{n}$-asymptotic normality for the proposed TE estimator. For the ANNs given in (3.4), the coefficients γ_j no longer have the ℓ^1 constraint. Each of them only needs to be bounded. For the convenience of theoretical investigation, we set the bound to be 1. Moreover, $B = B_n$ depends on the sample size n. Unlike the LASSO method that penalizes a large number of regression coefficients yielding a sparse solution, the ANN estimation takes all explanatory variables into account, and it is possible that all hidden-to-output weights are nonzero. Assumption 4 is a smoothness condition imposed on the PS functions.

The following result establishes the convergence rates of g_d^* and π_d^*. The proof of Theorem 1 is presented in online Supplemental Materials [33].

Theorem 1. *Suppose Assumptions 2-4 hold. Then*

$$\|\widehat{g}_d - g_d^*\|_{L^2(dF_X)} = O_P\left(\max\left\{v_{\pi_d^*,m} \cdot r_n^{-\frac{1}{2}-\frac{1}{p}}, \sqrt{\frac{r_n \cdot p \cdot \log n}{n}}\right\}\right) = o_P\left(n^{-1/4}\right),$$

and

$$\|\widehat{\pi}_d - \pi_d^*\|_{L^2(dF_X)} = O_P\left(\max\left\{v_{\pi_d^*,m} \cdot r_n^{-\frac{1}{2}-\frac{1}{p}}, \sqrt{\frac{r_n \cdot p \cdot \log n}{n}}\right\}\right) = o_P\left(n^{-1/4}\right),$$

where the constants hiding inside O_p and o_p do not depend on p and n.

Theorem 1 shows that under a suitable smoothness condition, the M-estimates based on ANNs with a single hidden layer circumvent the curse of dimensionality and achieve a desirable rate for establishing the asymptotic normality of plug-in estimators [32]. [14] showed that their least squares estimator based on multilayer neural networks with a smooth activation function can achieve the convergence rate of $n^{-2s/(2s+d^*)}$ (up to a log factor), if the regression function satisfies a s-smooth generalized hierarchical interaction model of order d^*, where d^* is fixed. [102] established a similar rate for the ReLU activation function. However, the target function class considered in [14] and [102] is different from that used in our research. We refer to the online Supplemental Materials [33] for more discussion for the extension of our results for multilayer neural networks .

Assumption 5. *(i) Let Θ be a compact set of $\mathbb{R}^{J+1}$ containing the true parameters $\boldsymbol{\beta}^*$. (ii) The propensity scores are uniformly bounded away from zero and one, i.e., there exist a constant $\underline{c}$ such that $0 < \underline{c} \leq P(D_i = d|\boldsymbol{X}_i = x)$ for all $x \in \mathcal{X}$ and $d \in \{0, 1, ..., J\}$. (iii) The function $\mathcal{E}_d(\cdot, \beta_d^*) := \mathbb{E}[\mathcal{L}'(Y_i^*(d) - \beta_d^*)|\boldsymbol{X}_i = \cdot]$ is uniformly bounded and belongs to $\mathcal{W}^{m,\delta_0,\infty}(\mathcal{X})$ for every $d \in \{0, 1, ..., J\}$.*

Assumption 6. *(Approximation error) Let $\overline{g}(g_d, \epsilon_n) := (1 - \epsilon_n) \cdot g_d + \epsilon_n \cdot \{u^* + g_d^*\}$ be a local alternative value around $g_d \in \mathcal{G}_n$, where $u^* \in \mathcal{W}^{m,\delta_0,\infty}(\mathcal{X})$ denotes a perturbation. We assume the following conditions hold:*

$$\sup_{\{g_d \in \mathcal{G}_n : \|g_d - g_d^*\|_{L^2(dF_X)} \leq \delta_n\}} \|Proj_{\mathcal{G}_n} \overline{g}(g_d, \epsilon_n) - \overline{g}(g_d, \epsilon_n)\|_{L^2(dF_X)} = O\left(\frac{\epsilon_n^2}{\delta_n}\right),$$

$$\sup_{\{g_d \in \mathcal{G}_n : \|g_d - g_d^*\|_{L^2(dF_X)} \leq \delta_n\}} \frac{1}{n} \sum_{i=1}^{n} \left(\frac{d}{d\pi_d} \ell_d(D_{di}, \boldsymbol{X}_i; g_d^*)[\overline{g}(\boldsymbol{X}_i; g_d, \epsilon_n) - Proj_{\mathcal{G}_n} \overline{g}(\boldsymbol{X}_i; g_d, \epsilon_n)] \right) = O_P(\epsilon_n^2),$$

where $Proj_{\mathcal{G}_n} \overline{g}(g_d, \epsilon_n)$ denotes the $L^2(dF_X)$-projection of $\overline{g}(g_d, \epsilon_n)$ on the ANN space $\mathcal{G}_n$.

Assumption 7. *For every $d \in \mathcal{D} = \{0, 1, ..., J\}$,*

1. *the probability density of $Y^*(d)$, denoted by $f_{Y^*(d)}(y)$, is continuously differentiable.*

2. *the conditional probability density of Y_i given $\boldsymbol{X}_i = x$ and $D_i = d$, denoted by $f_{Y|X,D}(y|x, d)$, is continuously differentiable in $y \in \mathbb{R}$.*

3. *the following integrals are finite*

$$\sup_{\beta \in \Theta} \int \left| \mathcal{L}'(y - \beta) \frac{\partial}{\partial y} f_{Y^*(d)}(y) \right| dy < \infty \text{ and } \sup_{\beta \in \Theta} \int \left| \mathcal{L}'(y - \beta) \frac{\partial}{\partial y} f_{Y|D,X}(y|d, x) \right| dy < \infty,$$

for almost all $(d, x) \in \mathcal{D} \times \mathcal{X}$.

4. $H_d := -\partial_\beta \mathbb{E}[\mathcal{L}'(Y^*(d) - \beta_d^*)] > 0.$

Assumption 8.

1. *There exists a finite positive constant C such that for any $\beta \in \Theta$ and any $\delta > 0$ in a neighborhood of zero,*

$$\mathbb{E}\left[\sup_{\widetilde{\beta}: |\widetilde{\beta} - \beta| < \delta} \left\{ \mathcal{L}'(Y - \widetilde{\beta}) - \mathcal{L}'(Y - \beta) \right\}^2 \right] \leq const \times \delta.$$

2. $\mathbb{E}\left[\sup_{\beta \in \Theta} |\mathcal{L}'(Y - \beta)|^{2+\delta}\right] < \infty$ *for some $\delta > 0$.*

Assumption 5 (i) is a standard condition for the parameter space. Assumption 5 (ii) is a strict overlap condition ensuring the existence of participants at all treatment levels, which is commonly assumed in the literature. [42] discussed the applicability of the strict overlap condition with high-dimensional covariates, and provided a variety of circumstances under which this conditions hold. They also argued that the strict overlap condition may not be necessary if other smoothness conditions are imposed on the potential outcomes, or it can be technically relaxed with some non-standard asymptotic analyses (e.g., [64, 86]) and the sacrifice of uniform inference on ATE. Assumption 5 (iii) is a smoothness condition for approximation. The functions $\{\pi_d^*(x), \mathcal{E}_d(x, \beta_d^*)\}_{d=0}^J$ generally depend on the sample size n. Assumption 6 is also imposed in [105] and [34], which controls the approximation error. Assumption 7 imposes smoothness on data distributions which are needed for asymptotic analysis and ensuring finite asymptotic variance. Assumption 8 concerns L^2 continuity and envelope conditions, which are needed for the applicability of the uniform law of large numbers, establishing stochastic equicontinuity and weak convergence, see [91] and [6]. Again, they are satisfied by widely used loss functions such as $\mathcal{L}(v) = v^2$, $\mathcal{L}(v) = v\{\tau - \mathbb{1}(v \leq 0)\}$, and $\mathcal{L}(v) = v^2 \cdot |\tau - \mathbb{1}(v \leq 0)|$ discussed in Section 3.2.

The following theorem shows the asymptotic distribution of the proposed estimator $\widehat{\boldsymbol{\beta}}$, whose proof is presented in online Supplemental Materials [33].

Theorem 2. *Under Assumptions 1-8, for any $d \in \{0, 1, .., J\}$, we have $\widehat{\beta}_d \xrightarrow{p} \beta_d^*$ and*

$$\sqrt{n}(\widehat{\beta}_d - \beta_d^*) = H_d^{-1} \cdot \frac{1}{\sqrt{n}} \sum_{i=1}^n S_d(Y_i, D_{di}, \boldsymbol{X}_i; \beta_d^*) + o_P(1), \tag{3.9}$$

where $H_d = -\partial_\beta \mathbb{E}[\mathcal{L}'(Y^*(d) - \beta_d^*)]$ and

$$S_d = S_d(Y_i, D_{di}, \boldsymbol{X}_i; \beta_d^*) := \frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} \mathcal{L}'\{Y_i - \beta_d^*\} - \left\{ \frac{D_{di} - \pi_d^*(\boldsymbol{X}_i)}{\pi_d^*(\boldsymbol{X}_i)} \right\} \mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*).$$

Consequently,

$$\boldsymbol{V}^{-1/2} \cdot \sqrt{n} \left\{ \widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^* \right\} \xrightarrow{d} N\left(0, I_{(J+1)\times(J+1)}\right),$$

where $I_{(J+1)\times(J+1)}$ is the $(J+1) \times (J+1)$ identity matrix, $\boldsymbol{V} = \boldsymbol{H}^{-1}\mathbb{E}[\boldsymbol{S}\boldsymbol{S}^\top]\boldsymbol{H}^{-1}$, $\boldsymbol{H} = Diag\{H_0, ..., H_J\}$ and $\boldsymbol{S} = (S_0, ..., S_J)^\top$.

Based on the strict overlap condition and the integrability of the outcome, Assumption 5 (ii) and Assumption 8 (ii), we have that the asymptotic variance is finite, which implies the proposed estimator $\widehat{\boldsymbol{\beta}}$ is $\sqrt{n}$-consistency. In addition, when p is a fixed number, [3] derive the efficient influence function (EIF) for the TE parameter β_d^*, which is $\{-\partial_\beta \mathbb{E}[\mathcal{L}'(Y_i^*(d) - \beta_d^*)]\}^{-1} S_d(Y_i, D_{di}, \boldsymbol{X}_i; \beta_d^*)$, hence the proposed estimator attains the semiparametric efficiency bound.

The EIF can be applied to different loss functions. When EIF is given, the estimator of β_d^* can also be obtained by solving the estimated efficient score function [113]. For example, when the loss function $\mathcal{L}(v) = v^2$ corresponding to ATE, the EIF of $\beta_d^* = \mathbb{E}[Y^*(d)]$ is

$$\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} Y_i - \left\{ \frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1 \right\} \mathbb{E}[Y_i | D_{di} = 1, \boldsymbol{X}_i] - \beta_d^*. \tag{3.10}$$

It involves the PS function $\pi_d^*(x)$ and the OR function $\mathbb{E}[Y_i | D_{di} = 1, \boldsymbol{X}_i = x]$. [36], [93] and [46] estimate the two nuisance functions by state-of-the-art machine learning and deep learning algorithms with a sample splitting strategy to circumvent over-fitting.

When the loss function $\mathcal{L}(v) = v \cdot \{\tau - \mathbb{1}(v \leq 0)\}$ corresponding to the τ^{th}-quantile TE, the specific form of EIF for $\beta_d^* = F_{Y^*(d)}^{-1}(\tau)$ can also be derived from $H_d^{-1} S_d(Y_i, D_{di}, \boldsymbol{X}_i; \beta_d^*)$. As a result, its estimator can be obtained from solving the estimated efficient score equation

$$\sum_{i=1}^{n} \left[\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)} \{\tau - \mathbb{1}(Y_i \leq \beta)\} - \left\{ \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)} - 1 \right\} \left\{ \tau - \widehat{\mathbb{E}}[\mathbb{1}(Y_i \leq \beta) | D_{di} = 1, \boldsymbol{X}_i] \right\} \right] = 0, \tag{3.11}$$

where $\widehat{\pi}_d(x)$ and $\widehat{\mathbb{E}}[\mathbb{1}(Y_i \leq \beta) | D_{di} = 1, \boldsymbol{X}_i = x]$ are estimates of $\pi_d^*(x)$ and $\mathbb{E}[\mathbb{1}(Y_i \leq \beta) | D_{di} = 1, \boldsymbol{X}_i = x]$, respectively. [19] propose to use Lasso regression to estimate these two functions when they are assumed to have a generalized linear structure. The estimation of quantile TEs from (3.11) can be challenging when ANNs or other nonlinear machine learning methods are employed to approximate $\mathbb{E}[\mathbb{1}(Y_i \leq \beta)|D_{di} = 1, \boldsymbol{X}_i = x]$, as it involves the unknown parameter β.

Different from the aforementioned estimators constructed based on the estimated EIF, our TE estimators are directly obtained from optimizing an objective function which only involves the ANN-based estimated PS function. This approach greatly facilitates the computation of obtaining TE estimates and conducting causal inference. Computational convenience is of critical importance when we have large-scale observational data and ANNs approximation is involved.

3.4.2 Statistical Inference

This section presents two approaches to conduct statistical inference for $\boldsymbol{\beta}^*$ in practice. The first approach is based on the weighted bootstrap, the other one is based on the asymptotic distribution given in Theorem 2 with an estimated asymptotic variance.

Weighted Bootstrap

For every $d \in \{0, 1, ..., J\}$, let $\{w_{d1}, ..., w_{dn}\}$ be *i.i.d.* positive random weights that are independent of data satisfying $\mathbb{E}[w_{di}] = 1$ and $Var(w_{di}) = 1$. The weighted bootstrap estimator of the inverse logistic PS g_d^* is defined by satisfying

$$L_{d,n}^*(\widehat{g}_d^*) \geq \sup_{g_d \in \mathcal{G}_n} L_{d,n}^*(g_d) - O(\epsilon_n^2),$$

where $L^*_{d,n}(g_d) := n^{-1}\sum_{i=1}^n w_{di}\ell_d(D_{di}, \boldsymbol{X}_i; g_d(\cdot))$ is the empirical criterion, and $\epsilon_n = o(n^{-1/2})$. The weighted bootstrap estimator of β^*_d is defined by satisfying

$$\widehat{\beta}^*_d = \arg\min_{\beta\in\Theta} \frac{1}{n}\sum_{i=1}^n \frac{w_{di}D_{di}}{\widehat{\pi}^*_d(\boldsymbol{X}_i)}\mathcal{L}\left(Y_i - \beta\right),\ d \in \{0, 1, ..., J\},$$

where $\widehat{\pi}^*_d := L(\widehat{g}^*_d)$.

Let $\widehat{\boldsymbol{\beta}}^* := (\widehat{\beta}^*_0, ..., \widehat{\beta}^*_{J+1})^\top$. The following theorem justifies the validation of the proposed bootstrap inference.

Theorem 3. *Under Assumptions 1-8, for any $d \in \{0, 1, .., J\}$, then conditionally on the data we have*

$$\boldsymbol{V}^{-1/2} \cdot \sqrt{n}\left(\widehat{\boldsymbol{\beta}}^* - \widehat{\boldsymbol{\beta}}\right) \xrightarrow{d} \mathcal{N}(0, I_{(J+1)\times(J+1)}).$$

The proof of Theorem 3 is obtained by mimicking the proof of Theorem 1 and Theorem 2. The detailed proof can be found in the online Supplemental Materials [33].

Variance Estimation

This section studies the estimation of $\boldsymbol{V}$ in Theorem 2. Since the nonsmooth loss function may invalidate the exchangeability between the expectation and derivative operator, some care in the estimation of H_d is warranted. Using the tower property of conditional expectation and Leibniz integration rule (see Appendix A.3), we rewrite H_d as:

$$H_d = -\mathbb{E}\left[\frac{D_{di}}{\pi^*_d(\boldsymbol{X}_i)}\mathcal{L}'(Y_i - \beta^*_d)\frac{\partial}{\partial y}\log f_{Y,X|D}(Y_i, \boldsymbol{X}_i|d)\right]. \tag{3.12}$$

The log density $\log f_{Y,X|D}(y, x|d)$ can be estimated via the widely used ANN extremum estimator:

$$\widehat{f}_{Y,X|D}(y, x|d) := \frac{\exp\left(\widehat{a}_d(y, x)\right)}{\int_{\mathcal{Y}\times\mathcal{X}}\exp\left(\widehat{a}_d(y, x)\right)dydx},$$

where $\widehat{a}_d \in \widetilde{\mathcal{G}}_n$ approximately maximizes the following criterion function, which is similar to Assumption 2:

$$\widetilde{L}_{d,n}\left(\widehat{a}_d\right) \geq \sup_{a \in \widetilde{G}_n} \widetilde{L}_{d,n}\left(a\right) - O(\epsilon_n^2),$$

where

$$\widetilde{L}_{d,n}(a) := \frac{1}{n}\sum_{i=1}^{n} D_{di}\left[a(Y_i, \boldsymbol{X}_i) - \log \int_{\mathcal{Y}\times\mathcal{X}} \exp\left(a(y,x)\right) dydx\right],$$

and $\widetilde{\mathcal{G}}_n$ is a neural network similar to $\mathcal{G}_n$ with input variable (x,y). Then H_d can be estimated by

$$\widehat{H}_d := -\frac{1}{n}\sum_{i=1}^{n} \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}'(Y_i - \widehat{\beta}_d) \cdot \left\{\frac{\partial}{\partial y}\widehat{a}_d(Y_i, \boldsymbol{X}_i)\right\}. \tag{3.13}$$

Under Assumption 1, $\mathcal{E}_d(x, \beta_d^*) = \mathbb{E}[\mathcal{L}'(Y_i - \beta_d^*)|\boldsymbol{X}_i = x, D_i = d]$, hence $\mathcal{E}_d(x, \beta_d^*)$ can be estimated by using ANN extremum estimates:

$$\widehat{\mathcal{E}}_d(\cdot) =: \arg\min_{g(\cdot)\in\mathcal{G}_n} \frac{1}{n}\sum_{i=1}^{n} D_{di}\left\{\mathcal{L}'\left(Y_i - \widehat{\beta}_d\right) - g(\boldsymbol{X}_i)\right\}^2. \tag{3.14}$$

Therefore, the plug-in estimates of $S_d(Y_i, D_{di}, \boldsymbol{X}_i; \beta_d^*)$ is

$$\widehat{S}_{di} = \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}'\{Y_i - \widehat{\beta}_d\} - \left\{\frac{D_{di} - \widehat{\pi}_d(\boldsymbol{X}_i)}{\widehat{\pi}_d(\boldsymbol{X}_i)}\right\}\widehat{\mathcal{E}}_d(\boldsymbol{X}_i). \tag{3.15}$$

Finally, by (3.13) and (3.15), the asymptotic covariance matrix of the estimator is estimated by

$$\widehat{\boldsymbol{V}} := \widehat{\boldsymbol{H}}^{-1}\left\{\frac{1}{n}\sum_{i=1}^{n}\widehat{\boldsymbol{S}}_i\widehat{\boldsymbol{S}}_i^\top\right\}(\widehat{\boldsymbol{H}}^\top)^{-1}.$$

where $\widehat{\boldsymbol{H}} = \text{Diag}\{\widehat{H}_0, ..., \widehat{H}_J\}$ and $\widehat{\boldsymbol{S}}_i = (\widehat{S}_{0i}, ..., \widehat{S}_{Ji})^\top$. Since $|\widehat{\beta}_d - \beta_d^*| \xrightarrow{P} 0$, $\widehat{\pi}_d(\cdot) \xrightarrow{L^2(dF_X)} \pi_d^*(\cdot)$ and $\widehat{\mathcal{E}}_d(\cdot) \xrightarrow{L^2(dF_X)} \mathcal{E}_d(\cdot, \beta_d^*)$ for all $d \in \{0, 1, ..., J\}$. Based on these results, the consistency of $\widehat{\boldsymbol{V}}$, i.e. $\widehat{\boldsymbol{V}} \xrightarrow{p} \boldsymbol{V}$ follows from standard arguments.

The asymptotic normality of $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, \widehat{\beta}_1, ..., \widehat{\beta}_J)^\top$ established in Theorem 2 together with the consistency of $\widehat{\boldsymbol{V}}$ provides a theoretical support for conducting statistical inference of the TE parameter vector $\boldsymbol{\beta}^* = (\beta_0^*, \beta_1^*, ..., \beta_J^*)^\top$. For instance, based on these results, we can construct a $100(1-\alpha)\%$ confidence interval for each β_d^*, $d \in \{0, 1, ..., J\}$, given by

$$\left[\widehat{\beta}_d - n^{-1/2} z_{\alpha/2} \widehat{V}_{dd}^{1/2}, \quad \widehat{\beta}_d + n^{-1/2} z_{\alpha/2} \widehat{V}_{dd}^{1/2}\right],$$

where $\widehat{V}_{dd}$ is the (d,d)-element of the estimated covariance matrix $\widehat{\boldsymbol{V}}$, and $z_{\alpha/2}$ is the $100(1-\alpha/2)$ percentile of the standard normal. We can also construct confidence intervals for a contrast of $\boldsymbol{\beta}^*$ for a comparison of different TE parameters. That is, for any given $\boldsymbol{a} \in \mathbb{R}^{J+1}$, a $100(1-\alpha)\%$ confidence interval for $\boldsymbol{a}^\top \boldsymbol{\beta}^*$ is given by

$$\left[\boldsymbol{a}^\top \widehat{\boldsymbol{\beta}} - n^{-1/2} z_{\alpha/2} (\boldsymbol{a}^\top \widehat{\boldsymbol{V}} \boldsymbol{a})^{1/2}, \quad \boldsymbol{a}^\top \widehat{\boldsymbol{\beta}} + n^{-1/2} z_{\alpha/2} (\boldsymbol{a}^\top \widehat{\boldsymbol{V}} \boldsymbol{a})^{1/2}\right].$$

It is worth noting that estimation of the asymptotic variance is straightforward for average TE, but it can be difficult for other types of TEs, such as quantile TEs. Thus, the weighted bootstrap method is recommended for conducting inference of $\boldsymbol{\beta}^*$, and it can yield a more stable inferential result. Since our TE estimator is obtained by optimizing a generalized objective function, it is very convenient to apply the weighted bootstrap in our estimation procedure.

3.5 Outcome regression estimation

Using Assumption 1 and the property of conditional expectation, we can rewrite Equation (3.1) as follows:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \sum_{d=0}^{J} \mathbb{E}\left[\mathbb{E}[L(Y-\beta_d)|X, D=d]\right]. \tag{3.16}$$

Based on above expression, an alternative estimation strategy is to first estimate the conditional expectation $\mathbb{E}[L(Y-\beta_d)|X, D=d]$ (with β_d being fixed), then estimate $\boldsymbol{\beta}^*$ by minimizing the empirical version of (3.16) with estimated $\mathbb{E}[L(Y-\beta_d)|X, D=d]$. However, unlike the linear sieve estimation, there may not exist a closed form for ANN estimator of $\mathbb{E}[L(Y-\beta_d)|X, D=d]$, hence the outcome regression estimation for a general $L(\cdot)$ is difficult to obtain. In this section, we consider a particular but important parameter ATE which corresponds to $L(v)=v^2$. In this case, $\beta_d^* = \mathbb{E}[Y^*(d)] = \mathbb{E}[g_d^*(X)]$, where $g_d^*(X) := \mathbb{E}[Y|X, D=d]$ is the the outcome regression function. We can estimate $g_d^*(X)$ through neural networks:

$$\widehat{g}_d(\cdot) = \arg\min_{g(\cdot)\in\mathcal{G}_n} \frac{1}{2n}\sum_{i=1}^{n} D_{di}\left\{Y_i - g(X_i)\right\}^2.$$

Then the outcome regression (OR) estimator of β_d^* is defined to be

$$\widehat{\beta}_d^{OR} = \frac{1}{n}\sum_{i=1}^{n}\widehat{g}_d(X_i). \tag{3.17}$$

Assumption 9. *Let* $q(Y,\beta) := \{L'(Y-\beta) - L'(Y-\beta_d^*)\}^2$,

1. *there exist some finite positive constants* C *and* γ *such that for any* $\beta \in \Theta$ *and any* $\delta > 0$ *in a neighborhood of* 0, $\mathbb{E}\left[\sup_{\widetilde{\beta}:|\widetilde{\beta}-\beta|<\delta}\left\{q(Y,\widetilde{\beta}) - q(Y,\beta)\right\}^2\right] \le C\cdot\delta^{\gamma}$.

2. $\mathbb{E}\left[\sup_{\beta\in\Theta}|q(Y,\beta)|^{2+\delta}\right] < \infty$ *for some* $\delta > 0$.

Theorem 4. *Under Assumptions 1-9, for any $d \in \{0, 1, .., J\}$, we have that*

$$\sqrt{n}(\widehat{\beta}_d^{OR} - \beta_d^*) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \left[\frac{D_{di}}{\pi_d^*(X_i)} Y_i - \left\{ \frac{D_{di} - \pi_d^*(X_i)}{\pi_d^*(X_i)} \right\} \cdot g_d^*(X_i) - \mathbb{E}[g_d^*(X_i)] \right] + o_P(1).$$

Hence the estimator $\widehat{\boldsymbol{\beta}}^{OR}$ attains the semiprametric efficiency bound of $\boldsymbol{\beta}^*$ derived by [3].

The proof of Theorem 4 is presented in the online Supplemental Materials [33].

3.6 Simulation Studies

3.6.1 Background and Methods Used

In this section, we illustrate the finite sample performance of our proposed methods via simulations in which we generate data from models in Section 3.6.2. Our proposed IPW estimator can be applied to various types of treatment effects. We use ATE, ATT (average treatment effects on the treated), QTE and QTT (quantile treatment effects on the treated) for illustration of the performance of the IPW estimator. For QTE and QTT, we consider the 25th (Q1), 50th (Q2) and 75th (Q3) percentiles. We also illustrate the performance of the OR estimator for ATE and ATT. To obtain the IPW and OR estimators, we estimate the PS and OR functions by using our proposed ANN method as well as five other popular methods, including the generalized linear models (GLM), the generalized additive models (GAM), the random forests (RF), the gradient boosted machines (GBM) and the deep neural networks with three hidden layers (DNN). We make a comparison of the performance of the resulting TE estimators with the nuisance functions estimated by the aforementioned six methods. Moreover, we compare our IPW and OR estimators with the doubly robust (DR) estimator [46] and the Oracle estimator for ATE. For the DR estimator, the IPW and

OR functions are also approximated by ANNs. The Oracle estimator is constructed based on the efficient influence function with the true PS and OR functions plugged in, see [54]. The Oracle estimators are infeasible in practice, but they are expected to perform the best for estimation of ATE, and are served as a benchmark to compare with.

We use the Rectified Linear Unit (ReLU) as the activation function for both ANN and DNN. We use cubic regression spline basis functions for GAM. We apply grid search with 5-fold cross-validation to select hyperparameters for all methods, including the number of neurons for ANN DNN, the number of trees and max depths of trees for RF and GBM, and the learning rate for GBM. All the simulation studies are implemented in Python 3.9. The DNN, GLM, GAM, RF and GBM methods are implemented using the packages tensorflow, statsmodel, pyGAM and scikit-learn, respectively.

3.6.2 Date Generating Process

We generate the treatment and outcome variables from a nonlinear model and a linear model, respectively, given as follows.

Model 1 (nonlinear model) :

$$\begin{aligned}
\text{logit}\{\mathbb{E}(D_i|X_i)\} &= 0.5(X_{i1}^*X_{i2}^* - 0.7sin((X_{i3}^* + X_{i4}^*)(X_{i5}^* - 0.2)) - 0.1),\\
\mathbb{E}(Y_i(1)|X_i) = \mathbb{E}(Y_i|X_i, D_i = 1) &= 0.3(X_{i1}^* - 0.9)^2 + 0.1(X_{i2}^* - 0.5)^2\\
&\quad - 0.6X_{i2}^*X_{i3}^* + sin(-1.7(X_{i1}^* + X_{i3}^* - 1.1) + X_{i4}^*X_{i5}^*) + 1,\\
\mathbb{E}(Y_i(0)|X_i) = \mathbb{E}(Y_i|X_i, D_i = 0) &= 0.64(X_{i1}^* - 0.9)^2 + 0.16(X_{i2}^* + 0.2)^2\\
&\quad - 0.6X_{i2}^*X_{i3}^* + sin(-1.7(X_{i1}^* + X_{i3}^* - 1.1) + X_{i4}^*X_{i5}^*) - 1;
\end{aligned}$$

Model 2 (linear model) :

$$\text{logit}\{\mathbb{E}(D_i|X_i)\} = 0.1(X_{i1}^* + X_{i2}^* - 2X_{i3}^* + 3X_{i4}^* - 3X_{i5}^*),$$

$$\mathbb{E}(Y_i(1)|X_i) = \mathbb{E}(Y_i|X_i, D_i = 1) = 4X_{i1}^* + 3X_{i2}^* - X_{i3}^* - 5X_{i4}^* + 7X_{i5}^* + 1,$$

$$\mathbb{E}(Y_i(0)|X_i) = \mathbb{E}(Y_i|X_i, D_i = 0) = 4X_{i1}^* + 3X_{i2}^* - X_{i3}^* - 5X_{i4}^* + 7X_{i5}^* - 1,$$

where $X_{ij'}^* = \frac{5}{p}\sum_{j=p(j'-1)/5+1}^{pj'/5} X_{ij}$ for $1 \leq j' \leq 5, 1 \leq i \leq n$, and $Y_i(d) = \mathbb{E}(Y_i(d) \mid X_i) + \epsilon_i$, $d = \{0, 1\}$, $\epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, 1)$ for $1 \leq i \leq n$.

The confounders X_{ij} are generated from $X_{ij} = 2(F(Z_{ij}) - 0.5)$, where $Z_i = (Z_{i1}, ..., Z_{ip})^\top \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{\Sigma})$, $\mathbf{\Sigma} = \{\sigma_{kk'}\}$, $\sigma_{kk'} = 0.2^{|k-k'|}$ for $1 \leq k, k' \leq p$, and $F(\cdot)$ is the cumulative distribution function of standard normal distribution for $1 \leq i \leq n, 1 \leq j \leq p$. We partition the confounders into 5 subgroups, and $X_{ij'}^*$ is the average of the $p/5$ confounders in the j'-th subgroup for $j' = 1, ..., 5$, so that every confounder is included in our models. We consider $p = 5$, 10 and $n = 1000$, 2000, 5000. All simulation results are based on 400 realizations.

We also consider a simulation setting with $p = 100$ and $n = 2000$ for the nonlinear model (Model 1) to illustrate the performance of our proposed methods. The confounders are generated in the same way as described above except that $Z_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{\Sigma})$, $\mathbf{\Sigma} = \{\sigma_{kk'}\}$, $\sigma_{kk'} = 0.2^{m(|k-k'|)}$ for $1 \leq k, k' \leq 100$, where $m(x) = x - 5\lfloor x/5 \rfloor$, and $\lfloor a \rfloor$ denotes the largest integer no greater than a.

3.6.3 Treatment Effect Estimators

Treatment effects are estimated in two steps. First, we obtain the estimates of PS function and OR function from optimizing these two functions, $d \in \{0,1\}$:

$$\widehat{\pi}_d(\cdot) := \arg \max_{\pi_d(\cdot)\in\mathcal{G}_n} \frac{1}{n}\sum_{i=1}^{n} \ell_d(D_{di}, X_i; \pi_d(\cdot)),$$
$$\widehat{g}_d(\cdot) = \arg \min_{g(\cdot)\in\mathcal{G}_n} \frac{1}{2n}\sum_{i=1}^{n} D_{di}\left\{Y_i - g(X_i)\right\}^2.$$

Set the loss function $L(v) = v^2$, we can obtain the IPW and OR estimators of treatment effect:

$$\widehat{\beta}_d^{IPW} = \frac{1}{n}\sum_{i=1}^{n} \frac{D_{di}Y_i/\widehat{\pi}_d(X_i)}{\frac{1}{n}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(X_i)}}, \tag{3.18}$$

$$\widehat{\beta}_d^{OR} = \frac{1}{n}\sum_{i=1}^{n} \widehat{g}_d(X_i). \tag{3.19}$$

The IPW and OR estimators of treatment effect for treatment group $d' = 1$ are:

$$\widehat{\beta}_{d,d'}^{IPW} = \frac{1}{n}\sum_{i=1}^{n} \frac{D_{di}\widehat{\pi}_{d'}(X_i)Y_i/\widehat{\pi}_d(X_i)}{\frac{1}{n}\sum_{i=1}^{n} D_{di}\widehat{\pi}_{d'}(X_i)/\widehat{\pi}_d(X_i)}, \tag{3.20}$$

$$\widehat{\beta}_{d,d'}^{OR} = \frac{1}{n}\sum_{i=1}^{n} \frac{D_{d'i}}{\frac{1}{n}\sum_{i=1}^{n} D_{d'i}}\widehat{g}_d(X_i). \tag{3.21}$$

Next we obtain the ATE and ATT, denoted by η^{IPW} and η^{OR} for IPW and OR estimators of ATE, and $\eta_{treated}^{IPW}$ and $\eta_{treated}^{OR}$ for IPW and OR estimators of ATT, respectively:

$$\widehat{\eta}^{IPW} = \widehat{\beta}_1^{IPW} - \widehat{\beta}_0^{IPW}, \tag{3.22}$$

$$\widehat{\eta}^{OR} = \widehat{\beta}_1^{OR} - \widehat{\beta}_0^{OR}, \tag{3.23}$$

$$\widehat{\eta}_{treated}^{IPW} = \widehat{\beta}_{1,1}^{IPW} - \widehat{\beta}_{0,1}^{IPW}, \tag{3.24}$$

$$\widehat{\eta}_{treated}^{OR} = \widehat{\beta}_{1,1}^{OR} - \widehat{\beta}_{0,1}^{OR}. \tag{3.25}$$

Set the loss function $L(v) = \rho_\tau(Y_i - q) = v \cdot \{\tau - I(v \leq 0)\}$ for some $\tau \in (0,1)$, we can obtain the treatment effect from the minimization problem, $d \in \{0,1\}$:

$$\widehat{\beta}_d^\tau = \arg\min_q \frac{1}{n} \sum_{i=1}^{n} \frac{D_{di}}{\widehat{\pi}_d(X_i))} \rho_\tau(Y_i - q). \tag{3.26}$$

We solve this minimization problem using linear programming. Similarly the treatment effect for the treatment group $d' = 1$ is obtained from the minimization problem:

$$\widehat{\beta}_{d,d'}^\tau = \arg\min_q \frac{1}{n} \sum_{i=1}^{n} \frac{D_{di}}{\widehat{\pi}_d(X_i))} \frac{\widehat{\pi}_{d'}(X_i)}{\frac{1}{n}\sum_{i=1}^{n} D_{d'i}} \rho_\tau(Y_i - q). \tag{3.27}$$

For each $\tau \in (0,1)$, we obtain the QTE and QTT, denoted by η^τ and $\eta_{treated}^\tau$ for QTE and QTT, respectively:

$$\widehat{\eta}^\tau = \widehat{\beta}_1^\tau - \widehat{\beta}_0^\tau, \tag{3.28}$$

$$\widehat{\eta}_{treated}^\tau = \widehat{\beta}_{1,1}^\tau - \widehat{\beta}_{0,1}^\tau. \tag{3.29}$$

3.6.4 Estimation Algorithms

Let $\gamma_0 = (\gamma_{01}, ..., \gamma_{0p})^\top$, $a_j = (a_{j1}, ..., a_{jp})^\top$, $\boldsymbol{\alpha} = (a_{10}, a_{11}, ..., a_{jk}, ...a_{r_n p})$, $\boldsymbol{\gamma} = (\gamma_{00}, \gamma_{01}, ..., \gamma_{0p}, \gamma_1, ..., \gamma_{r_n})$, and

$$f(x; \boldsymbol{\alpha}, \boldsymbol{\gamma}) = \gamma_0^\top x + \gamma_{00} + \sum_{j=1}^{r_n} \gamma_j \psi(a_j^\top x + a_{j0}).$$

The ANN estimators of PS and OR functions can be obtained through solving the following optimization problem:

$$(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\gamma}}) := \arg\min_{\boldsymbol{\alpha},\boldsymbol{\gamma}} L(\boldsymbol{\alpha}, \boldsymbol{\gamma}; \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{D_d}),$$

where

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\gamma}; \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{D_d}) &= -\frac{1}{n}\sum_{i=1}^{n} \Big[D_{di} \log(\frac{\exp(f(X_i; \boldsymbol{\alpha}, \boldsymbol{\gamma}))}{1+\exp(f(X_i; \boldsymbol{\alpha}, \boldsymbol{\gamma}))}) \\ &\quad + (1-D_{di}) \log(\frac{1}{1+\exp(f(X_i; \boldsymbol{\alpha}, \boldsymbol{\gamma}))}) \Big], \\ L(\boldsymbol{\alpha}, \boldsymbol{\gamma}; \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{D_d}) &= \frac{1}{2n}\sum_{i=1}^{n} D_{di} \left\{ Y_i - f(X_i; \boldsymbol{\alpha}, \boldsymbol{\gamma}) \right\}^2, \end{aligned}$$

for PS and OR functions, respectively, and $\boldsymbol{X} = (X_1^\top, ..., X_n^\top)^\top$, $\boldsymbol{Y} = (Y_1, ..., Y_n)^\top$, $\boldsymbol{D_d} = (D_{d1}, ..., D_{dn})^\top$.

These two optimization problems are solved using ADAM method ([71]) with modification, see Algorithm 2. We apply the ADAM method to update $\boldsymbol{\alpha}$. For the update in $\boldsymbol{\gamma}$, we solve for generalized linear models with covariates being $\tilde{X}_i = (1, X_i^\top, \psi(a_1^\top X_i + a_{10}), ..., \psi(a_j^\top X_i + a_{j0}), ..., \psi(a_{r_n}^\top X_i + a_{r_n 0}))^\top$, and outcomes being D_{di} and Y_i for PS and OR functions, respectively. The number of hidden units is selected through 5-fold cross validation. The starting values are generated using HE initialization method ([59]).

3.6.5 Variance Estimation

The variance of our estimators are estimated using the asymptotic estimators in Section 3.4.2 as well as using weighted bootstrapping method. We use variance estimators in [54] for ATE and ATT, and variance estimators in [47] for QTE and QTT.

Algorithm 2

```
Require: $\boldsymbol{\alpha}_{ini}$: Initial value for $\boldsymbol{\alpha}$
Require: $\epsilon_0$: converge criterion
Require: $\tilde{\alpha}$: step size with default value of 0.001
Require: $\tilde{\beta}_1$: decay rate with default value of 0.9
Require: $\tilde{\beta}_2$: decay rate with default value of 0.999
Require: $\tilde{\epsilon}$: stabilizer with default value of $10^{-8}$
  $t \leftarrow 0$ (Initialize timestep)
  $m_0 \leftarrow 0$
  $v_0 \leftarrow 0$
  $\boldsymbol{\alpha}_0 \leftarrow \boldsymbol{\alpha}_{ini}$
  while $\|\boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t\| < \epsilon_0$ do
      $t \leftarrow t + 1$
      $\boldsymbol{\gamma}_t \leftarrow \arg\min_\gamma L(\boldsymbol{\alpha}, \boldsymbol{\gamma}; \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{D_d})$ (Update $\boldsymbol{\gamma}$ from GLM estimates)
      $g_t \leftarrow \nabla_\alpha L(\boldsymbol{\alpha}, \boldsymbol{\gamma}; \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{D_d})$ (Obtain gradient for $\boldsymbol{\alpha}$ at timestep t)
      $m_t \leftarrow \tilde{\beta}_1 m_{t-1} + (1 - \tilde{\beta}_1) g_t$
      $v_t \leftarrow \tilde{\beta}_2 v_{t-1} + (1 - \tilde{\beta}_2) g_t \odot g_t$
      $\widehat{m}_t \leftarrow m_t / (1 - \tilde{\beta}_1^t)$
      $\widehat{v}_t \leftarrow v_t / (1 - \tilde{\beta}_2^t)$
      $\Delta(\boldsymbol{\alpha}_t) \leftarrow \widehat{m}_t / (\sqrt{\widehat{v}_t} + \tilde{\epsilon})$
      $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} - \tilde{\alpha}\Delta(\boldsymbol{\alpha}_t)$ (Update $\boldsymbol{\alpha}$ at timestep t)
  end while
```

The variance estimator for ATE and ATT are:

$$\widehat{\boldsymbol{V}} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{D_{1i}(Y_i - \widehat{g_1}(X_i)}{\widehat{\pi}_1(X_i)} - \frac{D_{0i}(Y_i - \widehat{g_0}(X_i)}{\widehat{\pi}_0(X_i)} + \widehat{g_1}(X_i) - \widehat{g_0}(X_i) - \widehat{\eta}\right]^2,$$

$$\widehat{\boldsymbol{V}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\widehat{\pi}_1(X_i)}{n^{-1}\sum_{i=1}^{n}\widehat{\pi}_1(X_i)}\left[\frac{D_{1i}(Y_i - \widehat{g_1}(X_i)}{\widehat{\pi}_1(X_i)} - \frac{D_{0i}(Y_i - \widehat{g_0}(X_i)}{\widehat{\pi}_0(X_i)} + \widehat{g_1}(X_i) - \widehat{g_0}(X_i) - \widehat{\eta}_{treated}\right]\right\}^2,$$

where

$$\begin{aligned}
\widehat{\eta} &= \frac{1}{n}\sum_{i=1}^{n}\left[\frac{D_{1i}(Y_i - \widehat{g_1}(X_i)}{\widehat{\pi}_1(X_i)} - \frac{D_{0i}(Y_i - \widehat{g_0}(X_i)}{\widehat{\pi}_0(X_i)} + \widehat{g_1}(X_i) - \widehat{g_0}(X_i)\right],\\
\widehat{\eta}_{treated} &= \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\widehat{\pi}_1(X_i)}{n^{-1}\sum_{i=1}^{n}\widehat{\pi}_1(X_i)}\left[\frac{D_{1i}(Y_i - \widehat{g_1}(X_i)}{\widehat{\pi}_1(X_i)} - \frac{D_{0i}(Y_i - \widehat{g_0}(X_i)}{\widehat{\pi}_0(X_i)} + \widehat{g_1}(X_i) - \widehat{g_0}(X_i)\right]\right\}.
\end{aligned}$$

The variance estimator of QTE for $\tau \in (0,1)$ is $\widehat{\boldsymbol{V_\tau}} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{\boldsymbol{\phi}}_{\tau,i} + \widehat{\boldsymbol{\alpha}}_{\tau,i})^2$, where

$$\begin{aligned}
\widehat{\boldsymbol{\phi}}_{\tau,i} &= \frac{D_{1i}\widehat{g}_{1,\tau}(X_i)}{\widehat{\pi}_1(X_i)} - \frac{D_{0i}\widehat{g}_{0,\tau}(X_i)}{\widehat{\pi}_0(X_i)},\\
\widehat{\boldsymbol{\alpha}}_{\tau,i} &= -\widehat{\mathbb{E}}\left[\frac{D_1\widehat{g}_{1,\tau}(X)}{\widehat{\pi}_1(X)} - \frac{D_0\widehat{g}_{0,\tau}(X)}{\widehat{\pi}_0(X)}\middle| X = X_i\right]\cdot(D_i - \widehat{\pi}_1(X_i)),\\
\widehat{g}_{d,\tau}(X_i) &= -\frac{\mathbf{1}\{Y \le \widehat{q_d^\tau}\} - \tau}{\widehat{f}_d(\widehat{q_d^\tau})}, (d = 0,1).
\end{aligned}$$

Here, $\widehat{f_d}$ is the estimator of the density of the potential outcome $Y(d)$:

$$\widehat{f_d}(y) = \frac{1}{n}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(X_i)}\mathcal{K}_{h_h,d}(Y_i - y)$$

where $\mathcal{K}_{h_d,d}(\cdot) = h_d^{-1}\mathcal{K}_d(\cdot/h_d^{-1})$, $\mathcal{K}_d(\cdot)$ is a kernel function and h_d^{-1} is a bandwidth for $d = 0,1$. Here we choose the Gaussian kernel, and bandwidth is determined by rule of thumb.

Next, $\widehat{\boldsymbol{\alpha}}_{\tau,i}$ involves estimating an unknown conditional expectation, we propose estimating it using ANN for our proposed estimator, and we estimate this unknown expectation using corresponding methods for other estimators (e.g. use GLM to estimate this expectation for GLM estimator).

Similarly, the variance estimator of QTT is $\widehat{\boldsymbol{V_\tau}} = \frac{1}{n}\sum_{i=1}^n(\widehat{\boldsymbol{\phi}}_{\tau,i} + \widehat{\boldsymbol{\alpha}}_{\tau,i})^2$, where

$$\begin{aligned}
\widehat{\boldsymbol{\phi}}_{\tau,i} &= \frac{D_{1i}\widehat{g}_{1,\tau}(X_i)}{\widehat{\pi}_1(X_i)} - \frac{D_{0i}\widehat{g}_{0,\tau}(X_i)}{\widehat{\pi}_0(X_i)},\\
\widehat{\boldsymbol{\alpha}}_{\tau,i} &= -\widehat{\mathbb{E}}\left[\frac{D_1\widehat{g}_{1,\tau}(X)}{\widehat{\pi}_1(X)} - \frac{D_0\widehat{g}_{0,\tau}(X)}{\widehat{\pi}_0(X)}\middle| X = X_i\right]\cdot(D_i - \widehat{\pi}_1(X_i)),\\
\widehat{g}_{d,\tau}(X_i) &= -\frac{\mathbf{1}\{Y \le \widehat{q}^{\tau}_{d,d'}\} - \tau}{\widehat{f}_{d,d'}(\widehat{q}^{\tau}_{d,d'})}, (d = 0, 1, d' = 1),
\end{aligned}$$

where

$$\widehat{f}_{d,d'}(y) = \frac{1}{n}\sum_{i=1}^n \frac{D_{di}}{\widehat{\pi}_d(X_i)}\cdot\frac{\widehat{\pi}'_d(X_i)}{n^{-1}\sum_{i=1}^n\widehat{\pi}'_d(X_i)}\mathcal{K}_{h_h,d}(Y_i - y), (d = 0, 1, d' = 1).$$

For the weighted bootstrapping method, we sample $W_i, i = 1, ..., n$ as n i.i.d. positive random weights, satisfying $E(W_i) = 1$ and $Var(W_i) = v_0 < \infty$ and independent of $(\widehat{\beta}_d, \widehat{\pi}_d(\cdot))$. Here we generate W_i from the exponential distribution with mean parameter being 1.

The weighted IPW estimator of treatment effect can be estimated in two steps. First we will obtain weighted estimates of propensity score function from maximizing the weighted log-likelihood, $d \in \{0, 1\}$:

$$\widehat{\pi}_d(\cdot) = \arg\max_{\pi_d(\cdot)\in\mathcal{G}_n}\frac{1}{n}\sum_{i=1}^n W_i\ell_d(D_{di}, X_i; \pi_d(\cdot)). \tag{3.30}$$

Next the weighted estimator of $\widehat{\beta'}_d$ is obtained from

$$\widehat{\beta'}_d^{IPW} = \frac{1}{n}\sum_{i=1}^n\frac{W_iD_{di}Y_i/\widehat{\pi}_d(X_i)}{\frac{1}{n}\sum_{i=1}^n\frac{W_iD_{di}}{\widehat{\pi}_d(X_i)}}. \tag{3.31}$$

The weighted OR estimator of treatment effect can also be obtained from similar procedures. First we obtain weighted estimates of outcome function from minimizing the weighted loss function , $d \in \{0,1\}$, then we obtain the weighted estimator of $\widehat{\beta'}_d$:

$$\widehat{g}_d(\cdot) = \arg \min_{g(\cdot)\in\mathcal{G}_n} \frac{1}{2n}\sum_{i=1}^{n} W_i D_{di} \left\{Y_i - g(X_i)\right\}^2 . \tag{3.32}$$

$$\widehat{\beta'}_d^{OR} = \frac{1}{n}\sum_{i=1}^{n} \frac{W_i \widehat{g}_d(X_i)}{\frac{1}{n}\sum_{i=1}^{n} W_i}. \tag{3.33}$$

The weighted quantile estimator uses the weighted estimates of propensity score and is obtained from minimizing the weighted function:

$$\widehat{\beta'}_d^{\tau} = \arg \min_{q} \frac{1}{n}\sum_{i=1}^{n} W_i \frac{D_{di}}{\widehat{\pi}_d(X_i))}\rho_\tau(Y_i - q). \tag{3.34}$$

Similarly the treatment effects on the treated are

$$\widehat{\beta'}_{d,d'}^{IPW} = \frac{1}{n}\sum_{i=1}^{n} \frac{W_i D_{di}\widehat{\pi}_{d'}(X_i)Y_i/\widehat{\pi}_d(X_i)}{\frac{1}{n}\sum_{i=1}^{n} W_i D_{di}\widehat{\pi}_{d'}(X_i)/\widehat{\pi}_d(X_i)}, \tag{3.35}$$

$$\widehat{\beta'}_{d,d'}^{OR} = \frac{1}{n}\sum_{i=1}^{n} \frac{W_i D_{d'i}}{\frac{1}{n}\sum_{i=1}^{n} W_i D_{d'i}}\widehat{g}_d(X_i), \tag{3.36}$$

$$\widehat{\beta'}_{d,d'}^{\tau} = \arg \min_{q} \frac{1}{n}\sum_{i=1}^{n} W_i \frac{D_{di}\widehat{\pi}_{d'}(X_i)}{\widehat{\pi}_d(X_i)}\rho_\tau(Y_i - q). \tag{3.37}$$

Finally, we obtain the weighted estimators of ATE and ATT and the weighted estimators of QTE and QTT:

$$\widehat{\eta'}^{IPW} = \widehat{\beta'}_1^{IPW} - \widehat{\beta'}_0^{IPW}, \tag{3.38}$$

$$\widehat{\eta'}^{OR} = \widehat{\beta'}_1^{OR} - \widehat{\beta'}_0^{OR}, \tag{3.39}$$

$$\widehat{\eta'}_{treated}^{IPW} = \widehat{\beta'}_{1,1}^{IPW} - \widehat{\beta'}_{0,1}^{IPW}, \tag{3.40}$$

$$\widehat{\eta'}_{treated}^{OR} = \widehat{\beta'}_{1,1}^{OR} - \widehat{\beta'}_{0,1}^{OR}, \tag{3.41}$$

$$\widehat{\eta'}^{\tau} = \widehat{\beta'}_1^{\tau} - \widehat{\beta'}_0^{\tau}, \tag{3.42}$$

$$\widehat{\eta'}_{treated}^{\tau} = \widehat{\beta'}_{1,1}^{\tau} - \widehat{\beta'}_{0,1}^{\tau}. \tag{3.43}$$

For our simulation study, each time we generate a vector of positive random weights $W = (W_1, ..., W_n)^\top$ and obtain corresponding weighted estimators of all types of TE. With the procedure repeated N_b times, we can estimate the variance of all types of TE from the bootstrapping estimators. Here we set N_b as 400.

3.6.6 Simulation Results

To compare the performance of different methods for estimating the TEs, we report the following statistics: the absolute value of bias (bias), the empirical standard deviation (emp_sd), the average value of the estimated standard deviations based on the asymptotic formula (est_sd) and obtained from the weighted bootstrapping (est_sd_boot), and the empirical coverage rates of the 95% confidence intervals based on the estimated asymptotic standard deviations (cover_rate) and the weighted bootstrapping method (cover_rate_boot). The 95% confidence intervals based on bootstrapping are obtained from the 2.5th percentile and 97.5th percentile of the weighted bootstrapping estimates.

Tables 3.1 - 3.2 report the numerical results for different estimators of ATE for Model 1 with $p = 5$, 10, respectively. We see that as n increases, the empirical coverage rates (cover_rate and cover_rate_boot) based on our proposed ANN-based IPW and OR estimates become closer to the nominal level 95%. The biases are close to zero, and the values of emp_sd, est_sd and est_sd_boot decrease as n increases. These results corroborate our asymptotic theories. We observe that our ANN-based IPW and OR estimators have comparable performance to the DR and the Oracle estimators when estimating ATE. The proposed ANN-based OR estimator slightly outperforms the ANN-based IPW and DR estimators in the sense that it has the smallest emp_sd value. It is possible that the estimated

PS functions have a few values close to zero. This can affect the emp_sd value of the IPW estimate for ATE. The DR estimator which is constructed based on the estimates of both IPW and OR functions has larger emp_sd values than the OR estimator, but it yields smaller emp_sd values than the IPW estimator. Our numerical results suggest that the proposed ANN-based OR estimator is preferred for the estimation of ATE. However, in practice it can be difficult to construct OR and DR estimators for other types of TEs, such as quantile TEs. Then the proposed ANN-based IPW estimator becomes a more appealing tool. Moreover, our numerical results given in Tables 3.3 - 3.4 show that the performance of the ANN-based IPW estimators for quantile TEs is less influenced by the small values of the estimated PS functions because of the robustness of the quantile objective functions. For our proposed ANN-based IPW and OR estimators, it is convenient to apply the proposed weighted bootstrap procedure for conducting inference. We find that the empirical coverage rates of 95% confidence intervals obtained from the weighted bootstrapping are closer to the nominal level than those obtained from the estimated asymptotic standard deviations.

Next, we compare the performance of different machine learning (ML) methods for estimation of ATE. We see that the GLM and GAM methods yield large estimation biases due to the model misspecification problem. Our numerical results show that the proposed ANN method outperforms the other two ML methods, RF and GBM, for estimation of TEs. The empirical coverage rates based on the ANN method are closer to the nominal level in all cases than the rates obtained from RF and GBM. It is worth noting that our ANN-based TE estimators enjoy the properties of root-n consistency and semiparametric efficiency. In general, our numerical results corroborate those theoretical properties. Moreover, for

RF and GBM, the OR estimator also performs better than the IPW estimator for ATE estimation. The empirical coverage rates of the 95% confidence intervals obtained from the weighted bootstrapping are improved compared to the rates obtained from the estimated asymptotic standard deviation. The DNN method has comparable performance to the ANN method.

In contrast, the GLM and GAM estimates have much smaller coverage rates because of the large biases. This implies that when the PS and OR functions are nonlinear, the estimates from GLM and GAM can be very biased and inefficient due to the model mis-specification. The RF and GBM estimates perform reasonably well for OR estimator. However, both methods don't correctly estimate the PS function, resulting in large bias in IPW estimator. We also observe that the estimated asymptotic standard deviations of the RF and GBM estimates are quite small, resulting in very small coverage rates (cover_rate), however the standard deviations estimated from bootstrapping are more reasonable and the coverage rates (cover_rate_boot) are closer to nominal 95%. The DNN estimate is comparable to ANN estimate for both the IPW and OR estimators. The DR estimate performs as good as ANN estimate of the OR estimator for most cases, and slightly worse than the latter in cases with small n, this is because DR estimate is an combination of IPW and OR estimator, and its variance is inflated by IPW estimator. In summary, the ANN estimates of the IPW and OR estimators perform well in estimating ATE, and the ANN estimate of OR estimator outperforms other methods, especially when n is 1000. Our proposed method outperforms other machine learning methods because our method has reliable statistical asymptotic theory support. Although the OR estimator has superior performance compared to that of IPW estimator, we can't apply OR estimator for other type of TE such as QTE. It is also difficult to construct an influence function based estimate of QTE, thus we can't obtain DR and Oracle estimates of QTE. However, our proposed method can still be applied to estimating QTE from the estimated PS function, and such estimation is robust, QTE would be resistant to the instability of weights.

Tables 3.3 - 3.4 show the numerical results of different methods for estimation of QTEs for Model 1 with $p = 5$, 10, respectively. It is difficult to construct OR and DR estimators for QTEs, so we only report the results for the IPW estimators, which are very convenient to be obtained in this context. The PS functions are estimated by different ML methods, and the numerical results of the resulting IPW estimates are summarized in Tables 3.3 - 3.4. In general, we observe similar patterns of numerical performance of different methods as shown in Table 3.1 - 3.2. It is worth noting that the proposed ANN-based IPW method has very stable performance for estimation of QTEs. The resulting emp_sd values are not influenced by possibly small values of the estimated PS functions because of the robustness nature of the quantile objective function. Moreover, in the QTE settings, estimation of the asymptotic standard deviations can involve a complicated procedure, and several approximations are needed. As a result, the estimation is not guaranteed to perform well. Figure 3.1 shows the boxplots of the estimated asymptotic standard deviations of QTE (Q1) for Model 1 with $p = 5, 10$, $n = 1000$. We see that the estimated values are large for some simulation replicates. In contrast, the estimated standard deviations obtained from the weighed bootstrapping have more reliable performance. In complex TE settings such as QTEs, the proposed weighted bootstrap method that avoids the estimation of the asymptotic variance provides a robust way to conduct statistical inference, and thus it is recommended in practice. It is convenient to apply the weighted bootstrap method in our proposed TE estimation procedure, as the TE estimators are obtained from optimizing a general objective function. We also apply different ML methods to estimate the PS function. The numerical results show that the ANN and DNN methods have comparable performance, and they still outperform other methods for estimation and inference of QTEs in all simulation settings.

The ANN and DNN estimates outperform other estimates, the bias are close to zero, and coverage rates (cover_rate and cover_rate_boot) are closer to the nominal 95% as n increases. We also observe that the est_sd values are quite large when sample when n is 1000, while the est_sd_boot values are comparable with the emp_sd values, indicating bootstrapping provides a reliable estimate of standard deviations for QTE. In summary, ANN estimate performs well in estimating QTE.

Estimating standard deviation of QTE from asymptotic formula is more challenging than that of ATE, the former requires correct estimation of marginal density of the potential outcomes and an unknown conditional expectation term which also relies on correct estimation of PS function. Figure 3.1 shows the boxplot of the standard deviation estimator of QTE of Q1 for Model 1 with $p = 10$, $n = 1000$. There are some outliers that are quite large, resulting in an inaccurate estimate of the variance. In summary, bootstrapping provides a robust way to estimate the standard deviation, especially when n is small. However, we can't apply bootstrapping for DR estimate which shows another limitation of this method.

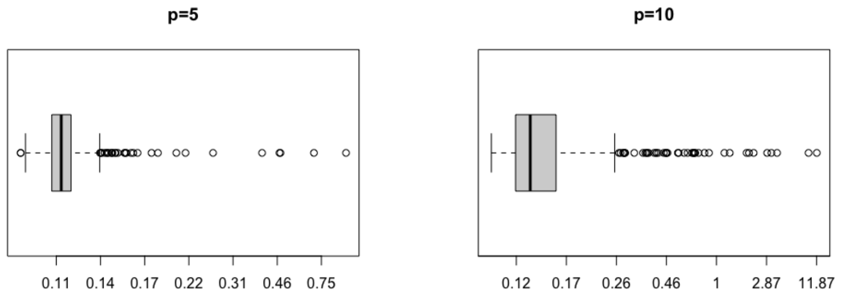


Figure 3.1: Boxplots of the estimated asymptotic standard deviations of QTE(Q1) for Model 1, $n = 1000$.

The numerical results of different methods for ATTs and QTTs for Model 1 are presented in Tables 3.5 - 3.8. It is shown that the numerical results of different methods for estimating ATTs and QTTs have similar patterns as those given in Tables 3.1 - 3.4 for ATEs and QTEs.

All numerical results of different methods for ATTs and QTTs for Model 2 are presented in Tables 3.9 - 3.16. In Model 2, both PS and OR functions are generated from linear models, so the GLM and GAM methods no longer have the model misspecification problem, and GLM is expected to have the best performance. However, we can see from Tables 3.9 - 3.16 that the ANN and DNN methods have comparable performance to GLM for estimation of TEs in all cases. It is worth noting that our proposed method can also be applied to the estimation of asymmetric least squares TEs and other types of TEs, and it has similar patterns of numerical performance as the estimation of ATEs and QTEs. The numerical results are not presented due to space limitations.

At last, we evaluate the performance of our proposed TE estimators in the settings with $p = 100$ and $n = 2000$. In this scenario, the number of confounders is very large compared to the sample size, and it does not satisfy the order requirement given in Assumption 4. Note that when dealing with high-dimensional covariates, one often assumes a parametric structure on the regression model and imposes a sparsity condition such that a small number of covariates are useful for the prediction [20, 19]. The sparsity assumption and the parametric structure are not required in our setting. For the purpose of dimensionality reduction, we apply Principal Component Analysis (PCA) to extract the leading principal components that can explain the variance by at least 90% of the covariates matrix, and use them to estimate the PS and OR functions via ANNs. For comparison, we also use the original covariates matrix without PCA to fit the nuisance models via ANNs. The resulting TE estimators with and without the PCA procedure are called ANN-PCA and ANN, respectively. Table 3.17 - 3.18 report the summary statistics of the ANN-based TE estimators for ATE, ATT, QTE and QTT for Model 1 with $p = 100$ and $n = 2000$, based on 400 simulation realizations. For QTE and QTT, we only report the estimated standard deviations and empirical coverage rates from the weighted bootstrapping, as it is difficult to estimate the asymptotic standard deviations in the quantile settings. The ATE and ATT are estimated by the IPW, OR and DR methods, respectively, while the QTE and QTT are only estimated by the IPW method.

From Table 3.17, we see that the empirical coverage rates obtained from all of the three methods are smaller than the nominal level 0.95, and the values of bias and emp_sd are larger than those values given in Tables 3.1 - 3.2 for $p = 5, 10$. The ANN-PCA method performs better, although its empirical coverage rates still cannot reach the nominal level. It is expected that these ANN-based methods do not perform well, as the order assumption on the dimension p does not hold anymore. As a result, the ANN-based estimators of the nuisance functions are yielding deteriorated performance, and those estimates further affect the estimation of ATE and ATT. The formula of est_sd involves the estimates of both OR and PS functions, so it is not surprising that its value is also affected. However, from Table 3.18, we see that the IPW method has a better and more stable performance for estimation of QTE and QTT for $p = 100$, the empirical coverage rates for QTE are quite close to 95% since the QTE estimators are more robust to the estimate of the nuisance function.

	IPW						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0008**	0.0584	0.0594	0.0523	0.0562	0.0015	**0.0010**	0.0586	0.0574	0.0151	0.0144	0.0012	0.0026	0.0006
emp_sd	**0.0758**	0.0907	0.0924	0.0830	0.0844	0.0789	**0.0713**	0.0906	0.0930	0.0769	0.0828	0.0726	0.0752	0.0668
est_sd	**0.0701**	0.0923	0.0903	0.0511	0.0483	0.0715	**0.0701**	0.0923	0.0903	0.0511	0.0483	0.0715	0.0701	0.0686
cover_rate	**0.9275**	0.9025	0.8750	0.6700	0.6400	0.9250	**0.9425**	0.9050	0.8900	0.8000	0.7450	0.9425	0.9325	0.9600
est_sd_boot	**0.0771**	0.0901	0.0922	0.0821	0.0853	0.0791	**0.0751**	0.0902	0.0924	0.0773	0.0841	0.0737		
cover_rate_boot	**0.9375**	0.8850	0.0888	0.9025	0.8950	0.9375	**0.9475**	0.8875	0.8875	0.9275	0.9350	0.9500		
n=2000														
bias	**0.0031**	0.0648	0.0635	0.0535	0.0427	0.0028	**0.0040**	0.0650	0.0633	0.0156	0.0171	0.0010	0.0057	0.0050
emp_sd	**0.0524**	0.0684	0.0675	0.0620	0.0634	0.0563	**0.0499**	0.0685	0.0677	0.0564	0.0575	0.0502	0.0523	0.0498
est_sd	**0.0488**	0.0652	0.0637	0.0371	0.0370	0.0508	**0.0488**	0.0652	0.0637	0.0371	0.0370	0.0508	0.0493	0.0485
cover_rate	**0.9375**	0.8175	0.8100	0.6250	0.6500	0.9175	**0.9425**	0.8175	0.8050	0.7875	0.7575	0.9475	0.9425	0.9425
est_sd_boot	**0.0573**	0.0679	0.0665	0.0612	0.0645	0.0599	**0.0493**	0.0685	0.0697	0.0563	0.0572	0.0505		
cover_rate_boot	**0.9475**	0.8225	0.8350	0.8375	0.8550	0.9350	**0.9475**	0.8250	0.8175	0.9150	0.9025	0.9475		
n=5000														
bias	**0.0001**	0.0558	0.0545	0.0407	0.0225	0.0002	**0.0003**	0.0558	0.0545	0.0043	0.0073	0.0003	0.0009	0.0010
emp_sd	**0.0334**	0.0397	0.0390	0.0362	0.0376	0.0335	**0.0312**	0.0397	0.0389	0.0324	0.0327	0.0305	0.0322	0.0309
est_sd	**0.0309**	0.0413	0.0403	0.0244	0.0258	0.0307	**0.0309**	0.0413	0.0403	0.0244	0.0258	0.0307	0.0306	0.0307
cover_rate	**0.9350**	0.7350	0.7175	0.5700	0.7400	0.9250	**0.9475**	0.7375	0.7150	0.8500	0.8450	0.9525	0.9400	0.9600
est_sd_boot	**0.0331**	0.0402	0.0401	0.0355	0.0365	0.0337	**0.0308**	0.0399	0.0402	0.0327	0.0331	0.0305		
cover_rate_boot	**0.9475**	0.7350	0.7225	0.7850	0.8725	0.9475	**0.9575**	0.7350	0.7250	0.9325	0.9250	0.9525		

Table 3.1: The summary statistics of the estimated ATEs for Model 1 with p=5

	IPW						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0010**	0.0635	0.0655	0.0574	0.0536	0.0010	**0.0064**	0.0637	0.0640	0.0264	0.0279	0.0037	0. 0073	0.0077
emp_sd	**0.0855**	0.0900	0.1127	0.0887	0.0915	0.0859	**0.0793**	0.0901	0.1041	0.0825	0.0832	0.0785	0.0823	0.0666
est_sd	**0.0789**	0.0943	0.0982	0.0547	0.0512	0.0784	**0.0789**	0.0943	0.0982	0.0547	0.0512	0.0784	0.0789	0.0695
cover_rate	**0.9250**	0.9075	0.8675	0.6725	0.6500	0.9275	**0.9400**	0.9075	0.8800	0.7775	0.7375	0.9450	0.9325	0.9550
est_sd_boot	**0.0884**	0.0923	0.1089	0.0893	0.0921	0.0864	**0.0798**	0.0913	0.0965	0.0833	0.0856	0.0778		
cover_rate_boot	**0.9350**	0.8950	0.8825	0.8925	0.8950	0.9375	**0.9475**	0.8950	0.8650	0.9150	0.9125	0.9450		
n=2000														
bias	**0.0034**	0.0659	0.0661	0.0584	0.0462	0.0032	**0.0001**	0.0660	0.0675	0.0227	0.0266	0.0009	0.0012	0.0050
emp_sd	**0.0597**	0.0660	0.0692	0.0634	0.0629	0.0657	**0.0543**	0.0661	0.0683	0.0588	0.0590	0.0519	0.0558	0.0501
est_sd	**0.0532**	0.0668	0.0667	0.0397	0.0382	0.0539	**0.0532**	0.0668	0.0667	0.0397	0.0382	0.0539	0.0530	0.0492
cover_rate	**0.9250**	0.8275	0.8225	0.6050	0.6400	0.9250	**0.9400**	0.8250	0.8275	0.7825	0.7475	0.9650	0.9400	0.9500
est_sd_boot	**0.0604**	0.0668	0.0687	0.0624	0.0628	0.0649	**0.0560**	0.0664	0.0679	0.0590	0.0595	0.0530		
cover_rate_boot	**0.9475**	0.8250	0.8250	0.8125	0.8375	0.9425	**0.9500**	0.8275	0.8350	0.9025	0.8875	0.9525		
n=5000														
bias	**0.0015**	0.0679	0.0670	0.0565	0.0362	0.0012	**0.0001**	0.0679	0.0670	0.0158	0.0177	0.0009	0. 0017	0.0020
emp_sd	**0.0363**	0.0439	0.0439	0.0416	0.0419	0.0357	**0.0332**	0.0440	0.0439	0.0376	0.0376	0.0319	0.0345	0.0319
est_sd	**0.0322**	0.0422	0.0418	0.0258	0.0267	0.0319	**0.0322**	0.0422	0.0418	0.0258	0.0267	0.0319	0.0323	0.0311
cover_rate	**0.9275**	0.6100	0.5975	0.4225	0.6150	0.9250	**0.9425**	0.6125	0.6000	0.7875	0.7700	0.9500	0.9475	0.9550
est_sd_boot	**0.0368**	0.0441	0.0443	0.0420	0.0423	0.0349	**0.0333**	0.0440	0.0442	0.0376	0.0378	0.0323		
cover_rate_boot	**0.9475**	0.6550	0.6475	0.6575	0.7875	0.9450	**0.9525**	0.6250	0.6175	0.8275	0.8150	0.9525		

Table 3.2: The summary statistics of the estimated ATEs for Model 1 with p=10

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0045**	0.0601	0.0589	0.0630	0.0684	0.0036	**0.0012**	0.0534	0.0574	0.0507	0.0539	0.0025	**0.0098**	0.0363	0.0407	0.0286	0.0302	0.0067
emp_sd	**0.1266**	0.1358	0.1418	0.1274	0.1301	0.1235	**0.1075**	0.1116	0.1166	0.1055	0.1074	0.1099	**0.1196**	0.1151	0.1201	0.1121	0.1130	0.1206
est_sd	**0.1414**	0.1403	0.1596	0.1398	0.1405	0.1404	**0.1238**	0.1240	0.1447	0.1241	0.1249	0.1222	**0.1251**	0.1232	0.1412	0.1237	0.1247	0.1243
cover_rate	**0.9500**	0.9275	0.9375	0.9350	0.9250	0.9525	**0.9675**	0.9550	0.9500	0.9675	0.9650	0.9575	**0.9600**	0.9650	0.9650	0.9650	0.9625	0.9625
est_sd_boot	**0.1259**	0.1378	0.1432	0.1275	0.1305	0.1278	**0.1091**	0.1154	0.1171	0.1087	0.1072	0.1093	**0.1231**	0.1131	0.1241	0.1172	0.1187	0.1236
cover_rate_boot	**0.9350**	0.9225	0.9300	0.9275	0.9175	0.9375	**0.9600**	0.9500	0.9475	0.9500	0.9475	0.9500	**0.9575**	0.9450	0.9500	0.9575	0.9550	0.9600
n=2000																		
bias	**0.0126**	0.0794	0.0760	0.0731	0.0746	0.0096	**0.0004**	0.0547	0.0523	0.0454	0.0467	0.0014	**0.0056**	0.0494	0.0506	0.0372	0.0369	0.0043
emp_sd	**0.0928**	0.1029	0.1012	0.0944	0.0944	0.0954	**0.0819**	0.0883	0.0884	0.0828	0.0844	0.0831	**0.0852**	0.0872	0.0888	0.0848	0.0860	0.0864
est_sd	**0.0900**	0.0986	0.0976	0.0984	0.0989	0.0933	**0.0791**	0.0870	0.0872	0.0871	0.0875	0.0804	**0.0814**	0.0860	0.0871	0.0862	0.0867	0.0824
cover_rate	**0.9275**	0.8550	0.8650	0.8900	0.8775	0.9375	**0.9350**	0.9050	0.9050	0.9300	0.9225	0.9325	**0.9550**	0.9100	0.9175	0.9425	0.9350	0.9375
est_sd_boot	**0.0911**	0.1014	0.1023	0.0954	0.0968	0.0961	**0.0811**	0.0889	0.0892	0.0857	0.0853	0.0842	**0.0841**	0.0872	0.0891	0.0857	0.0871	0.0858
cover_rate_boot	**0.9350**	0.8575	0.8825	0.8750	0.8650	0.9450	**0.9425**	0.9100	0.9025	0.9125	0.9200	0.9500	**0.9375**	0.9150	0.9200	0.9375	0.9350	0.9450
n=5000																		
bias	**0.0034**	0.0594	0.0560	0.0456	0.0255	0.0029	**0.0033**	0.0489	0.0478	0.0367	0.0188	0.0036	**0.0026**	0.0454	0.0460	0.0320	0.0176	0.0014
emp_sd	**0.0560**	0.0600	0.0591	0.0555	0.0559	0.0570	**0.0514**	0.0545	0.0539	0.0520	0.0522	0.0517	**0.0494**	0.0513	0.0508	0.0497	0.0503	0.0490
est_sd	**0.0558**	0.0624	0.0607	0.0624	0.0639	0.0531	**0.0493**	0.0546	0.0537	0.0546	0.0557	0.0510	**0.0508**	0.0542	0.0540	0.0542	0.0551	0.0513
cover_rate	**0.9575**	0.8625	0.8600	0.9125	0.9575	0.9575	**0.9400**	0.8675	0.8675	0.9200	0.9525	0.9475	**0.9625**	0.8900	0.8850	0.9300	0.9575	0.9675
est_sd_boot	**0.0560**	0.0614	0.0603	0.0576	0.0578	0.0560	**0.0509**	0.0553	0.5420	0.0534	0.0539	0.0509	**0.5010**	0.0517	0.0515	0.0508	0.0512	0.5010
cover_rate_boot	**0.9575**	0.8575	0.8550	0.8825	0.9325	0.9575	**0.9500**	0.8725	0.8725	0.9075	0.9500	0.9500	**0.9600**	0.8575	0.8550	0.9200	0.9325	0.9625

Table 3.3: The summary statistics of the estimated QTEs by the IPW method for Model 1 with p=5

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0019**	0.0717	0.0750	0.0700	0.0696	0.0023	**0.0047**	0.0544	0.0541	0.0507	0.0472	0.0033	**0.0030**	0.0488	0.0476	0.0394	0.0367	0.0036
emp_sd	**0.1585**	0.1345	0.1634	0.1325	0.1364	0.1592	**0.1437**	0.1195	0.1485	0.1179	0.1206	0.1437	**0.1332**	0.1185	0.1529	0.1193	0.1208	0.1331
est_sd	**0.2530**	0.1444	0.4770	0.1442	0.1487	0.2630	**0.2639**	0.1269	0.4708	0.1269	0.1312	0.2684	**0.2034**	0.1253	0.4458	0.1255	0.1294	0.2109
cover_rate	**0.9675**	0.9200	0.9950	0.9350	0.9375	0.9675	**0.9425**	0.9450	1.0000	0.9475	0.9525	0.9450	**0.9575**	0.9325	1.0000	0.9350	0.9500	0.9575
est_sd_boot	**0.1623**	0.1345	0.1552	0.1376	0.1425	0.1687	**0.1523**	0.1231	0.1253	0.1198	0.1225	0.1523	**0.1376**	0.1203	0.1623	0.1198	0.1256	0.1392
cover_rate_boot	**0.9325**	0.9150	0.9125	0.9300	0.9250	0.9350	**0.9350**	0.9350	0.9375	0.9400	0.9475	0.9500	**0.9325**	0.9225	0.9375	0.9000	0.9225	0.9350
n=2000																		
bias	**0.0030**	0.0663	0.0678	0.0632	0.0535	0.0023	**0.0015**	0.0576	0.0580	0.0529	0.0421	0.0005	**0.0105**	0.0613	0.0620	0.0527	0.0426	0.0087
emp_sd	**0.1070**	0.1040	0.1083	0.1008	0.1016	0.1109	**0.0937**	0.0902	0.0943	0.0874	0.0872	0.0954	**0.0873**	0.0855	0.0890	0.0833	0.0830	0.0923
est_sd	**0.1057**	0.1021	0.1149	0.1019	0.1052	0.1134	**0.0919**	0.0889	0.1026	0.0889	0.0917	0.0939	**0.0913**	0.0878	0.1001	0.0878	0.0904	0.0919
cover_rate	**0.9425**	0.8775	0.8850	0.8900	0.9150	0.9475	**0.9200**	0.8900	0.9175	0.9100	0.9250	0.9350	**0.9425**	0.9125	0.9325	0.9275	0.9400	0.9525
est_sd_boot	**0.1072**	0.1043	0.1097	0.1011	0.1045	0.1122	**0.0932**	0.0901	0.0994	0.0883	0.0892	0.0952	**0.0885**	0.0861	0.0923	0.0869	0.0873	0.0925
cover_rate_boot	**0.9450**	0.8800	0.8000	0.8900	0.9125	0.9450	**0.9325**	0.8975	0.9150	0.9100	0.9175	0.9450	**0.9375**	0.9050	0.9125	0.9250	0.9275	0.9575
n=5000																		
bias	**0.0065**	0.0774	0.0764	0.0696	0.0472	0.0055	**0.0046**	0.0637	0.0633	0.0547	0.0356	0.0051	**0.0021**	0.0537	0.0535	0.0433	0.0273	0.0031
emp_sd	**0.0657**	0.0682	0.0680	0.0650	0.0656	0.0634	**0.0568**	0.0580	0.0584	0.0560	0.0567	0.0532	**0.0554**	0.0555	0.0562	0.0549	0.0563	0.0531
est_sd	**0.0629**	0.0644	0.0638	0.0644	0.0659	0.0615	**0.0537**	0.0556	0.0555	0.0556	0.0568	0.0531	**0.0542**	0.0550	0.0551	0.0549	0.0560	0.0522
cover_rate	**0.9275**	0.7675	0.7725	0.8025	0.8875	0.9275	**0.9350**	0.7800	0.7725	0.8225	0.9075	0.9450	**0.9375**	0.8375	0.8300	0.8650	0.9250	0.9375
est_sd_boot	**0.0655**	0.0686	0.0678	0.0648	0.0658	0.0635	**0.0565**	0.0577	0.0579	0.0552	0.0571	0.0545	**0.0548**	0.0561	0.0568	0.0552	0.0561	0.0528
cover_rate_boot	**0.9325**	0.7825	0.7875	0.8125	0.8875	0.9450	**0.9475**	0.7900	0.7850	0.8200	0.9100	0.9525	**0.9400**	0.8500	0.8525	0.8675	0.9275	0.9425

Table 3.4: The summary statistics of the estimated QTEs by the IPW method for Model 1 with p=10

	PS						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0002**	0.0720	0.0767	0.0982	0.1182	0.0034	**0.0009**	0.0725	0.0700	0.0188	0.0198	0.0012	0.0045	0.0009
emp_sd	**0.0905**	0.0912	0.1024	0.0814	0.0877	0.0873	**0.0813**	0.0910	0.1018	0.0818	0.0975	0.0809	0.0826	0.0723
est_sd	**0.0804**	0.0933	0.1021	0.0632	0.0677	0.0799	**0.0804**	0.0933	0.1021	0.0632	0.0677	0.0799	0.0804	0.0733
cover_rate	**0.9225**	0.8850	0.8875	0.6025	0.5400	0.9250	**0.9500**	0.8875	0.8875	0.8700	0.8375	0.9500	0.9400	0.9500
est_sd_boot	**0.0899**	0.0921	0.1025	0.0816	0.0884	0.0865	**0.0815**	0.0925	0.1025	0.0825	0.0945	0.0805		
cover_rate_boot	**0.9350**	0.8825	0.8850	0.8875	0.8650	0.9325	**0.9525**	0.8850	0.8875	0.9325	0.9050	0.9500		
n=2000														
bias	**0.0066**	0.0778	0.0802	0.0893	0.0969	0.0078	**0.0052**	0.0783	0.0797	0.0177	0.0220	0.0024	0.0036	0.0050
emp_sd	**0.0581**	0.0687	0.0705	0.0614	0.0634	0.0677	**0.0585**	0.0685	0.0704	0.0585	0.0642	0.0574	0.0576	0.0518
est_sd	**0.0549**	0.0657	0.0684	0.0458	0.0473	0.0552	**0.0549**	0.0657	0.0684	0.0458	0.0473	0.0552	0.0542	0.0518
cover_rate	**0.9275**	0.7850	0.7725	0.5175	0.4800	0.9050	**0.9400**	0.7850	0.7800	0.8475	0.8275	0.9450	0.9375	0.9525
est_sd_boot	**0.0653**	0.0679	0.0694	0.0609	0.0643	0.0667	**0.0582**	0.0674	0.0693	0.0599	0.0631	0.0593		
cover_rate_boot	**0.9325**	0.7950	0.7800	0.6875	0.6775	0.9300	**0.9475**	0.7900	0.7825	0.9025	0.8575	0.9475		
n=5000														
bias	**0.0004**	0.0689	0.0712	0.0666	0.0405	0.0010	**0.0012**	0.0691	0.0711	0.0066	0.0092	0.0003	0.0017	0.0010
emp_sd	**0.0356**	0.0398	0.0404	0.0366	0.0415	0.0357	**0.0353**	0.0398	0.0402	0.0349	0.0366	0.0357	0.0365	0.0339
est_sd	**0.0339**	0.0415	0.0417	0.0302	0.0328	0.0341	**0.0339**	0.0415	0.0417	0.0302	0.0328	0.0341	0.0334	0.0328
cover_rate	**0.9250**	0.6225	0.6125	0.4250	0.7175	0.9225	**0.9350**	0.6150	0.6050	0.9225	0.8850	0.9450	0.9350	0.9350
est_sd_boot	**0.0359**	0.0399	0.0403	0.0360	0.0376	0.0365	**0.0348**	0.0405	0.0406	0.0353	0.0369	0.0351		
cover_rate_boot	**0.9375**	0.6075	0.6050	0.5875	0.8050	0.9375	**0.9450**	0.6075	0.6000	0.9325	0.9200	0.9475		

Table 3.5: The summary statistics of the estimated ATTs for Model 1 with p=5

	PS						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0003**	0.0805	0.0920	0.0922	0.0954	0.0006	**0.0105**	0.0812	0.0847	0.0350	0.0380	0.0620	0. 0111	0.0085
emp_sd	**0.1062**	0.0903	0.1437	0.0897	0.1086	0.1033	**0.0931**	0.0905	0.1271	0.0825	0.0904	0.0928	0.0989	0.0723
est_sd	**0.0936**	0.0961	0.1220	0.0681	0.0759	0.0931	**0.0936**	0.0961	0.1220	0.0681	0.0759	0.0931	0.0936	0.0750
cover_rate	**0.9050**	0.8875	0.8350	0.6725	0.6525	0.9100	**0.9425**	0.8825	0.8725	0.8525	0.8750	0.9525	0.9275	0.9600
est_sd_boot	**0.1056**	0.0954	0.1332	0.0912	0.1034	0.1073	**0.0933**	0.0955	0.1251	0.0833	0.0899	0.0933		
cover_rate_boot	**0.9350**	0.8850	0.8450	0.8650	0.8475	0.9375	**0.9475**	0.8825	0.8750	0.9125	0.9075	0.9575		
n=2000														
bias	**0.0025**	0.0833	0.0856	0.0891	0.0745	0.0024	**0.0008**	0.0836	0.0878	0.0283	0.0332	0.0010	0.0016	0.0073
emp_sd	**0.0693**	0.0667	0.0767	0.0628	0.0688	0.0701	**0.0627**	0.0668	0.0751	0.0602	0.0632	0.0631	0.0661	0.0546
est_sd	**0.0620**	0.0676	0.0754	0.0497	0.0544	0.0635	**0.0620**	0.0676	0.0754	0.0497	0.0544	0.0635	0.0616	0.0531
cover_rate	**0.9175**	0.7650	0.7675	0.5575	0.6650	0.9050	**0.9350**	0.7650	0.7850	0.8550	0.8725	0.9425	0.9325	0.9375
est_sd_boot	**0.0701**	0.0668	0.0769	0.0634	0.0692	0.0702	**0.0622**	0.0675	0.0755	0.0609	0.0644	0.0632		
cover_rate_boot	**0.9425**	0.7700	0.7775	0.6650	0.0733	0.9250	**0.9475**	0.7625	0.7850	0.8925	0.9075	0.9450		
n=5000														
bias	**0.0013**	0.0853	0.0863	0.0814	0.0538	0.0004	**0.0030**	0.0854	0.0861	0.0194	0.0224	0.0002	0. 004	0.0028
emp_sd	**0.0398**	0.0442	0.0460	0.0415	0.0449	0.0403	**0.0372**	0.0443	0.0458	0.0377	0.0390	0.0362	0.0369	0.0343
est_sd	**0.0364**	0.0425	0.0442	0.0323	0.0353	0.0361	**0.0364**	0.0425	0.0442	0.0323	0.0353	0.0361	0.0361	0.0336
cover_rate	**0.9200**	0.4575	0.5000	0.3150	0.6200	0.9200	**0.9475**	0.4500	0.5000	0.8650	0.8475	0.9625	0.9425	0.9600
est_sd_boot	**0.0414**	0.0441	0.0445	0.0423	0.0432	0.0414	**0.0371**	0.0442	0.0453	0.0365	0.0378	0.0366		
cover_rate_boot	**0.9475**	0.4600	0.4575	0.4850	0.7925	0.9475	**0.9525**	0.4675	0.5125	0.8875	0.8650	0.9650		

Table 3.6: The summary statistics of the estimated ATTs for Model 1 with p=10

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0074**	0.0751	0.0794	0.1311	0.1623	0.0056	**0.0019**	0.0644	0.0686	0.0973	0.1151	0.0024	**0.0093**	0.0445	0.0513	0.0502	0.0610	0.0076
emp_sd	**0.1503**	0.1360	0.1527	0.1289	0.1372	0.1515	**0.1203**	0.1124	0.1295	0.1048	0.1116	0.1241	**0.1293**	0.1162	0.1306	0.1116	0.1199	0.1299
est_sd	**0.1660**	0.1409	0.1638	0.1334	0.1352	0.1679	**0.1517**	0.1245	0.1481	0.1199	0.1218	0.1567	**0.1360**	0.1238	0.1447	0.1204	0.1227	0.1378
cover_rate	**0.9550**	0.9050	0.9125	0.8350	0.7625	0.9525	**0.9650**	0.9425	0.9475	0.9150	0.8575	0.9600	**0.9375**	0.9575	0.9550	0.9425	0.9350	0.9400
est_sd_boot	**0.1576**	0.1367	0.1578	0.1298	0.1385	0.1592	**0.1272**	0.1176	0.1337	0.1067	0.1134	0.1281	**0.1205**	0.1182	0.1376	0.1134	0.1231	0.1285
cover_rate_boot	**0.9475**	0.8850	0.8950	0.8225	0.7850	0.9500	**0.9450**	0.9375	0.9325	0.8975	0.8350	0.9450	**0.9250**	0.9475	0.9375	0.9350	0.9400	0.9375
n=2000																		
bias	**0.0111**	0.0949	0.0960	0.1258	0.1388	0.0113	**0.0006**	0.0657	0.0669	0.0824	0.0918	0.0014	**0.0059**	0.0575	0.0614	0.0565	0.0594	0.0044
emp_sd	**0.1002**	0.1027	0.1066	0.0941	0.0966	0.1032	**0.0905**	0.0890	0.0923	0.0831	0.0875	0.0921	**0.0885**	0.0864	0.0914	0.0841	0.0867	0.0893
est_sd	**0.0974**	0.0988	0.1013	0.0955	0.0968	0.0987	**0.0842**	0.0872	0.0902	0.0849	0.0859	0.0903	**0.0844**	0.0862	0.0898	0.0843	0.0855	0.0856
cover_rate	**0.9375**	0.8225	0.8400	0.7425	0.6925	0.9300	**0.9225**	0.8800	0.8600	0.8275	0.8150	0.9425	**0.9425**	0.9000	0.9100	0.8925	0.8800	0.9375
est_sd_boot	**0.1009**	0.1028	0.1050	0.0951	0.0970	0.1019	**0.0912**	0.0889	0.0915	0.0838	0.0869	0.0917	**0.0854**	0.0865	0.0914	0.0845	0.0859	0.0869
cover_rate_boot	**0.9400**	0.8350	0.8525	0.7550	0.7050	0.9375	**0.9350**	0.8950	0.8850	0.8350	0.8225	0.9450	**0.9400**	0.9025	0.9175	0.9000	0.8925	0.9450
n=5000																		
bias	**0.0046**	0.0749	0.0749	0.0826	0.0516	0.0036	**0.0029**	0.0601	0.0616	0.0621	0.0377	0.0025	**0.0016**	0.0526	0.0564	0.0455	0.0269	0.0027
emp_sd	**0.0623**	0.0601	0.0611	0.0564	0.0610	0.0630	**0.0553**	0.0547	0.0553	0.0524	0.0555	0.0561	**0.0523**	0.0513	0.0521	0.0501	0.0538	0.0531
est_sd	**0.0578**	0.0625	0.0618	0.0615	0.0658	0.0590	**0.0499**	0.0546	0.0546	0.0538	0.0567	0.0502	**0.0510**	0.0543	0.0549	0.0534	0.0556	0.0521
cover_rate	**0.9250**	0.8000	0.7875	0.7625	0.8925	0.9175	**0.9275**	0.8125	0.7900	0.8000	0.9050	0.9250	**0.9450**	0.8600	0.8500	0.8800	0.9250	0.9425
est_sd_boot	**0.0608**	0.0605	0.0609	0.0584	0.0602	0.0614	**0.0544**	0.0545	0.0547	0.0533	0.0573	0.0557	**0.0519**	0.0529	0.0533	0.0516	0.0548	0.0528
cover_rate_boot	**0.9375**	0.7850	0.7775	0.7350	0.8525	0.9275	**0.9375**	0.8125	0.7925	0.7975	0.9125	0.9425	**0.9475**	0.8500	0.8425	0.8725	0.9225	0.9475

Table 3.7: The summary statistics of the estimated QTTs by the IPW method for Model 1 with p=5

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0061**	0.0871	0.1015	0.1230	0.1359	0.0081	**0.0098**	0.0659	0.0732	0.0815	0.0907	0.0046	**0.0022**	0.0621	0.0701	0.0594	0.0572	0.0027
emp_sd	**0.2019**	0.1370	0.2086	0.1362	0.1684	0.2025	**0.1749**	0.1200	0.1860	0.1210	0.1435	0.1799	**0.1549**	0.1181	0.1800	0.1211	0.1369	0.1555
est_sd	**0.3678**	0.1458	0.4513	0.1355	0.1481	0.3421	**0.3018**	0.1281	0.4455	0.1209	0.1317	0.2975	**0.2504**	0.1264	0.4221	0.1209	0.1309	0.2321
cover_rate	**0.9600**	0.9150	0.9750	0.8475	0.8175	0.9575	**0.9425**	0.9225	0.9975	0.8875	0.8475	0.9450	**0.9675**	0.9275	0.9925	0.9075	0.9175	0.9575
est_sd_boot	**0.2165**	0.1399	0.2353	0.1366	0.1702	0.2065	**0.1842**	0.1256	0.1932	0.1232	0.1452	0.1820	**0.1634**	0.1203	0.2010	0.1232	0.1379	0.1655
cover_rate_boot	**0.9300**	0.9250	0.9425	0.8575	0.8475	0.9250	**0.9350**	0.9175	0.9525	0.8950	0.8550	0.9400	**0.9425**	0.9225	0.9475	0.9150	0.9275	0.9500
n=2000																		
bias	**0.0022**	0.0853	0.0900	0.1090	0.1021	0.0024	**0.0011**	0.0711	0.0737	0.0830	0.0715	0.0023	**0.0090**	0.0733	0.0783	0.0692	0.0562	0.0047
emp_sd	**0.1229**	0.1054	0.1200	0.1005	0.1105	0.1321	**0.1066**	0.0911	0.1031	0.0868	0.0942	0.1086	**0.0948**	0.0852	0.0971	0.0840	0.0907	0.0963
est_sd	**0.1264**	0.1026	0.1189	0.0974	0.1070	0.1361	**0.1089**	0.0894	0.1053	0.0857	0.0931	0.1093	**0.1009**	0.0882	0.1026	0.0853	0.0917	0.1024
cover_rate	**0.9475**	0.8325	0.8500	0.7825	0.8200	0.9500	**0.9350**	0.8725	0.8700	0.8500	0.8725	0.9375	**0.9650**	0.8925	0.8975	0.8850	0.9225	0.9650
est_sd_boot	**0.1225**	0.1049	0.1199	0.0998	0.1112	0.1325	**0.1072**	0.9070	0.1053	0.0871	0.0942	0.1082	**0.0996**	0.0863	0.0971	0.0852	0.0911	0.0978
cover_rate_boot	**0.9450**	0.8350	0.8500	0.7900	0.8325	0.9475	**0.9300**	0.8750	0.8725	0.8550	0.8800	0.9400	**0.9525**	0.8900	0.8950	0.8850	0.9200	0.9575
n=5000																		
bias	**0.0061**	0.0966	0.0980	0.1047	0.7330	0.0042	**0.0019**	0.0770	0.0776	0.0771	0.0520	0.0032	**0.0004**	0.0654	0.0661	0.0567	0.0352	0.0012
emp_sd	**0.0731**	0.0682	0.0705	0.0643	0.0695	0.0727	**0.0615**	0.0586	0.0604	0.0555	0.0600	0.0595	**0.0578**	0.0557	0.0574	0.0547	0.0582	0.0571
est_sd	**0.0654**	0.0646	0.0660	0.0628	0.0676	0.0714	**0.0555**	0.0558	0.0572	0.0544	0.0577	0.0565	**0.0552**	0.0551	0.0567	0.0539	0.0564	0.0564
cover_rate	**0.9075**	0.6575	0.6600	0.6000	0.7825	0.9375	**0.9225**	0.6850	0.7050	0.6975	0.8475	0.9325	**0.9375**	0.7850	0.7850	0.8100	0.9000	0.9475
est_sd_boot	**0.0725**	0.0681	0.0699	0.0639	0.0682	0.0715	**0.0592**	0.0572	0.0598	0.0558	0.0594	0.0592	**0.0572**	0.0561	0.0569	0.0554	0.0577	0.0562
cover_rate_boot	**0.9125**	0.6700	0.6675	0.6875	0.7900	0.9325	**0.9350**	0.6925	0.7125	0.7000	0.8575	0.9400	**0.9425**	0.7900	0.7925	0.8225	0.9075	0.9425

Table 3.8: The summary statistics of the estimated QTTs by the IPW method for Model 1 with p=10

	PS						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0043**	0.0207	0.0214	0.2254	0.1886	0.0037	**0.0203**	0.0194	0.0195	0.0499	0.0385	0.0189	0.0200	0.0190
emp_sd	**0.1676**	0.1182	0.1424	0.1933	0.1839	0.1701	**0.1235**	0.1185	0.1204	0.1367	0.1345	0.1227	0.1219	0.1190
est_sd	**0.1207**	0.1203	0.1205	0.1198	0.1178	0.1213	**0.1207**	0.1203	0.1205	0.1198	0.1178	0.1213	0.1208	0.1203
cover_rate	**0.9225**	0.9525	0.9000	0.4950	0.5700	0.9200	**0.9475**	0.9525	0.9525	0.8850	0.8850	0.9475	0.9500	0.9575
est_sd_boot	**0.1702**	0.1189	0.1398	0.1821	0.1671	0.1702	**0.1223**	0.1193	0.1209	0.1351	0.1331	0.1231		
cover_rate_boot	**0.9375**	0.9500	0.9250	0.5425	0.6850	0.9300	**0.9500**	0.9500	0.9550	0.9150	0.9325	0.9525		
n=2000														
bias	**0.0069**	0.0072	0.0094	0.1674	0.0574	0.0054	**0.0073**	0.0073	0.0072	0.0210	0.0169	0.0021	0.0070	0.0074
emp_sd	**0.0910**	0.0828	0.0881	0.1183	0.0936	0.0923	**0.0838**	0.0829	0.0834	0.0914	0.0905	0.0845	0.0838	0.0828
est_sd	**0.0851**	0.0852	0.0852	0.0833	0.0832	0.0854	**0.0851**	0.0852	0.0852	0.0833	0.0832	0.0854	0.0851	0.0851
cover_rate	**0.9250**	0.9550	0.9425	0.4875	0.8550	0.9225	**0.9500**	0.9550	0.9550	0.9150	0.9200	0.9500	0.9475	0.9550
est_sd_boot	**0.0878**	0.0833	0.0871	0.1007	0.0927	0.0921	**0.0841**	0.0834	0.0841	0.0921	0.0911	0.0844		
cover_rate_boot	**0.9375**	0.9550	0.9475	0.5650	0.9275	0.9425	**0.9500**	0.9550	0.9500	0.9275	0.9425	0.9475		
n=5000														
bias	**0.0007**	0.0001	0.0004	0.1105	0.0399	0.0005	**0.0008**	0.0001	0.0002	0.0008	0.0050	0.0003	0.0013	0.0001
emp_sd	**0.0560**	0.0532	0.0542	0.0661	0.0607	0.0566	**0.0529**	0.0527	0.0528	0.0567	0.0570	0.0525	0.0536	0.0526
est_sd	**0.0538**	0.0539	0.0539	0.0519	0.0527	0.0536	**0.0538**	0.0539	0.0539	0.0519	0.0527	0.0536	0.0538	0.0538
cover_rate	**0.9450**	0.9500	0.9450	0.4400	0.8350	0.9425	**0.9525**	0.9525	0.9550	0.9275	0.9300	0.9525	0.9450	0.9500
est_sd_boot	**0.0554**	0.0535	0.0542	0.0646	0.0603	0.0554	**0.0523**	0.0528	0.0528	0.0587	0.0567	0.0523		
cover_rate_boot	**0.9475**	0.9500	0.9475	0.5850	0.8850	0.9475	**0.9500**	0.9500	0.9525	0.9450	0.9475	0.9500		

Table 3.9: The summary statistics of the estimated ATEs for Model 2 with p=5

	PS						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0107**	0.0004	0.0067	0.3506	0.1334	0.0980	**0.0013**	0.0020	0.0049	0.1367	0.0679	0.0045	0.0073	0.0020
emp_sd	**0.1582**	0.1285	0.2410	0.2218	0.1641	0.1632	**0.1323**	0.1265	0.1283	0.1832	0.1635	0.1330	0.1382	0.1262
est_sd	**0.1359**	0.1293	0.1317	0.1427	0.1369	0.1360	**0.1359**	0.1293	0.1317	0.1427	0.1369	0.1360	0.1357	0.1293
cover_rate	**0.9275**	0.9575	0.7225	0.3700	0.7675	0.9250	**0.9525**	0.9550	0.9525	0.7650	0.8850	0.9525	0.9500	0.9500
est_sd_boot	**0.1543**	0.1290	0.2214	0.2098	0.1680	0.1653	**0.1339**	0.1200	0.1287	0.1627	0.1662	0.1342		
cover_rate_boot	**0.9450**	0.9575	0.8475	0.4475	0.8025	0.9475	**0.9525**	0.9575	0.9500	0.7875	0.9200	0.9525		
n=2000														
bias	**0.0043**	0.0018	0.0017	0.2885	0.1060	0.0062	**0.0031**	0.0014	0.0018	0.1005	0.0385	0.0047	0.0027	0.0016
emp_sd	**0.1081**	0.0945	0.1127	0.1454	0.1164	0.9980	**0.0937**	0.0924	0.0938	0.1237	0.1126	0.0939	0.0962	0.0922
est_sd	**0.0923**	0.0916	0.0919	0.0997	0.0943	0.0921	**0.0923**	0.0916	0.0919	0.0997	0.0943	0.0921	0.0925	0.0916
cover_rate	**0.9225**	0.9400	0.9000	0.2650	0.7550	0.9200	**0.9475**	0.9475	0.9350	0.7950	0.8975	0.9450	0.9475	0.9500
est_sd_boot	**0.1089**	0.9530	0.1098	0.1380	0.1241	0.1004	**0.0940**	0.0925	0.9420	0.1194	0.1200	0.0944		
cover_rate_boot	**0.9450**	0.9500	0.9350	0.4550	0.8350	0.9375	**0.9500**	0.9500	0.9475	0.8225	0.9175	0.9500		
n=5000														
bias	**0.0054**	0.0022	0.0017	0.2172	0.0780	0.0035	**0.0024**	0.0025	0.0023	0.0718	0.0236	0.0031	0.0023	0.0026
emp_sd	**0.0672**	0.0523	0.0550	0.0730	0.0685	0.0633	**0.0527**	0.0519	0.0521	0.0650	0.0594	0.0519	0.0543	0.0518
est_sd	**0.0581**	0.0580	0.0580	0.0622	0.0591	0.0584	**0.0581**	0.0580	0.0580	0.0622	0.0591	0.0584	0.0581	0.0580
cover_rate	**0.9350**	0.9625	0.9575	0.1000	0.7300	0.9450	**0.9650**	0.9625	0.9625	0.7800	0.9325	0.9650	0.9650	0.9650
est_sd_boot	**0.0681**	0.0531	0.0561	0.0744	0.0648	0.0645	**0.0534**	0.0533	0.0548	0.0681	0.0601	0.0526		
cover_rate_boot	**0.9525**	0.9625	0.9575	0.1150	0.7850	0.9550	**0.9625**	0.9625	0.9625	0.7925	0.9450	0.9625		

Table 3.10: The summary statistics of the estimated ATEs for Model 2 with p=10

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0098**	0.0046	0.0104	0.2192	0.1814	0.0113	**0.0075**	0.0211	0.0242	0.2209	0.1772	0.0046	**0.0175**	0.0174	0.0262	0.2362	0.1955	0.0099
emp_sd	**0.3349**	0.3357	0.3618	0.3462	0.3479	0.3531	**0.3236**	0.3031	0.3299	0.3180	0.3255	0.3481	**0.5221**	0.3401	0.3609	0.3504	0.3496	0.4981
est_sd	**0.4023**	0.3315	0.4511	0.3507	0.3680	0.3988	**0.6874**	0.2882	0.4017	0.3124	0.3304	0.7012	**0.3824**	0.3366	0.4260	0.3535	0.3729	0.4221
cover_rate	**0.9525**	0.9475	0.9775	0.9100	0.9400	0.9500	**0.9500**	0.9300	0.9550	0.8850	0.9150	0.9600	**0.9550**	0.9450	0.9650	0.9075	0.9300	0.9500
est_sd_boot	**0.3476**	0.3375	0.3981	0.3510	0.3441	0.3476	**0.3541**	0.3012	0.3412	0.3155	0.3301	0.3522	**0.4781**	0.3398	0.3812	0.3511	0.3501	0.4911
cover_rate_boot	**0.9500**	0.9500	0.9475	0.9100	0.9325	0.9500	**0.9475**	0.9425	0.9450	0.8950	0.9150	0.9475	**0.9550**	0.9475	0.9550	0.9050	0.9375	0.9575
n=2000																		
bias	**0.0015**	0.0096	0.0185	0.1638	0.0651	0.0033	**0.0188**	0.0103	0.0141	0.1719	0.0649	0.0580	**0.0028**	0.0103	0.0080	0.1587	0.0361	0.0034
emp_sd	**0.2033**	0.2179	0.2230	0.2146	0.2247	0.2041	**0.1922**	0.1914	0.2038	0.2091	0.2011	0.1923	**0.2292**	0.2443	0.2535	0.2395	0.2496	0.2176
est_sd	**0.2028**	0.2339	0.2402	0.2408	0.2350	0.2029	**0.1961**	0.2021	0.2114	0.2108	0.2030	0.1977	**0.2032**	0.2374	0.2441	0.2427	0.2385	0.2032
cover_rate	**0.9350**	0.9675	0.9675	0.9200	0.9525	0.9350	**0.9600**	0.9650	0.9600	0.8775	0.9450	0.9600	**0.9250**	0.9600	0.9425	0.9025	0.9375	0.9275
est_sd_boot	**0.2051**	0.2241	0.2287	0.2371	0.2231	0.2055	**0.1951**	0.1921	0.2056	0.2099	0.2009	0.1971	**0.2213**	0.2447	0.2511	0.2471	0.2418	0.2183
cover_rate_boot	**0.9400**	0.9600	0.9600	0.9125	0.9500	0.9425	**0.9575**	0.9625	0.9575	0.8750	0.9450	0.9575	**0.9450**	0.9550	0.9500	0.9150	0.9400	0.9450
n=5000																		
bias	**0.0065**	0.0074	0.0084	0.1096	0.0497	0.0077	**0.0079**	0.0046	0.0036	0.1032	0.0376	0.0042	**0.0069**	0.0082	0.0069	0.1086	0.0330	0.0078
emp_sd	**0.1426**	0.1516	0.1497	0.1372	0.1492	0.1431	**0.1216**	0.1273	0.1274	0.1284	0.1299	0.1247	**0.1501**	0.1602	0.1587	0.1519	0.1629	0.1421
est_sd	**0.1226**	0.1475	0.1472	0.1489	0.1478	0.1214	**0.1158**	0.1269	0.1281	0.1288	0.1271	0.1152	**0.1229**	0.1493	0.1489	0.1500	0.1495	0.1273
cover_rate	**0.9250**	0.9400	0.9500	0.9175	0.9400	0.9250	**0.9375**	0.9600	0.9500	0.8750	0.9300	0.9350	**0.8975**	0.9325	0.9350	0.8925	0.9175	0.9125
est_sd_boot	**0.1399**	0.1498	0.1489	0.1405	0.1489	0.1402	**0.1201**	0.1266	0.1251	0.1289	0.1240	0.1221	**0.1479**	0.1576	0.1551	0.1513	0.1631	0.1401
cover_rate_boot	**0.9450**	0.9425	0.9500	0.9050	0.9450	0.9425	**0.9425**	0.9525	0.9450	0.8750	0.9275	0.9450	**0.9325**	0.9400	0.9475	0.8950	0.9200	0.9350

Table 3.11: The summary statistics of the estimated QTEs by the IPW method for Model 2 with p=5

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0153**	0.0263	0.0011	0.3586	0.1531	0.0143	**0.0055**	0.0052	0.0216	0.3419	0.1266	0.0055	**0.0141**	0.0063	0.0196	0.3714	0.1319	0.0183
emp_sd	**0.4930**	0.3731	0.4678	0.3979	0.3882	0.5320	**0.4603**	0.2955	0.4150	0.3514	0.3194	0.4980	**0.5110**	0.3733	0.5267	0.4044	0.3845	0.5980
est_sd	**0.6360**	0.3657	1.9238	0.3921	0.3707	0.7050	**0.6659**	0.3171	1.6744	0.3495	0.3217	0.7053	**0.6370**	0.3726	1.4469	0.3940	0.3758	0.6510
cover_rate	**0.9700**	0.9275	1.0000	0.8600	0.9175	0.9800	**0.9750**	0.9625	0.9975	0.8275	0.9300	0.9775	**0.9675**	0.9475	1.0000	0.8375	0.9300	0.9600
est_sd_boot	**0.4860**	0.3886	0.5360	0.4098	0.3900	0.5060	**0.4805**	0.3098	0.5629	0.3671	0.3276	0.4990	**0.4980**	0.3778	0.5873	0.4173	0.3905	0.5348
cover_rate_boot	**0.9650**	0.9300	0.9750	0.8625	0.9225	0.9700	**0.9625**	0.9625	0.9800	0.8350	0.9350	0.9600	**0.9475**	0.9475	0.9875	0.8475	0.9350	0.9500
n=2000																		
bias	**0.0150**	0.0201	0.0295	0.3091	0.1360	0.0134	**0.0085**	0.0145	0.0161	0.2804	0.0988	0.0090	**0.0081**	0.0109	0.0208	0.2954	0.0997	0.0087
emp_sd	**0.2879**	0.2640	0.2887	0.2713	0.2664	0.2907	**0.2445**	0.2103	0.2293	0.2276	0.2169	0.2543	**0.2905**	0.2510	0.2732	0.2754	0.2586	0.2967
est_sd	**0.3547**	0.2576	0.3387	0.2680	0.2592	0.3854	**0.3502**	0.2220	0.2963	0.2346	0.2234	0.3704	**0.3704**	0.2620	0.3245	0.2695	0.2629	0.3905
cover_rate	**0.9525**	0.9500	0.9575	0.7650	0.9100	0.9625	**0.9625**	0.9625	0.9750	0.7875	0.9350	0.9600	**0.9400**	0.9650	0.9675	0.7950	0.9325	0.9425
est_sd_boot	**0.2973**	0.2643	0.2856	0.2675	0.2670	0.2976	**0.2476**	0.2114	0.2371	0.2306	0.2200	0.2598	**0.3024**	0.2534	0.2981	0.2785	0.2606	0.3024
cover_rate_boot	**0.9450**	0.9525	0.9500	0.7700	0.9125	0.9500	**0.9600**	0.9600	0.9425	0.7850	0.9325	0.9575	**0.9425**	0.9600	0.9350	0.8050	0.9300	0.9425
n=5000																		
bias	**0.0001**	0.0013	0.0033	0.2033	0.0707	0.0014	**0.0056**	0.0077	0.0076	0.2109	0.0713	0.0042	**0.0109**	0.0123	0.0117	0.2357	0.0933	0.0063
emp_sd	**0.1686**	0.1625	0.1639	0.1552	0.1638	0.1578	**0.1532**	0.1302	0.1349	0.1380	0.1367	0.1548	**0.1777**	0.1719	0.1789	0.1726	0.1801	0.1689
est_sd	**0.1442**	0.1617	0.1641	0.1640	0.1620	0.1389	**0.1473**	0.1396	0.1436	0.1424	0.1398	0.1424	**0.1537**	0.1646	0.1668	0.1655	0.1646	0.1523
cover_rate	**0.9050**	0.9575	0.9600	0.7600	0.9000	0.9100	**0.9625**	0.9725	0.9625	0.7000	0.9100	0.9650	**0.8950**	0.9450	0.9300	0.6925	0.8750	0.9125
est_sd_boot	**0.1671**	0.1635	0.1644	0.1598	0.1660	0.1582	**0.1533**	0.1300	0.1350	0.1389	0.1389	0.1539	**0.1780**	0.1720	0.1684	0.1703	0.1793	0.1682
cover_rate_boot	**0.9225**	0.9575	0.9600	0.7550	0.9125	0.9250	**0.9650**	0.9700	0.9600	0.6950	0.9225	0.9675	**0.9325**	0.9475	0.9325	0.7150	0.9025	0.9450

Table 3.12: The summary statistics of the estimated QTEs by the IPW method for Model 2 with p=10

	PS						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0184**	0.0029	0.0018	0.1538	0.0628	0.0134	**0.0033**	0.0017	0.0003	0.1098	0.0642	0.0045	0.0044	0.0023
emp_sd	**0.2037**	0.1776	0.3042	0.2421	0.2083	0.2137	**0.1781**	0.1681	0.1724	0.2277	0.2207	0.1809	0.1823	0.1690
est_sd	**0.1855**	0.1731	0.1809	0.1778	0.1834	0.1905	**0.1855**	0.1731	0.1809	0.1778	0.1834	0.1905	0.1836	0.1725
cover_rate	**0.9125**	0.9350	0.7575	0.7800	0.9025	0.9125	**0.9450**	0.9500	0.9575	0.8050	0.9000	0.9450	0.9475	0.9550
est_sd_boot	**0.2098**	0.1790	0.2265	0.1991	0.2043	0.2135	**0.1809**	0.1708	0.1780	0.2131	0.2290	0.1832		
cover_rate_boot	**0.9300**	0.9375	0.8250	0.8075	0.9100	0.9350	**0.9525**	0.9500	0.9525	0.8450	0.9150	0.9500		
n=2000														
bias	**0.0063**	0.0007	0.0001	0.1227	0.0558	0.0099	**0.0015**	0.0009	0.0010	0.0969	0.0462	0.0027	0.0065	0.0010
emp_sd	**0.1361**	0.1249	0.1492	0.1673	0.1366	0.1362	**0.1229**	0.1194	0.1220	0.1557	0.1503	0.1242	0.1287	0.1190
est_sd	**0.1244**	0.1228	0.1249	0.1261	0.1283	0.1205	**0.1244**	0.1228	0.1249	0.1261	0.1283	0.1205	0.1244	0.1225
cover_rate	**0.9000**	0.9400	0.9125	0.7525	0.9125	0.8950	**0.9550**	0.9550	0.9625	0.8325	0.8875	0.9425	0.9475	0.9575
est_sd_boot	**0.1389**	0.1255	0.1387	0.1570	0.1381	0.1335	**0.1236**	0.1205	0.1274	0.1472	0.1532	0.1239		
cover_rate_boot	**0.9350**	0.9450	0.9225	0.8475	0.9275	0.9325	**0.9550**	0.9550	0.9650	0.8850	0.9225	0.9475		
n=5000														
bias	**0.0019**	0.0011	0.0026	0.0931	0.0369	0.0014	**0.0006**	0.0010	0.0006	0.0744	0.0257	0.0003	0.0010	0.0011
emp_sd	**0.0866**	0.0710	0.0778	0.0923	0.0803	0.0897	**0.0710**	0.0697	0.0700	0.0876	0.0824	0.0716	0.0713	0.0694
est_sd	**0.0780**	0.0776	0.0780	0.0794	0.0809	0.0785	**0.0780**	0.0776	0.0780	0.0794	0.0809	0.0785	0.0782	0.0775
cover_rate	**0.9325**	0.9675	0.9525	0.7475	0.9275	0.9450	**0.9700**	0.9700	0.9700	0.8200	0.9250	0.9675	0.9700	0.9700
est_sd_boot	**0.0851**	0.0723	0.0793	0.0914	0.0815	0.0891	**0.0750**	0.0707	0.0711	0.0869	0.0841	0.0722		
cover_rate_boot	**0.9575**	0.9650	0.9650	0.7850	0.9300	0.9550	**0.9700**	0.9675	0.9675	0.8575	0.9325	0.9650		

Table 3.13: The summary statistics of the estimated ATTs for Model 2 with p=5

	PS						OR						DR	Oracle
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN		
n=1000														
bias	**0.0107**	0.0004	0.0067	0.3506	0.1334	0.0980	**0.0013**	0.0020	0.0049	0.1367	0.0679	0.0045	0.0073	0.0020
emp_sd	**0.1582**	0.1285	0.2410	0.2218	0.1641	0.1632	**0.1323**	0.1265	0.1283	0.1832	0.1635	0.1330	0.1382	0.1262
est_sd	**0.1359**	0.1293	0.1317	0.1427	0.1369	0.1360	**0.1359**	0.1293	0.1317	0.1427	0.1369	0.1360	0.1357	0.1293
cover_rate	**0.9275**	0.9575	0.7225	0.3700	0.7675	0.9250	**0.9525**	0.9550	0.9525	0.7650	0.8850	0.9525	0.9500	0.9500
est_sd_boot	**0.1543**	0.1290	0.2214	0.2098	0.1680	0.1653	**0.1339**	0.1200	0.1287	0.1627	0.1662	0.1342		
cover_rate_boot	**0.9450**	0.9575	0.8475	0.4475	0.8025	0.9475	**0.9525**	0.9575	0.9500	0.7875	0.9200	0.9525		
n=2000														
bias	**0.0043**	0.0018	0.0017	0.2885	0.1060	0.0062	**0.0031**	0.0014	0.0018	0.1005	0.0385	0.0047	0.0027	0.0016
emp_sd	**0.1081**	0.0945	0.1127	0.1454	0.1164	0.9980	**0.0937**	0.0924	0.0938	0.1237	0.1126	0.0939	0.0962	0.0922
est_sd	**0.0923**	0.0916	0.0919	0.0997	0.0943	0.0921	**0.0923**	0.0916	0.0919	0.0997	0.0943	0.0921	0.0925	0.0916
cover_rate	**0.9225**	0.9400	0.9000	0.2650	0.7550	0.9200	**0.9475**	0.9475	0.9350	0.7950	0.8975	0.9450	0.9475	0.9500
est_sd_boot	**0.1089**	0.9530	0.1098	0.1380	0.1241	0.1004	**0.0940**	0.0925	0.9420	0.1194	0.1200	0.0944		
cover_rate_boot	**0.9450**	0.9500	0.9350	0.4550	0.8350	0.9375	**0.9500**	0.9500	0.9475	0.8225	0.9175	0.9500		
n=5000														
bias	**0.0054**	0.0022	0.0017	0.2172	0.0780	0.0035	**0.0024**	0.0025	0.0023	0.0718	0.0236	0.0031	0.0023	0.0026
emp_sd	**0.0672**	0.0523	0.0550	0.0730	0.0685	0.0633	**0.0527**	0.0519	0.0521	0.0650	0.0594	0.0519	0.0543	0.0518
est_sd	**0.0581**	0.0580	0.0580	0.0622	0.0591	0.0584	**0.0581**	0.0580	0.0580	0.0622	0.0591	0.0584	0.0581	0.0580
cover_rate	**0.9350**	0.9625	0.9575	0.1000	0.7300	0.9450	**0.9650**	0.9625	0.9625	0.7800	0.9325	0.9650	0.9650	0.9650
est_sd_boot	**0.0681**	0.0531	0.0561	0.0744	0.0648	0.0645	**0.0534**	0.0533	0.0548	0.0681	0.0601	0.0526		
cover_rate_boot	**0.9525**	0.9625	0.9575	0.1150	0.7850	0.9550	**0.9625**	0.9625	0.9625	0.7925	0.9450	0.9625		

Table 3.14: The summary statistics of the estimated ATTs for Model 2 with p=10

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0118**	0.0085	0.0052	0.1674	0.1446	0.0068	**0.0119**	0.0093	0.0274	0.0916	0.0478	0.0112	**0.0170**	0.0045	0.0297	0.0051	0.0463	0.0167
emp_sd	**0.3751**	0.3662	0.4243	0.3725	0.3939	0.3751	**0.3429**	0.3248	0.3628	0.3519	0.3806	0.3522	**0.3784**	0.3618	0.3859	0.3697	0.3927	0.4612
est_sd	**0.3867**	0.3513	0.4605	0.3985	0.4366	0.3867	**0.3441**	0.2939	0.3992	0.3503	0.3858	0.3551	**0.4489**	0.3260	0.4126	0.3807	0.4136	0.4531
cover_rate	**0.9300**	0.9475	0.9625	0.9550	0.9650	0.9300	**0.9325**	0.9200	0.9475	0.9450	0.9600	0.9325	**0.9400**	0.9150	0.9600	0.9675	0.9600	0.9500
est_sd_boot	**0.3812**	0.3687	0.4431	0.3778	0.4011	0.3812	**0.3445**	0.3341	0.3891	0.3581	0.3812	0.3432	**0.3912**	0.3412	0.3992	0.3771	0.4005	0.4782
cover_rate_boot	**0.9350**	0.9500	0.9575	0.9475	0.9575	0.9350	**0.9375**	0.9350	0.9450	0.9525	0.0958	0.9300	**0.9375**	0.9250	0.9575	0.9600	0.9550	0.9575
n=2000																		
bias	**0.0005**	0.0080	0.0183	0.1430	0.0460	0.0012	**0.0148**	0.0091	0.0049	0.0673	0.0305	0.0088	**0.0092**	0.0219	0.0188	0.0267	0.0050	0.0087
emp_sd	**0.2342**	0.2427	0.2540	0.2500	0.2476	0.2384	**0.2194**	0.2134	0.2298	0.2304	0.2173	0.2214	**0.2387**	0.2445	0.2610	0.2574	0.2499	0.2276
est_sd	**0.2399**	0.2476	0.2567	0.2666	0.2509	0.2399	**0.2031**	0.2055	0.2171	0.2273	0.2091	0.2034	**0.2088**	0.2293	0.2388	0.2513	0.2340	0.2123
cover_rate	**0.9275**	0.9575	0.9550	0.9375	0.9500	0.9350	**0.9225**	0.9425	0.9450	0.9400	0.9400	0.9325	**0.9000**	0.9400	0.9425	0.9425	0.9450	0.9100
est_sd_boot	**0.2411**	0.2489	0.2578	0.2561	0.2476	0.2411	**0.2131**	0.2116	0.2309	0.2256	0.2181	0.2197	**0.2289**	0.2490	0.2541	0.2512	0.2471	0.2301
cover_rate_boot	**0.9300**	0.9600	0.9550	0.9350	0.9450	0.9375	**0.9350**	0.9475	0.9500	0.9375	0.9425	0.9350	**0.9250**	0.9525	0.9500	0.9425	0.9500	0.9275
n=5000																		
bias	**0.0053**	0.0082	0.0103	0.0952	0.0324	0.0024	**0.0112**	0.0086	0.0077	0.0312	0.0123	0.0112	**0.0060**	0.0085	0.0084	0.0085	0.0089	0.0056
emp_sd	**0.1607**	0.1662	0.1685	0.1629	0.1665	0.1656	**0.1345**	0.1377	0.1372	0.1474	0.1406	0.1402	**0.1575**	0.1573	0.1609	0.1575	0.1613	0.1602
est_sd	**0.1300**	0.1562	0.1570	0.1613	0.1568	0.1302	**0.1176**	0.1288	0.1309	0.1347	0.1297	0.1171	**0.1224**	0.1438	0.1443	0.1497	0.1451	0.1301
cover_rate	**0.8950**	0.9350	0.9400	0.9100	0.9325	0.8875	**0.9150**	0.9350	0.9425	0.9300	0.9150	0.9125	**0.8725**	0.9350	0.9400	0.9400	0.9350	0.8750
est_sd_boot	**0.1567**	0.1602	0.1659	0.1645	0.1623	0.1567	**0.1298**	0.1279	0.1321	0.1421	0.1305	0.1305	**0.1442**	0.1522	0.1611	0.1598	0.1613	0.1536
cover_rate_boot	**0.9325**	0.9425	0.9500	0.9150	0.9425	0.9325	**0.9275**	0.9350	0.9450	0.9345	0.9200	0.9250	**0.9250**	0.9425	0.9450	0.9500	0.9450	0.9175

Table 3.15: The summary statistics of the estimated QTTs by the IPW method for Model 2 with p=5

	Q1						Q2						Q3					
	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN	ANN	GLM	GAM	RF	GBM	DNN
n=1000																		
bias	**0.0072**	0.0029	0.0042	0.2291	0.0772	0.0109	**0.0508**	0.0146	0.0313	0.1348	0.0501	0.0443	**0.0125**	0.0188	0.0158	0.0932	0.0597	0.0143
emp_sd	**0.6496**	0.4152	0.5834	0.4378	0.4260	0.7649	**0.6306**	0.3256	0.4781	0.3725	0.3483	0.6896	**0.5388**	0.3874	0.5469	0.4051	0.4045	0.5788
est_sd	**0.8153**	0.3965	1.8390	0.4587	0.4094	0.9054	**0.8247**	0.3262	1.5928	0.4008	0.3414	0.8643	**0.6777**	0.3577	1.3671	0.4341	0.3742	0.7009
cover_rate	**0.9575**	0.9400	0.9900	0.9275	0.9350	0.9650	**0.9650**	0.9450	0.9950	0.9425	0.9225	0.9675	**0.9525**	0.9425	0.9900	0.9600	0.9500	0.9625
est_sd_boot	**0.6874**	0.4215	0.5911	0.4430	0.4361	0.7751	**0.6813**	0.3327	0.6513	0.3888	0.3611	0.7013	**0.5823**	0.3851	0.5821	0.4275	0.3998	0.5961
cover_rate_boot	**0.9500**	0.9450	0.9850	0.9250	0.9400	0.9550	**0.9600**	0.9550	0.9775	0.9400	0.9350	0.9650	**0.9500**	0.9525	0.9800	0.9475	0.9550	0.9525
n=2000																		
bias	**0.0267**	0.0200	0.0199	0.2048	0.0834	0.0289	**0.0083**	0.0122	0.0190	0.1109	0.0443	0.0153	**0.0085**	0.0030	0.0109	0.0628	0.0393	0.0082
emp_sd	**0.3074**	0.2794	0.3069	0.2874	0.2804	0.3154	**0.2827**	0.2372	0.2666	0.2587	0.2403	0.2809	**0.2957**	0.2635	0.2887	0.2814	0.2702	0.3075
est_sd	**0.3391**	0.2780	0.3515	0.3035	0.2814	0.3489	**0.3102**	0.2272	0.2975	0.2584	0.2322	0.3074	**0.2972**	0.2507	0.3127	0.2837	0.2572	0.3019
cover_rate	**0.9450**	0.9550	0.9700	0.9200	0.9475	0.9425	**0.9600**	0.9500	0.9525	0.9325	0.9400	0.9575	**0.9575**	0.9575	0.9475	0.9450	0.9350	0.9475
est_sd_boot	**0.3110**	0.2799	0.3087	0.2975	0.2785	0.3165	**0.2904**	0.2389	0.2705	0.2598	0.2337	0.3003	**0.3010**	0.2511	0.3098	0.2950	0.2681	0.3029
cover_rate_boot	**0.9425**	0.9550	0.9600	0.9150	0.9400	0.9400	**0.9525**	0.9525	0.9475	0.9300	0.9400	0.9550	**0.9575**	0.9575	0.9450	0.9450	0.9375	0.9550
n=5000																		
bias	**0.0101**	0.0085	0.0087	0.1227	0.0279	0.0087	**0.0022**	0.0020	0.0062	0.0910	0.0397	0.0053	**0.0155**	0.0156	0.0202	0.0641	0.0509	0.0093
emp_sd	**0.1850**	0.1773	0.1832	0.1702	0.1781	0.1920	**0.1622**	0.1462	0.1529	0.1588	0.1496	0.1625	**0.1774**	0.1730	0.1839	0.1794	0.1793	0.1731
est_sd	**0.1707**	0.1738	0.1784	0.1800	0.1740	0.1736	**0.1468**	0.1424	0.1484	0.1509	0.1433	0.1534	**0.1463**	0.1572	0.1617	0.1667	0.1590	0.1592
cover_rate	**0.8825**	0.9500	0.9375	0.9150	0.9250	0.8850	**0.9125**	0.9425	0.9450	0.8950	0.9275	0.9250	**0.8700**	0.9250	0.9175	0.9125	0.9125	0.8850
est_sd_boot	**0.1884**	0.1778	0.1853	0.1794	0.1799	0.1904	**0.1587**	0.1478	0.1503	0.1589	0.1487	0.1624	**0.1647**	0.1789	0.1800	0.1821	0.1740	0.1672
cover_rate_boot	**0.9250**	0.9525	0.9400	0.9150	0.9275	0.9225	**0.9375**	0.9450	0.9475	0.8925	0.9275	0.9400	**0.9125**	0.9300	0.9225	0.9175	0.9200	0.9150

Table 3.16: The summary statistics of the estimated QTTs by the IPW method for Model 2 with p=10

		ATE		ATT	
		ANN	ANN-PCA	ANN	ANN - PCA
IPW	bias	0.0687	0.0715	0.1516	0.1323
	emp_sd	0.1464	0.1084	0.2025	0.1534
	est_sd	0.0613	0.0697	0.0561	0.0807
	cover_rate	0.5450	0.6925	0.3050	0.4825
	est_sd_boot	0.1512	0.1121	0.1782	0.1423
	cover_rate_boot	0.5725	0.7050	0.5425	0.6875
OR	bias	0.0554	0.0672	0.0682	0.1092
	emp_sd	0.1279	0.1035	0.1824	0.1478
	est_sd	0.0613	0.0697	0.0561	0.0807
	cover_rate	0.6050	0.7225	0.4675	0.6175
	est_sd_boot	0.1321	0.1093	0.1625	0.1372
	cover_rate_boot	0.6225	0.7850	0.7850	0.7425
DR	bias	0.0599	0.0710	0.0685	0.1103
	emp_sd	0.1303	0.1076	0.1892	0.1492
	est_sd	0.0618	0.0693	0.0572	0.0821
	cover_rate	0.5975	0.7175	0.4650	0.6200

Table 3.17: The summary statistics of the estimated ATEs and ATTs for Model 1 with p=100 and n=2000

		QTE		QTT	
		ANN	ANN-PCA	ANN	ANN - PCA
Q1	bias	0.0394	0.0500	0.1205	0.1007
	emp_sd	0.1775	0.1374	0.2496	0.1907
	est_sd_boot	0.1657	0.1423	0.2756	0.2305
	cover_rate_boot	0.9425	0.9475	0.8450	0.8750
Q2	bias	0.0081	0.0139	0.0561	0.0626
	emp_sd	0.1446	0.1174	0.1910	0.1480
	est_sd_boot	0.1423	0.1231	0.1832	0.1325
	cover_rate_boot	0.9500	0.9525	0.9050	0.9075
Q3	bias	0.0121	0.0057	0.0238	0.0366
	emp_sd	0.1280	0.1074	0.1696	0.1310
	est_sd_boot	0.1267	0.1123	0.1778	0.1442
	cover_rate_boot	0.9600	0.9650	0.9650	0.9525

Table 3.18: The summary statistics of the estimated QTEs and QTTs by the IPW method for Model 1 with p=100 and n=2000.

3.7 Application

In this section, we apply the proposed methods to the data from the National Health and Nutrition Examination Survey (NHANES) to investigate the causal effect of smoking on body mass index (BMI). The collected data consist of 6647 subjects, including 3359 smokers and 3288 nonsmokers. The confounding variables include four continuous variables: age, family poverty income ratio (Family PIR), systolic blood pressure (SBP), and diastolic blood pressure (DBP); six binary variables: gender, marital status, education, alcohol use, vigorous activity over past 30 days (PHSVIG), and moderate activity over past 30 days (PHSMOD). Table 3.19 presents the group comparisons of all confounding variables in the full dataset. Mean and standard deviation (SD) are presented for continuous variables, while the count and percentage (%) of observations for each group are presented for categorical variables. Standardized difference(Std. Dif.) is calculated as $(\bar{x}_{ns} - \bar{x}_s)/\sqrt{s_{ns}^2/n_{ns} + s_s^2/n_s}$ for continuous variables, and $(p_{ns} - p_s)/\sqrt{pq/n_{ns} + pq/n_s}$ for categorical variables, where $\bar{x}$, s^2 and p denote sample mean, sample variance and sample proportion, and the subscripts ns and s refer to nonsmokers and smokers respectively, and p, q are the overall proportions. The last column shows the p-value of group comparison for each covariate. We notice that smoking group and nonsmoking group differ greatly in their group characteristics. A naive comparison of the sample mean between smoking and nonsmoking groups will lead to a biased estimation of the smoking effects on BMI.

We apply our proposed ANN methods to estimate the PS and OR functions, respectively. We estimate ATE by the proposed IPW and OR methods, and estimate QTE by the IPW method only. The number of neurons are selected using grid search with 5-fold

Covariates		Non-smoker (N_{ns}=3288)		Smoker (N_{ns}=3359)		Std. Dif.	p-value
Gender	1 = Male	1404	(41.8%)	2019	(61.41%)	-15.99	<0.001
	0 = Female	1955	(58.2%)	1269	(38.59%)		
Age	Mean(SD)	48.97	(19)	51.73	(17.57)	-6.14	<0.001
Marital	1 = Yes	1989	(59.21%)	1867	(56.78%)	2.01	0.0446
	0 = No	1370	(40.79%)	1421	(43.22%)		
Education	1 = College or above	1626	(48.41%)	1297	(39.45%)	7.36	<0.001
	0 = Less than college	1733	(51.59%)	1991	(60.55%)		
Family PIR	Mean(SD)	2.79	(1.63)	2.57	(1.6)	5.62	<0.001
Alcohol	1 = Yes	1897	(56.48%)	2708	(82.36%)	-22.87	<0.001
	0 = No	1462	(43.52%)	580	(17.64%)		
PHSVIG	1 = Yes	1102	(32.81%)	908	(27.62%)	4.61	<0.001
	0 = No	2257	(67.19%)	2380	(72.38%)		
PHSMOD	1 = Yes	1491	(44.39%)	1376	(41.85%)	2.09	0.0366
	0 = No	1868	(55.61%)	1912	(58.15%)		
SBP	Mean(SD)	126.42	(21.04)	126.63	(19.98)	-0.43	0.6684
DBP	Mean(SD)	72.1	(13.56)	71.61	(14.1)	1.44	0.15

Table 3.19: Group comparisons

cross-validation. Table 3.20 reports the estimates of ATE and QTE, the estimated standard deviations based on the asymptotic formula (est_sd) and obtained from the weighted bootstrapping (est_sd_boot), and the corresponding z-values and p-values for testing ATE and QTE. The negative values of the estimates indicate that smoking has adverse effects on BMI. From the numerical results based on the estimated asymptotic standard deviations, we see that the p-values of testing ATE are 0.073 and 0.058 by the IPW and OR methods, respectively. We also notice that the p-value for testing QTE at the 25% quantile is very small, which is 0.005. However, the p-value increases to 0.071 at the 50% quantile (median), and further to 0.436 at the 75% quantile. This indicates that smoking has a more prominent effect on the population with smaller BMI, and its effect diminishes as BMI increases; i.e., the effect of smoking becomes less significant as the value of BMI becomes larger.

	ATE		QTE		
	IPW	OR	Q1	Q2	Q3
estimate	-0.224	-0.241	-0.400	-0.269	-0.040
est_sd	0.154	0.154	0.157	0.184	0.247
z-value	-1.454	-1.564	-2.547	-1.467	-0.162
p-value	0.073	0.058	0.005	0.071	0.436
est_sd_boot	0.162	0.149	0.156	0.187	0.254
z-value_boot	-1.383	-1.617	-2.564	-1.443	-0.157
p-value_boot	0.083	0.053	0.005	0.074	0.437

Table 3.20: The estimates and standard errors of ATE and QTE.

This interesting pattern cannot be reflected from ATE. We can draw the same inferential conclusions as above when the weighted bootstrap method is applied.

We also examine the relationship between BMI and two continuous confounding variables, age and family poverty income ratio (Family PIR). Figure 3.2 depicts the estimated conditional mean functions (OR functions) $\tau_1(\cdot)$ and $\tau_0(\cdot)$ versus the two continuous variables for the smoking and nonsmoking groups, and for males and females, respectively. For each comparison, all the other confounding variables are fixed as constants: the continuous variables take the values of their means while the categorical variables are kept as married, college or above, drinks alcohol, no vigorous activity and no moderate activity. It is interesting to notice that for the same age or Family PIR, the estimated conditional mean in the smoking group is smaller than that in the nonsmoking group for both male and female, and the estimated conditional mean in the male group is also smaller than that in the female group for both smoker and nonsmoker. We can clearly see nonlinear relationships

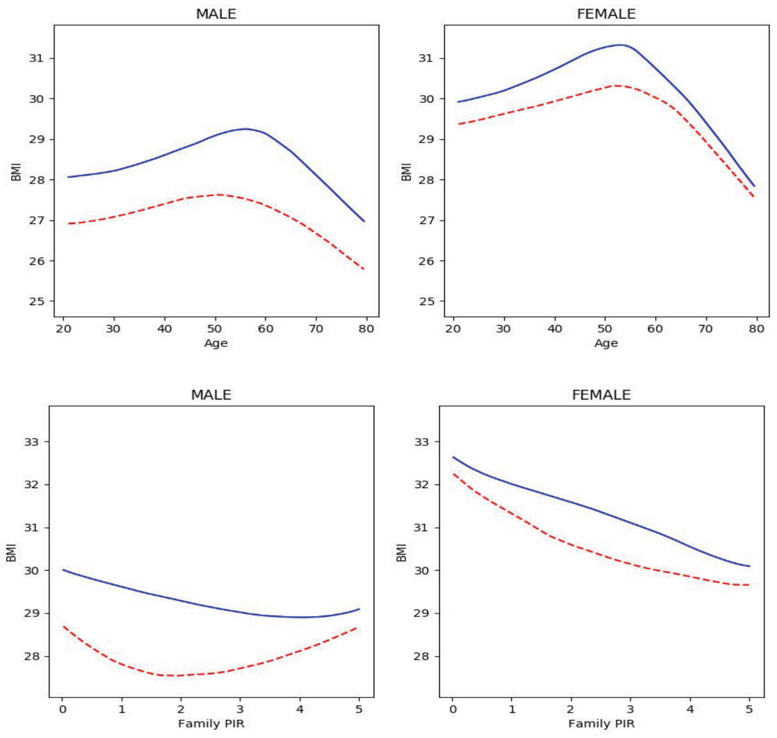


Figure 3.2: The plots of $\tau_1(\cdot)$ and $\tau_0(\cdot)$ versus two continuous variables for the smoking and nonsmoking groups, and for males and females, respectively, where the blue solid curves represent nonsmoking group and red dashed line represent smoking group.

between age and BMI as well as between Family PIR and BMI. Age is positively associated with BMI when it is less than 50, and the association between age and BMI becomes more negative as people get older. We also see that the smoking effects on BMI are very different between the male group and female group. Smoking has more significant effect on BMI for male than for female at the same age. In the male group, the BMI decreases as family income increases until it reaches the poverty threshold, and then the BMI increases with family income for smokers. For nonsmokers, it shows a relatively flatter trend. In the female group, the BMI keeps decreasing as family income increases for both smokers and nonsmokers.

3.8 Discussion

In this chapter, we provide a unified framework for efficient estimation of various types of TEs in observational data with a diverging number of covariates. The framework can be applied to the settings with binary or multi-valued treatment variables, and it includes the average, quantile and asymmetric least squares TEs as special cases. We propose to estimate the TEs through a generalized optimization. The resulting TE estimator only involves the estimate of one nuisance function, which is approximated by ANNs with one hidden layer. In contrast, for other existing related works that use machine learning, they construct the TE estimator based on its efficient influence function, so that the estimator can have desirable theoretical properties for conducting causal inference. However, this method loses generalizations as one has to work out the influence function first for each type of TE. Other than ATE, estimation of the influence function for different types of TEs can be a difficult undertaking. Moreover, since our TE estimator is obtained directly through a generalized optimization, it is convenient to apply a weighted bootstrap method for conducting inference without the need of estimating the asymptotic variance. Theoretically, we show that the number of confounders is allowed to increase with the sample size, and further investigate how fast it can grow with the sample size to ensure root-n consistency of the resulting TE estimator, when the nuisance function is approximated by ANNs with one hidden layer. Moreover, we establish asymptotic normality and semiparametric efficiency of the TE estimator. These statistical properties are essential for inferring causations. Practically, we illustrate the proposed method through simulation studies and a real data example. The numerical studies support our theoretical findings.

We have shown that the ANNs with one hidden layer can circumvent the "curse of dimensionality" and the resulting TE estimators enjoy root-n consistency under the condition that the target function is in a mixed Sobolev space. Our new results advance the understanding of the required conditions and the statistical properties for ANNs in causal inference, and lay a theoretical foundation to demonstrate that ANNs are promising tools for causality analysis when the dimension is allowed to diverge, whereas most existing works on ANNs estimation still assume the dimension to be fixed. In the online Supplemental Materials [33], we discuss the extension of our proposed method for TE estimation when the nuisance function is approximated by fully-connected ANNs with multiple hidden layers and its statistical properties. We show that when the target function belongs to the regular Sobolev space [125], the fully-connected deep ANNs has the desirable statistical properties for TE estimation in the data setting with a fixed number of covariates, whereas it enjoys narrower width than the single hidden layer ANNs when its depth grows with the sample size. The investigation of the deep ANNs in the scenarios with a diverging dimension is beyond the scope of this dissertation and can be a future research topic. We can also consider sparse deep ANNs to overcome the dimensionality issue of the fully-connected ones, and will investigate the statistical properties and the required conditions such as the function space in which the target function needs to belong in this framework. These interesting yet challenging technical problems deserve further studies. Moreover, the proposed method can be extended to causal analysis with continuous treatment variables and with longitudinal data designs. Thorough investigations are needed to develop the computational algorithms and establish the theoretical properties of the resulting estimators in these settings.

Chapter 4

Sparse Deep Neural Networks Regression and Functional Derivative Estimation

4.1 Introduction

In recent years, deep neural networks with multiple hidden layers have been shown to be powerful and effective for approximating multivariate functions, and have been successfully applied to many fields [61, 73, 51]. However, there is not enough studies for the theoretical properties of deep neural networks. It is known that using only one hidden layer, ANN can provide an optimal order of approximation for functions which satisfy certain smoothness conditions [35, 89]. A common choice of activation function in neural networks is ReLU function, and ReLU has been shown to have computational advantage over the sigmoid functions used mainly in shallow networks [39]. [125] proved that deep ReLU networks have better approximation than shallow ReLU networks. One difficulty in analyzing deep neural networks is fully connected deep neural networks suffer from "curse of dimensionality". As the need of analyzing high-dimensional data is growing, sparse grids approximation is gaining more popularity as an approximation tool [52, 25, 104]. An upper bound of sparse deep ReLU network approximation error of high-dimensional functions with bounded mixed derivatives is obtained in [90]. The connection between sparse grids and deep neural networks has also extended to neural network with other type activation function. [77] shows that deep neural networks using Rectified Power Units (RePUs), as activation functions have better approximation property for smooth functions than those using ReLUs. Following these research, in this dissertation we propose estimating target function upon a network architecture of sparsely-connected deep neural networks with the Rectified Quadratic Unit (ReQU) activation function, through an empirical risk minimization framework and apply regularization to prevent possible over-fitting.

Another good property of ReQU networks is that we can obtain functional derivatives for our estimates. While ANN is widely applied in regression problems, derivative estimation has received less attention. Derivative estimation is essential in many situations including comparing regression curves [97], analyzing significant trends [99] and identifying change points in longitudinal data [107]. Though we can obtain functional derivatives directly from parametric linear models, such models are suffering from model mis-specification and result in large bias in the derivatives, especially when the underlying function structure is complicated. For certain nonparametric methods and machine learning methods applied in regression problems, there is no analytical form solution to the estimated regression function and considered as "black boxes", thus we can't estimate the functional derivatives. Though we can't obtain the functional derivatives directly from such regression model, we can still use some other methods to estimate the derivatives. For instance, local polynomial regression method in [44], b-spline regression in [83]. There are also some new approaches using machine learning methods, such as Boosted Smooth Transition Regression Trees (BooST) for estimating partial effect [48], weighted difference quotients for nonparametric derivative estimation [80]. However, it would be ideal if we could obtain functional derivatives. Our proposed method is a remedy to smooth functional derivative estimation since the estimators are all represented by smooth functions.

This chapter is organized as follows. Section 4.2 provides the basic setup. Section 4.3.1 discusses approximation of the target function by the ReQU networks. Section 4.4 introduces the sparse deep ReQU network estimator obtained from empirical risk minimization. Section 4.5 reports results from simulation studies, and Section 4.6 illustrates the proposed method through real data applications. Some concluding remarks are given in Section 4.7.

4.2 Basic Setup

Notations: Denote $||\boldsymbol{a}||_p = (\sum_{i=1}^{m} |a_i|^p)^{1/p}$ as the L^p-norm of a vector $\boldsymbol{a}$, and $\boldsymbol{a}$= $(a_1, \ldots, a_m)^\top$, $|\boldsymbol{a}|_\infty = \max_{i=1,\ldots,m} |a_i|$. For two vectors $\boldsymbol{a} = (a_1, \ldots, a_m)^\top$ and $\boldsymbol{b} = (b_1, \ldots, b_m)^\top$, denote $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{m} a_i b_i$. Moreover, for any arithmetic operations involving vectors, they are performed element-by-element. For any two values a and b, denote $a \vee$

$b = \max(a,b)$. For two sequences of positive numbers a_n and b_n, $a_n \ll b_n$ means that $b_n^{-1}a_n = o(1)$, $a_n \lesssim b_n$ means that there exists a constant $C \in (0,\infty)$ such that $a_n \leq Cb_n$, and $a_n \asymp b_n$ means that there exist constants $C, C' \in (0,\infty)$ such that $a_n \leq Cb_n$ and $b_n \leq C'a_n$. Let $|A|$ be the cardinality of a set A. Denote by $\mathbb{R}$ (or $\mathbb{N}$) the set of all real numbers (or positive integers), and let $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$.

We consider the generalized regression problem. Let $Y \in \mathcal{Y} \subset \mathbb{R}$ be a real-valued response variable and $\boldsymbol{X} = (X_1, \ldots, X_d)^\top$ be d-dimensional independent variables with values in a compact support $\mathcal{X} \subset \mathbb{R}^d$. Without loss of generality, we let $\mathcal{X} = (-1,1)^d$. Let $(\boldsymbol{X}_i^\top, Y_i)^\top$, $i = 1, ..., n$ be i.i.d. samples drawn from the distribution of $(\boldsymbol{X}^\top, Y)^\top$.

let $\mu : \mathcal{X} \times \mathbb{R} \to [0,1]$ be a Borel probability measure of $(\boldsymbol{X}^\top, Y)^\top$. For every $\boldsymbol{x} \in \mathcal{X}$, let $\mu(y|\boldsymbol{x})$ be the conditional (w.r.t. $\boldsymbol{x}$) probability measure of Y. Let μ_X be the marginal probability measure of $\boldsymbol{X}$. Let $\mathcal{L}^2_\omega(\mathcal{X}) = \{f : \mathcal{X} \to \mathbb{R},\ f$ is Lebesgue measurable on $\mathcal{X}^d$ and $||f||_{L^2_\omega} < \infty\}$, where $||f||_{L^2_\omega} = (\int_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})^2 \omega(\boldsymbol{x})\, d\mu_X(\boldsymbol{x}))^{1/2}$ is the weighted L^2 norm with the weighting function $\omega(\cdot)$. Denote $||f||_\infty = \sup_{\boldsymbol{x}\in\mathcal{X}^d} |f(\boldsymbol{x})|$. For the weighted L^2 norm, we write $rB_{L^2_\omega} = \{f \in \mathcal{L}^2_\omega(\mathcal{X}) \colon ||f||_{L^2_\omega} \leq r\}$.

Define $g_0 : \mathcal{X} \to \mathbb{R}$ by

$$g_0(\boldsymbol{x}) = \int_{\mathcal{Y}} y d\mu(y|\boldsymbol{x}).$$

The function g_0 is the regression function or the conditional mean function of the response Y given $\boldsymbol{X} = \boldsymbol{x}$. Our goal is to estimate the unknown function $g_0(\boldsymbol{x})$ using deep neural networks from the observed data.

The partial derivatives of f with mult-index $\boldsymbol{k} = (k_1, ..., k_d)^\top \in \mathbb{N}^d$ is given as

$$D^{\boldsymbol{k}} f = \frac{\partial^{|\boldsymbol{k}|_1} f}{\partial x_1^{k_1} \cdots \partial x_d^{k_d}},$$

where $\mathbb{N} = \{0, 1, 2, ...,\}$ and $|\boldsymbol{k}|_1 = \sum_{j=1}^{d} k_j$. Denote $||f||_{\boldsymbol{k},\infty} = \sup_{\boldsymbol{x}\in\mathcal{X}} |D^{\boldsymbol{k}} f(\boldsymbol{x})|$. We will also obtain the partial derivatives of $g_0(\boldsymbol{x})$.

4.3 Approximation of the Target Function

4.3.1 Jacobi Polynomial Approximation

Jacobi polynomials

We will first discuss the Jacobi polynomials for $d = 1$ and will extend them to the multivariate case with $d > 1$. The Jacobi polynomials can be used to approximate a smooth function and its derivatives. Let $\omega^{\alpha,\beta}(x) = (1-x)^\alpha (1+x)^\beta$ be the Jacobi weight function over $(-1, 1)$. Note that $\int_{-1}^{1} \omega^{\alpha,\beta}(x)\, dx < \infty$ for any $\alpha, \beta > -1$. We assume $\alpha, \beta \geq 0$ throughout the chapter. The one-dimensional Jacobi polynomials, denoted by $J_r^{\alpha,\beta}(x)$, are the eigenfunctions of the Sturm-Liouville problem [104]:

$$-\frac{1}{\omega^{\alpha,\beta}(x)} \partial_x \left(\omega^{\alpha+1,\beta+1}(x)\, \partial_x J_r^{\alpha,\beta}(x) \right) = \lambda_r^{\alpha,\beta} J_r^{\alpha,\beta}(x),$$

with the corresponding eigenvalues $\lambda_r^{\alpha,\beta} = r(r+\alpha+\beta+1)$. The family of normalized Jacobi polynomials forms orthogonal system in $L^2_{\omega^{\alpha,\beta}}(\mathcal{X})$ such that

$$\int_{-1}^{1} J_r^{\alpha,\beta}(x)\, J_{r'}^{\alpha,\beta}(x)\, \omega^{\alpha,\beta}(x)\, dx = \delta_{rr'}, \tag{4.1}$$

where $\delta_{rr'} = 1$ if $r = r'$ and $\delta_{rr'} = 0$ otherwise. It is given in [103] that the normalized Jacobi polynomials have the function form given as

$$J_r^{\alpha,\beta}(x) = \sum\nolimits_{k=0}^{r} a_k^r (x-1)^k, \tag{4.2}$$

where a_k^r are constants given in [103]. Moreover,

$$\partial_x J_r^{\alpha,\beta}(x) = \sqrt{\lambda_r^{\alpha,\beta}} J_{r-1}^{\alpha+1,\beta+1}(x), \text{ for } r \geq 1 \tag{4.3}$$

so that the derivatives $\partial_x J_r^{\alpha,\beta}(x) := \frac{\partial J_r^{\alpha,\beta}(x)}{\partial x}$ are also orthogonal with respect to the weight $\omega^{\alpha+1,\beta+1}(x)$ such that

$$\begin{aligned} & \int_{-1}^{1} \partial_x J_r^{\alpha,\beta}(x)\, \partial_x J_{r'}^{\alpha,\beta}(x)\, \omega^{\alpha+1,\beta+1}(x)\, dx \\ = & \int_{-1}^{1} \sqrt{\lambda_r^{\alpha,\beta}} J_{r-1}^{\alpha+1,\beta+1}(x) \sqrt{\lambda_{r'}^{\alpha,\beta}} J_{r'-1}^{\alpha+1,\beta+1}(x)\, \omega^{\alpha+1,\beta+1}(x)\, dx \\ = & \sqrt{\lambda_r^{\alpha,\beta}} \sqrt{\lambda_{r'}^{\alpha,\beta}} \delta_{rr'}. \end{aligned} \tag{4.4}$$

Next, for $\boldsymbol{x} \in \mathcal{X}^d$, we define the d-dimensional tensor Jacobi polynomial and Jacobi weight function as

$$\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) = \prod_{j=1}^{d} J_{r_j}^{\alpha_j,\beta_j}(x_j), \boldsymbol{\omega}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) = \prod_{j=1}^{d} \omega^{\alpha_j,\beta_j}(x_j)$$

where $\boldsymbol{r} = (r_1, ..., r_d)^\top$, $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_d)^\top$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_d)^\top$, and $\alpha_j, \beta_j > -1$ for all $1 \leq j \leq d$. From (4.2), we have

$$\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) = \prod_{j=1}^{d} \sum_{k_j=0}^{r_j} a_{k_j}^{r_j} (x_j - 1)^{k_j} = \sum_{k_1,...,k_d=0}^{r_1,...,r_d} \left\{ \prod_{j=1}^{d} a_{k_j}^{r_j} (x_j - 1)^{k_j} \right\}.$$

Then Equation (4.1) leads to

$$\int_{\mathcal{X}} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, \boldsymbol{J}_{\boldsymbol{r}'}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, \boldsymbol{\omega}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x} = \boldsymbol{\delta}_{\boldsymbol{r}\boldsymbol{r}'},$$

where $\boldsymbol{\delta}_{\boldsymbol{r}\boldsymbol{r}'} = \prod_{j=1}^{d} \delta_{r_j r_j'}$ and $\boldsymbol{\delta}_{\boldsymbol{r}\boldsymbol{r}'} = 1$ if $\boldsymbol{r} = \boldsymbol{r}'$, $\boldsymbol{\delta}_{\boldsymbol{r}\boldsymbol{r}'} = 0$ otherwise. For any $1 \leq j \leq d$, let $\partial_{x_j} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) := \frac{\partial \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})}{\partial x}$. Denote $\widetilde{\boldsymbol{\omega}}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) = \prod_{j' \neq j} \omega^{\alpha_{j'},\beta_{j'}}\left(x_{j'}\right) \omega^{\alpha_j+1,\beta_j+1}(x_j)$. Then (4.1) and (4.4) imply that for $1 \leq j \leq d$,

$$\int_{\mathcal{X}} \partial_{x_j} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, \partial_{x_j} \boldsymbol{J}_{\boldsymbol{r}'}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, \widetilde{\boldsymbol{\omega}}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x} = \sqrt{\lambda_{r_j}^{\alpha_j,\beta_j}} \sqrt{\lambda_{r_j'}^{\alpha_j,\beta_j}} \boldsymbol{\delta}_{\boldsymbol{r}\boldsymbol{r}'}.$$

For any function $g \in \mathcal{L}^2_{\boldsymbol{\omega}\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\mathcal{X})$, it can be written as

$$g(\boldsymbol{x}) = \sum\nolimits_{\boldsymbol{r}\geq 0} \widetilde{\eta}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}), \text{ where } \widetilde{\eta}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}} = \int_{\mathcal{X}} g(\boldsymbol{x}) \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) \boldsymbol{\omega}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x}.$$

For each $J_{r_j}^{\alpha_j,\beta_j}(x_j), 1 \leq j \leq d$, its derivative with respect to x_j can be represented as

$$\partial_x^{k_j} J_{r_j}^{\alpha_j,\beta_j}(x_j) = d_{r_j,k_j}^{\alpha_j,\beta_j} J_{r_j-k_j}^{\alpha_j+k_j,\beta_j+k_j}(x_j). \tag{4.5}$$

where $d_{r_j,k_j}^{\alpha_j,\beta_j} = \frac{\Gamma(r_j+k_j+\alpha_j+\beta_j+1)}{2^{k_j}(r_j+\alpha_j+\beta_j+1)}$ is a constant ([103]). There exists an internal relation between the Jacobi polynomials and their derivatives. For the d-dimensional tensor Jacobi polynomial, is partial derivative with mult-index $\boldsymbol{k} = (k_1, ..., k_d)^\top \in \mathbb{N}^d$ is

$$D^{\boldsymbol{k}} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) = \frac{\partial^{|\boldsymbol{k}|_1} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})}{\partial x_1^{k_1} \cdots \partial x_d^{k_d}} = \prod_{j=1}^{d} d_{r_j,k_j}^{\alpha_j,\beta_j} J_{r_j-k_j}^{\alpha_j+k_j,\beta_j+k_j}(x_j). \tag{4.6}$$

Jacobi Approximation on the Sparse Grid

Let $\Upsilon_m \subset \mathbb{N}_0^d$ be an index set that will be specified later. We introduce the Jacobi polynomial space:

$$V_m^{\boldsymbol{\alpha},\boldsymbol{\beta}} := \operatorname{span}\{\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}} : \boldsymbol{r} \in \Upsilon_m\}.$$

Let $\widetilde{g}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})$ be the projection of the function $g \in \mathcal{L}^2_{\boldsymbol{\omega}\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\mathcal{X})$ onto the space $V_m^{\boldsymbol{\alpha},\boldsymbol{\beta}}$, and it satisfies

$$\int_{\mathcal{X}} (\widetilde{g}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) - g(\boldsymbol{x})) \boldsymbol{\omega}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x} = 0$$

for all $v(\boldsymbol{x}) \in V_m^{\boldsymbol{\alpha},\boldsymbol{\beta}}$, so $\widetilde{g}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})$ can be written as

$$\widetilde{g}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) = \sum\nolimits_{\boldsymbol{r}\in\Upsilon_m} \widetilde{\eta}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}).$$

If we consider the d-dimensional Jacobi polynomial space on the full grid with the index set $\Upsilon_m^F = \{\boldsymbol{r} \in \mathbb{N}_0^d : |\boldsymbol{a}|_\infty \leq m\}$. In this case, the cardinality of the Jacobi space

$V_m^{\boldsymbol{\alpha},\boldsymbol{\beta}}$ is $|V_m^{\boldsymbol{\alpha},\boldsymbol{\beta}}| = (m+1)^d$, which increases with d quickly. For the purpose of dimension reduction, we consider the space on the hyperbolic cross index set:

$$\Upsilon_{m,v} = \{\boldsymbol{r} \in \mathbb{N}_0^d : 1 \leq |\boldsymbol{r}|_{\text{mix}} |\boldsymbol{r}|_\infty^{-\nu} \leq m^{1-v}\}, -\infty \leq v < 1, \tag{4.7}$$

where $|\boldsymbol{r}|_{\text{mix}} = \prod_{j=1}^d \max\{1, r_j\}$ and $|\boldsymbol{r}|_\infty = \max\{r_j, 1 \leq j \leq d\}$. When $0 < v < 1$, there is a trade-off between m and $|\boldsymbol{r}|_\infty$ resulting in a dimensionality reduction. The corresponding Jacobi polynomial space is

$$V_{m,v}^{\boldsymbol{\alpha},\boldsymbol{\beta}} := \text{span}\{\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}} : \boldsymbol{r} \in \Upsilon_{m,v}\}.$$

Table 4.2 provides the number of basis functions for the Jacobi polynomial space with sparse grids. We see that the number of basis functions for the space with sparse grids ($0 < v < 1$) is dramatically reduced compared to the space with full grids ($v = -\infty$). As v increases, the dimensionality reduction effect is stronger.

4.3.2 Approximation of Polynomial Functions by Deep ReQU Networks

In this Chapter, we consider deep neural networks using rectified quadratic unites (ReQUs). The ReQU function is defined as

$$\sigma_2(x) = x^2 I(x \geq 0),$$

where $I(x \geq 0)$ is the indicator function such that $I(x \geq 0) = 1$ if $x \geq 0$ and $I(x \geq 0) = 0$ otherwise. Below we introduce a feedforward ReQU network structure. The feedforward neural networks move in one direction from the input variables of dimension d, through the hidden nodes, and to the ouput node. Then, the feedforward ReQU network with L hidden

v		N=1	N=2	N=3	N=4	N=5	N=6	N=7	N=8
$-\infty$	d=1	2	3	4	5	6	7	8	9
	d=2	4	9	16	25	36	49	64	81
	d=3	8	27	64	125	216	343	512	729
	d=4	16	81	256	625	1296	2401	4096	6561
	d=5	32	243	1024	3125	7776	16807	32768	59049
	d=6	64	729	4096	15625	46656	117649	262144	531441
	d=7	128	2187	16384	78125	279936	823543	2097152	4782969
	d=8	256	6561	65536	390625	1679616	5764801	16777216	43046721
	d=9	512	19683	262144	1953125	10077696	40353607	134217728	387420489
	d=10	1024	59049	1048576	9765625	60466176	282475249	1073741824	3486784401
0.3	d=1	2	3	4	5	6	7	8	9
	d=2	4	8	12	16	20	25	29	33
	d=3	8	20	32	44	56	74	86	98
	d=4	16	48	80	112	144	200	232	264
	d=5	32	112	192	272	352	512	592	672
	d=6	64	256	448	640	832	1264	1456	1648
	d=7	128	576	1024	1472	1920	3040	3488	3936
	d=8	256	1280	2304	3328	4352	7168	8192	9216
	d=9	512	2816	5120	7424	9728	16640	18944	21248
	d=10	1024	6144	11264	16384	21504	38144	43264	48384
0.5	d=1	2	3	4	5	6	7	8	9
	d=2	4	8	12	16	20	24	28	33
	d=3	8	20	32	44	56	68	80	98
	d=4	16	48	80	112	144	176	208	264
	d=5	32	112	192	272	352	432	512	672
	d=6	64	256	448	640	832	1024	1216	1648
	d=7	128	576	1024	1472	1920	2368	2816	3936
	d=8	256	1280	2304	3328	4352	5376	6400	9216
	d=9	512	2816	5120	7424	9728	12032	14336	21248
	d=10	1024	6144	11264	16384	21504	26624	31744	48384
0.7	d=1	2	3	4	5	6	7	8	9
	d=2	4	8	12	16	20	24	28	32
	d=3	8	20	32	44	56	68	80	92
	d=4	16	48	80	112	144	176	208	240
	d=5	32	112	192	272	352	432	512	592
	d=6	64	256	448	640	832	1024	1216	1408
	d=7	128	576	1024	1472	1920	2368	2816	3264
	d=8	256	1280	2304	3328	4352	5376	6400	7424
	d=9	512	2816	5120	7424	9728	12032	14336	16640
	d=10	1024	6144	11264	16384	21504	26624	31744	36864

Table 4.2: The number of basis functions for the space with different v.

layers is given as

$$\Phi(\cdot;\eta):\mathbb{R}^d\to\mathbb{R}^{N_{L+1}};\ \Phi(\boldsymbol{x};\eta)=\boldsymbol{z}_{L+1},$$

where $\Phi(\boldsymbol{x};\eta)$ is defined as $\boldsymbol{z}_0=\boldsymbol{x}$, $\boldsymbol{z}_\ell=\sigma_2(\boldsymbol{A}_\ell\boldsymbol{z}_{\ell-1}+\boldsymbol{b}_\ell)$ for $\ell=1,...,L$, and $\boldsymbol{z}_{L+1}=\boldsymbol{A}_{L+1}\boldsymbol{z}_L+\boldsymbol{b}_L$, where $\boldsymbol{A}_\ell$ are $N_\ell\times N_{\ell-1}$ matrices and $\boldsymbol{b}_\ell\in\mathbb{R}^{N_\ell}$ with $N_0=d$ and $N_\ell\in\mathbb{N}$ for $\ell=1,...,L+1$. Moreover, for any vector $\boldsymbol{v}=(v_1,...,v_N)^\top\in\mathbb{R}^N$, $\sigma_2(\boldsymbol{v})=\{\sigma_2(v_1),...,\sigma_2(v_N)\}^\top$. Following [77], we measure the NN complexity by using three quantities: the number of hidden layers (depth of the network) L, the number of nodes $\sum_{\ell=1}^L N_\ell$, and the number of nonzero weights, which is also the number of nonzeros in $\{(\boldsymbol{A}_\ell,\boldsymbol{b}_\ell),\ell=1,...,L+1\}$, denoted as $\#\Phi=\sum_{\ell=1}^{L+1}(\#\boldsymbol{A}_\ell+\#\boldsymbol{b}_\ell)$.

Let $f_r(\boldsymbol{x})$ be an arbitrary multivariate polynomial function of degree r on $\mathbb{R}^d$, which can be written as

$$f_r(\boldsymbol{x})=\sum\nolimits_{j_1+\cdots+j_d=0}^{r} a_{j_1\cdots j_d}x_1^{j_1}\cdots x_d^{j_d}, \tag{4.8}$$

where $a_{j_1\cdots j_d}\in\mathbb{R}$. [77] shows that $f_r(\boldsymbol{x})$ can be exactly represented by a ReQU network with specified structure.

Proposition 5. *For $f_r(\boldsymbol{x})$ given in (4.8), it can be represented exactly by a ReQU network with $d\lfloor\log_2 r\rfloor+d$ hidden layers, and the number of nodes and the number of nonzero weights are bounded by $\mathcal{O}\left(\mathcal{C}_d^{r+d}\right)$.*

Since Jacobi polynomials belongs to $f_r(\boldsymbol{x})$, next we will show the approximation of Jacobi polynomial functions by deep ReQU networks. First we rewrite $\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})$ as

$$\begin{aligned}\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})&=\prod\nolimits_{j=1}^{d}\sum\nolimits_{l_j=0}^{r_j}a_{l_j}^{r_j}(x_j-1)^{l_j}\\&=\sum\nolimits_{l_d=0}^{r_d}a_{l_d}^{r_d}\cdots\left(\sum\nolimits_{l_2=0}^{r_2}a_{l_2}^{r_2}\left(\sum\nolimits_{l_1=0}^{r_1}a_{l_1}^{r_1}(x_1-1)^{l_1}\right)(x_2-1)^{l_2}\right)\cdots(x_d-1)^{l_d}.\end{aligned}$$

To approximate $\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})$, we can first approximate $a_{x_1}^* = \sum_{l_1=0}^{r_1} a_{l_1}^{r_1}(x_1-1)^{l_1}$ using an univariate ReQU network, then treat $a_{x_1}^*$ as an network input and approximate $\sum_{l_2=0}^{r_2} a_{l_2}^{r_2}\left(a_{x_1}^*\right)(x_2-1)^{l_2}$ using a bi-variate ReQU network. We continue this network construction until we include all the variables. Next we introduce the detailed ReQU network construction process for $a_{x_1}^*$.

Denote the ReQU network by $\Phi_1(x_1) = z_{L_1}$, where we define $z_0 = x_1 - 1$, $\boldsymbol{z}_1 = \sigma_2(\boldsymbol{A}_1 z_0 + \boldsymbol{b}_1)$, $\boldsymbol{z}_l = \sigma_2(\boldsymbol{A}_l \boldsymbol{z}_{l-1} + \boldsymbol{b}_l)$ for $l = 2, ..., L_1 - 1$ and $z_{L_1} = \boldsymbol{A}_{L_1}\boldsymbol{z}_{L_1-1} + \boldsymbol{b}_L$. Here L_1 is determined by the degree of the target Jacobi polynomial. Let $L_1 = \lfloor \log_2(r_1) \rfloor + 2$, where $\lfloor a \rfloor$ denotes the biggest integer which is not bigger than a.

Denote $m(z_0) = \sum_{l=0}^{r_1} a_l^{r_1} z_0^l$ and we have $a_{x_1}^* = m(z_0)$. We first extend $m(z_0)$ to include monomials up to degree $2^{L_1-1}-1$ by zero padding,

$$m(z_0) := \sum_{l=0}^{r_1} a_l^{r_1} z_0^l + \sum_{l=r_1+1}^{2^{L_1-1}-1} 0 z_0^l = \sum_{l=0}^{2^{L_1-1}-1} \tilde{a}_l^{r_1} z_0^l,\ \tilde{a}_l^{r_1} = \begin{cases} a_l^{r_1}, & 0 \le l \le r_1 \\ 0, & r_1+1 \le l \le 2^{L_1-1}-1 \end{cases}.$$

If $r_1 = 0$, $m(z_0)$ would be a constant. There would be slight difference for the ReQU neural network construction for $r_1 = 1 (L_1 = 2)$, $r_1 = 2, 3 (L_1 = 3)$, and $r_1 > 3 (L_1 > 3)$. Several parameters would be shared by every layer: $\beta_1, w_1, \gamma_1 \in \mathbb{R}^4$, $\beta_2, w_2 \in \mathbb{R}^2$. Let $a_{21}^{r_1} = (\tilde{a}_1^{r_1}, \tilde{a}_3^{r_1}, ..., \tilde{a}_{2^{L_1-1}-1}^{r_1})^\top \in \mathbb{R}^{2^{L_1-2}}$, $a_{22}^{r_1} = (\tilde{a}_0^{r_1}, \tilde{a}_2^{r_1}, ..., \tilde{a}_{2^{L_1-1}-2}^{r_1})^\top \in \mathbb{R}^{2^{L_1-2}}$ and $\tilde{\boldsymbol{a}}^{r_1} = (\tilde{a}_0^{r_1}, \tilde{a}_1^{r_1}, ..., \tilde{a}_{2^{L_1-1}-1}^{r_1})^\top \in \mathbb{R}^{2^{L_1}}$, we also treat $a_{21}^{r_1}, a_{22}^{r_1}$ as parameters in the following neural network construction.

1. $L_1 = 2$, there would be two layers

   (a) Layer 1: $\boldsymbol{z_1} = \sigma_2(A_1 z_0 + b_1)$, where $A_1 = w_1$, $b_1 = \gamma_1$.

   (b) Layer 2 (output layer): $z_2 = A_{20}\boldsymbol{z_1} + b_{20}$, where $A_{20} = a_{21}^{r_1}\beta_1^\top$ and $b_{20} = a_{22}^{r_1}$.

2. $L_1 = 3$, there would be three layers:

   (a) Layer 1: $\boldsymbol{z_1} = \sigma_2(A_1 z_0 + b_1)$, where $A_1 = (w_1^\top, w_2^\top)^\top$, $b_1 = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^8$.

   (b) Layer 2: $\boldsymbol{z_2} = \sigma_2(A_2 \boldsymbol{z_1} + b_2)$, where $A_2 = A_{21}A_{20}$, $b_2 = A_{21}b_{20} + b_{21}$ and

$$A_{21} = \begin{bmatrix} w_1 & \mathbf{0} & \mathbf{0} \\ \\ \mathbf{0} & w_1 & \gamma_1 \end{bmatrix}, b_{21} = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^8,$$

$$A_{20} = \begin{bmatrix} a_{21}\beta_1^\top & \mathbf{0} \\ \\ \mathbf{0} & \beta_2^\top \end{bmatrix}, b_{20} = (a_{22}^{r_1\top}, 0)^\top \in \mathbb{R}^3.$$

   (c) Layer 3 (output layer): $z_3 = A_{30}\boldsymbol{z_2}$, where $A_{30} = (\beta_1^\top, \beta_1^\top)^\top \in \mathbb{R}^8$.

3. $L_1 > 3$, the first layer and output layer are of same structure as in the case $L_1 = 3$, but more hidden layers:

   (a) Layer 1: $\boldsymbol{z_1} = \sigma_2(A_1 z_0 + b_1)$, where $A_1 = (w_1^\top, w_2^\top)^\top$, $b_1 = (\gamma_1^\top, \mathbf{0})^\top$.

   (b) Layer 2: $\boldsymbol{z_2} = \sigma_2(A_2 \boldsymbol{z_1} + b_2)$, where $A_2 = A_{21}A_{20}$, $b_2 = A_{21}b_{20} + b_{21}$ and

$$A_{21} = \begin{bmatrix} w_1 & \mathbf{0} & & & & & & \mathbf{0} \\ \mathbf{0} & w_1 & & & & & & \gamma_1 \\ & & w_1 & \mathbf{0} & & & & \mathbf{0} \\ & & \mathbf{0} & w_1 & & & & \gamma_1 \\ & & & & \cdots & & & \\ & & & & & w_1 & \mathbf{0} & \mathbf{0} \\ & & & & & \mathbf{0} & w_1 & \gamma_1 \\ & & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{(8\times 2^{L_1-3}+1)\times(2^{L_1-2}+1)},$$

$$b_{21} = (\gamma_1^\top, \mathbf{0}, \gamma_1^\top, \mathbf{0}, ..., \gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^{(8\times 2^{L_1-3}+1)\times 1},$$

$$A_{20} = \begin{bmatrix} a_{21}^{r_1}\beta_1^\top & \mathbf{0} \\ \mathbf{0} & \beta_2^\top \end{bmatrix}, b_{20} = (a_{22}^{r_1\top}, 0)^\top \in \mathbb{R}^{2^{L_1-2}+1}.$$

(c) (If $L_1 > 4$) Layer k for $3 \leq k \leq L_1 - 2$: $\boldsymbol{z_k} = \sigma_2(A_k \boldsymbol{z_{k-1}} + b_k)$, where $A_k = A_{k1}A_{k0}$, $b_k = b_{k1}$ and

$$A_{k1} = \begin{bmatrix} w_1 & \mathbf{0} & & & & & & \mathbf{0} \\ \mathbf{0} & w_1 & & & & & & \gamma_1 \\ & & w_1 & \mathbf{0} & & & & \mathbf{0} \\ & & \mathbf{0} & w_1 & & & & \gamma_1 \\ & & & & ... & & & \\ & & & & & w_1 & \mathbf{0} & \mathbf{0} \\ & & & & & \mathbf{0} & w_1 & \gamma_1 \\ & & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{(8\times 2^{L_1-k-1}+1)\times(2^{L_1-k}+1)},$$

$$b_{k1} = (\gamma_1^\top, \mathbf{0}, \gamma_1^\top, \mathbf{0}, ..., \gamma_1^\top, \mathbf{0})^\top,$$

$$A_{k0} = \begin{bmatrix} \mathbf{I}_{2^{L_1-k}} \otimes (\beta_1^\top, \beta_1^\top)^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{(2^{L_1-k}+1)\times(8\times 2^{L_1-k}+1)}.$$

where $\mathbf{I}_{2^{L_1-k}}$ is the identity matrix in $\mathbb{R}^{2^{L_1-k}}$ and $\otimes$ stands for Kronecker product.

(d) Layer L_1-1: $\boldsymbol{z_{L_1-1}} = \sigma_2(A_{L_1-1}\boldsymbol{z_{L_1-2}} + b_{L_1-1})$, where $A_{L_1-1} = A_{L_1-1,1}A_{L_1-1,0}$, $b_{L_1-1} = b_{L_1-1,1}$ and

$$A_{L_1-1,1} = \begin{bmatrix} w_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & w_1 & \gamma_1 \end{bmatrix}, b_{L_1-1,1} = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^8,$$

$$A_{L_1-1,0} = \begin{bmatrix} \mathbf{I}_2 \otimes (\beta_1^\top, \beta_1^\top)^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}.$$

(e) Layer L_1 (output layer): $z_{L_1} = A_{L_1,0}\boldsymbol{z_{L_1-1}}$, where

$$A_{L_1,0} = (\beta_1^\top, \beta_1^\top)^\top.$$

When realizing $\sum_{l_2=0}^{r_2} a_{l_2}^{r_2} a_{x_1}^*(x_2-1)^{l_2}$, since $a_{x_1}^*$ is network input instead of parameters, we need to modify the neural network construction for the first and the second layer. Let $L_2 = \lfloor \log_2(r_2) \rfloor + 2$. Denote the ReQU network by $\Phi_2(z_{L_1}, x_2) = z_{L_2}$, where we modify the the nodes of the first layer as $\boldsymbol{z_0} = (z_{L_1}, x_2)^\top$. Again, there would be slight difference for the ReQU neural network construction for $r_2 = 1(L_2 = 2)$, $r_2 = 2, 3(L_2 = 3)$, and $r_2 > 3(L_2 > 3)$. Similarly we have $a_{21}^{r_2} = (\tilde{a}_1^{r_2}, \tilde{a}_3^{r_2}, ..., \tilde{a}_{2^{L_1-1}-1}^{r_2})^\top \in \mathbb{R}^{2^{L_1-2}}$, $a_{22}^{r_2} = (\tilde{a}_0^{r_2}, \tilde{a}_2^{r_2}, ..., \tilde{a}_{2^{L_1-1}-2}^{r_1})^\top \in \mathbb{R}^{2^{L_1-2}}$ and $\tilde{\boldsymbol{a}}^{r_2} = (\tilde{a}_0^{r_2}, \tilde{a}_1^{r_2}, ..., \tilde{a}_{2^{L_1-1}-1}^{r_2})^\top \in \mathbb{R}^{2^{L_2}}$, we also treat $a_{21}^{r_2}, a_{22}^{r_2}$ as parameters in the following neural network construction.

1. $L_2 = 2$:

   (a) Layer 1: $\boldsymbol{z^1} = \sigma_2(A_1\boldsymbol{z}_0 + b_1)$, where $A_1 = \begin{bmatrix} w_1 & \mathbf{0} \\ w_1 & \gamma_1 \end{bmatrix}$, $b_1 = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^8$.

   (b) Layer 2: (output layer): $z_2 = A_{20}\boldsymbol{z_1}$, where $A_{20} = (a_{22}^{r_2}\beta_1^\top, a_{21}^{r_2}\beta_1^\top)$.

2. $L_2 = 3$:

   (a) Layer 1: $\boldsymbol{z_1} = \sigma_2(A_1\boldsymbol{z}_0 + b_1)$, where $A_1 = \begin{bmatrix} w_1 & \mathbf{0} \\ w_1 & \gamma_1 \\ \mathbf{0} & w_2 \end{bmatrix}$, $b_1 = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^{10}$.

   (b) Layer 2: $\boldsymbol{z_2} = \sigma_2(A_2\boldsymbol{z_1} + b_2)$, where $A_2 = A_{21}A_{20}$, $b_2 = b_{21}$ and

$$A_{21} = \begin{bmatrix} w_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & w_1 & \gamma_1 \end{bmatrix}, b_{21} = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^8, A_{20} = \begin{bmatrix} a_{22}^{r_2}\beta_1^\top & a_{21}^{r_2}\beta_1^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \beta_2^\top \end{bmatrix}.$$

(c) Layer 3 (output layer): have same structure as the layer 3 in $a^*_{x_1}$ when $L_1 = 3$.

3. $L_2 > 3$:

(a) Layer 1: $\boldsymbol{z_1} = \sigma_2(A_1\boldsymbol{z}_0 + b_1)$, where $A_1 = \begin{bmatrix} w_1 & \mathbf{0} \\ w_1 & \gamma_1 \\ \mathbf{0} & w_2 \end{bmatrix}$, $b_1 = (\gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^{10}$.

(b) Layer 2: $\boldsymbol{z_2} = \sigma_2(A_2\boldsymbol{z_1} + b_2)$, where $A_2 = A_{21}A_{20}$, $b_2 = b_{21}$ and

$$A_{21} = \begin{bmatrix} w_1 & \mathbf{0} & & & & & & \mathbf{0} \\ \mathbf{0} & w_1 & & & & & & \gamma_1 \\ & & w_1 & \mathbf{0} & & & & \mathbf{0} \\ & & \mathbf{0} & w_1 & & & & \gamma_1 \\ & & & & \dots & & & \\ & & & & & w_1 & \mathbf{0} & \mathbf{0} \\ & & & & & \mathbf{0} & w_1 & \gamma_1 \\ & & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{(8\times 2^{L_2-3}+1)\times(2^{L_2-2}+1)},$$

$$b_{21} = (\gamma_1^\top, \mathbf{0}, \gamma_1^\top, \mathbf{0}, ..., \gamma_1^\top, \mathbf{0})^\top \in \mathbb{R}^{(8\times 2^{L_2-3}+1)\times 1},$$

$$A_{20} = \begin{bmatrix} a_{22}^{r_2}\beta_1^\top & a_{21}^{r_2}\beta_1^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \beta_2^\top \end{bmatrix}.$$

(c) Layer 3 and further: have same structure as the corresponding layers in $a^*_{x_1}$ when $L_1 > 3$.

We notice that x_2 is included in the overall network only after certain number of layers, we can keep a record of input x_2 in each layer using multiple $id_X(\cdot)$ operations, such that x_1 and x_2 can start from the same layer. Repeat the same process as we include

x_2 for x_3, ... x_d. After we have included all variables, we will obtain a ReQU network $\Phi_d^{\boldsymbol{r}}(\boldsymbol{x}) = \Phi_d^{\boldsymbol{r}}(x_1, x_2, ..., x_d)$. When $\beta_1, \beta_2, w_1, w_2, \gamma_1$ take specific values as given in [77], $\tilde{\boldsymbol{a}}^{\boldsymbol{r}} = \{\tilde{a}^{r_j}, 1 \le j \le d\}$ take specific values as given in [103], $\Phi_d^{\boldsymbol{r}}(\boldsymbol{x})$ can exactly represent $\boldsymbol{J}_{\boldsymbol{r}}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})$.

The last step is to assemble all such ReQU networks to complete our final ReQU network,

$$\Phi_d(\boldsymbol{x}) = \sum_{\boldsymbol{r} \in \Upsilon_{m,v}} \eta_{\boldsymbol{r}} \Phi_d^{\boldsymbol{r}}(\boldsymbol{x}), \tag{4.9}$$

where $\eta_{\boldsymbol{r}} \in \mathbb{R}, \boldsymbol{r} \in \Upsilon_{m,v}$.

4.4 Deep ReQU Network Estimator

Define the ReQU network class as

$$\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}, f(\boldsymbol{x}) = \sum_{\boldsymbol{r} \in \Upsilon_{m,v}} \eta_{\boldsymbol{r}} \Phi_d^{\boldsymbol{r}}(\boldsymbol{x}; \beta_1, \beta_2, w_1, w_2, \gamma_1, \tilde{\boldsymbol{a}}^{\boldsymbol{r}}), ||f||_\infty \le B\}. \tag{4.10}$$

For $\alpha_j, \beta_j \ge 0$, $1 = \omega^{\alpha_j,\beta_j}(0) \le \sup_{x_j \in [-1,1]} |\omega^{\alpha_j,\beta_j}(x_j)| \le \widetilde{b}$ for some constant $\widetilde{b} \in [1, \infty)$. Note that when $\alpha_j, \beta_j = 0$, $\sup_{x_j \in [-1,1]} |\omega^{\alpha_j,\beta_j}(x_j)| = 1$, so $\widetilde{b} = 1$. Then

$$\sup_{x \in \mathcal{X}} |\boldsymbol{\omega}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})| \le \widetilde{b}^d. \tag{4.11}$$

Moreover, $\int_{-1}^{1} \omega^{\alpha_j,\beta_j}(x_j)\, dx_j \le b$ for some constant $b \in (0, 2\widetilde{b}]$ and thus $\int_{\mathcal{X}} \boldsymbol{\omega}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x} \le b^d$. Then for any $f \in \mathcal{F}$,

$$\int_{\mathcal{X}} f(\boldsymbol{x})^2 \boldsymbol{\omega}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x} \le B^2 \int_{\mathcal{X}} \boldsymbol{\omega}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})\, d\boldsymbol{x} \le B^2 b^d. \tag{4.12}$$

Let

$$f^0 = \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X} \times \mathcal{Y}} \boldsymbol{\omega}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) (f(\boldsymbol{x}) - y)^2 d\mu(\boldsymbol{x}, y). \tag{4.13}$$

The function f^0 can be written as $f^0(\boldsymbol{x}) = \sum_{\boldsymbol{r}\in\Upsilon_m} \eta^0_{\boldsymbol{r}} \boldsymbol{J}^{\boldsymbol{\alpha},\boldsymbol{\beta}}_{\boldsymbol{r}}(\boldsymbol{x}) = \boldsymbol{\eta}^{0\top}\boldsymbol{J}^{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})$, where $\boldsymbol{\eta}^0 = \{\eta^0_{\boldsymbol{r}}, \boldsymbol{r}\in\Upsilon_m\}^\top$. We obtain the penalized deep ReQU network (PDRN) estimator $\widehat{f}$ of the true regression function g_0 from minimizing the following emipirical risk:

$$\widehat{f} = \arg\min_{f\in\mathcal{F}}\{n^{-1}\sum_{i=1}^{n}\boldsymbol{\omega}^{\ \boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{X}_i)\left(f(\boldsymbol{X}_i)-Y_i\right)^2 + \lambda||f||^2_{\Psi}\}, \tag{4.14}$$

where $||f||^2_\Psi = \boldsymbol{\eta}^\top\boldsymbol{\eta}$ and $\lambda > 0$ is a tuning parameter for the L_2 penalty.

There are three tuning parameters: m, v and λ and we select them through 5-fold cross validation. Let θ be the vector of parameters collecting all parameters including $\beta_1, \beta_2, w_1, w_2, \gamma_1, \tilde{\boldsymbol{a}}^{\boldsymbol{r}}$ for $\boldsymbol{r}\in\Upsilon_{m,v}$. We apply the ADAM method to update all the parameters.

Denote the parameter estimates by $\hat{\boldsymbol{\eta}} = \{\hat{\eta}_{\boldsymbol{r}}, \boldsymbol{r}\in\Upsilon_{m,v}\}$, $\hat{\beta}_1$, $\hat{\beta}_2$, $\hat{w}_1$, $\hat{w}_2$, $\hat{\gamma}_1$ and $\hat{\tilde{\boldsymbol{a}}} = \{\hat{\tilde{\boldsymbol{a}}}^{\boldsymbol{r}}, \boldsymbol{r}\in\Upsilon_{m,v}\}$, we have

$$\hat{f}(\boldsymbol{x}) = \sum_{\boldsymbol{r}\in\Upsilon_{m,v}} \hat{\eta}_{\boldsymbol{r}}\hat{\Phi}^{\boldsymbol{r}}_d(\boldsymbol{x}) = \hat{\boldsymbol{\eta}}^\top\hat{\Phi}_d(\boldsymbol{x}), \tag{4.15}$$

where $\hat{\Phi}_d(\boldsymbol{x}) = \{\hat{\Phi}^{\boldsymbol{r}}_d(\boldsymbol{x}), \boldsymbol{r}\in\Upsilon_{m,v}\} = \{\Phi^{\boldsymbol{r}}_d(\boldsymbol{x};\hat{\beta}_1,\hat{\beta}_2,\hat{w}_1,\hat{w}_2,\hat{\gamma}_1,\hat{\tilde{\boldsymbol{a}}}^{\boldsymbol{r}}), \boldsymbol{r}\in\Upsilon_{m,v}\}$.

The functional derivatives estimates of $f(\boldsymbol{x})$ are obtained directly from $\hat{f}(\boldsymbol{x})$. The partial derivatives of $\hat{f}(\boldsymbol{x})$ with mult-index $\boldsymbol{k} = (k_1, ..., k_d)^\top \in \mathbb{N}^d$ is given as

$$\begin{aligned} D^{\boldsymbol{k}}\hat{f}(\boldsymbol{x}) &= \frac{\partial^{|\boldsymbol{k}|_1}\hat{f}(\boldsymbol{x})}{\partial x_1^{k_1}\cdots\partial x_d^{k_d}} \\ &= \frac{\partial^{|\boldsymbol{k}|_1}\{\sum_{\boldsymbol{r}\in\Upsilon_{m,v}}\hat{\eta}_{\boldsymbol{r}}\hat{\Phi}^{\boldsymbol{r}}_d(\boldsymbol{x})\}}{\partial x_1^{k_1}\cdots\partial x_d^{k_d}} \\ &= \sum_{\boldsymbol{r}\in\Upsilon_{m,v}}\hat{\eta}_{\boldsymbol{r}}\frac{\partial^{|\boldsymbol{k}|_1}\hat{\Phi}^{\boldsymbol{r}}_d(\boldsymbol{x})}{\partial x_1^{k_1}\cdots\partial x_d^{k_d}}. \end{aligned}$$

To obtain $D^{\boldsymbol{k}}\hat{f}(\boldsymbol{x})$ we only need $\frac{\partial^{|\boldsymbol{k}|_1}\hat{\Phi}^{\boldsymbol{r}}_d(\boldsymbol{x})}{\partial x_1^{k_1}\cdots\partial x_d^{k_d}}$ for $\boldsymbol{r}\in\Upsilon_{m,v}$. Similar to the partial derivatives of d-dimensional Jacobi polynomial in Equation 4.6. We estimate the partial

Algorithm 3 ADAM

```
Require: $\theta_{ini}$: Initial value for $\boldsymbol{\theta}$
Require: $\epsilon_0$: converge criterion
Require: $\tilde{\alpha}$: step size with default value of 0.001
Require: $\tilde{\beta}_1$: decay rate with default value of 0.9
Require: $\tilde{\beta}_2$: decay rate with default value of 0.999
Require: $\tilde{\epsilon}$: stabilizer with default value of $10^{-8}$
  $t \leftarrow 0$ (Initialize timestep)
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $\theta_0 \leftarrow \theta_{ini}$
  while $\theta_t$ not converge do
      $t \leftarrow t + 1$
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Obtain gradient for $\theta$ at timestep $t$)
      $m_t \leftarrow \tilde{\beta}_1 m_{t-1} + (1 - \tilde{\beta}_1) g_t$ (Update biased first moment estimate)
      $v_t \leftarrow \tilde{\beta}_2 v_{t-1} + (1 - \tilde{\beta}_2) g_t \odot g_t$ (Update biased second raw moment estimate))
      $\widehat{m}_t \leftarrow m_t/(1 - \tilde{\beta}_1^t)$ (Compute bias-corrected first moment estimate)
      $\widehat{v}_t \leftarrow v_t/(1 - \tilde{\beta}_2^t)$ (Compute bias-corrected second raw moment estimate)
      $\Delta(\theta_t) \leftarrow \widehat{m}_t/(\sqrt{\widehat{v}_t} + \tilde{\epsilon})$
      $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} - \tilde{\alpha}\Delta(\boldsymbol{\alpha}_t)$ (Update $\theta$ at timestep $t$)
  end while
      return $\theta_t$
```

derivatives of $\hat{\Phi}_d^{\boldsymbol{r}}(\boldsymbol{x})$ from

$$D^{\boldsymbol{k}}\hat{\Phi}_d^{\boldsymbol{r}}(\boldsymbol{x}) = \frac{\partial^{|\boldsymbol{k}|_1}\hat{\Phi}_d^{\boldsymbol{r}}(\boldsymbol{x})}{\partial x_1^{k_1}\cdots\partial x_d^{k_d}} = \prod_{j=1}^{d} d_{r_j,k_j}\hat{\Phi^*}_j^{r_j-k_j}(x_j), \tag{4.16}$$

where $\boldsymbol{k} = (k_1, k_2, ..., k_d)$, $d_{r_j,k_j} = \frac{r_j!}{2^{k_j}(r_j-k_j)!}$ and $\hat{\Phi^*}_j^{r_j-k_j}(x_j)$ is obtained from $\hat{\Phi}_j^{r_j-k_j}(x_j)$ by changing the coefficients $\hat{a}_{l_j}^{r_j-k_j}$ for $1 \le l_j \le r_j - k_j$ to $\hat{a}_{l_j}^{r_j}/\hat{a}_{l_j}^{r_j-k_j}$ for $1 \le l_j \le r_j - k_j$.

4.5 Simulation Studies

4.5.1 Methods Used

In this section, we conduct simulation studies to assess the finite-sample performance of the proposed methods. We estimate the target function using our proposed PRDN method as well as five other popular methods, including the fully-connected feedforward neural networks (FNN), the linear models (LM), the generalized additive models (GAM), the gradient boosted machines (GBM) and the random forests (RF).

We use the Rectified Linear Unit (ReLU) as the activation function for FNN. We use cubic regression spline basis functions for GAM. We apply grid search with 5-fold cross-validation to select hyperparameters for all methods, including the number of hidden layers and the number of neurons in each layer for FNN, the number of trees and max depths of trees for RF and GBM, and the learning rate for GBM. All the simulation studies are implemented in Python 3.9.9. The FNN, LM, GAM and RF/GBM methods are implemented using the packages tensorflow, statsmodel, pygam and scikit-learn, respectively.

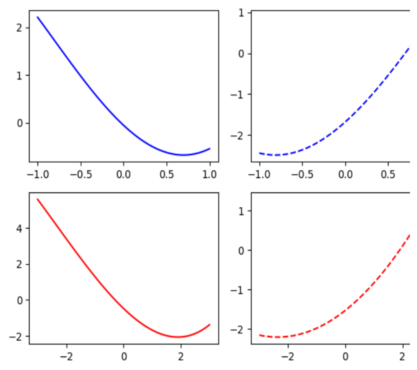
4.5.2 Date Generating Process

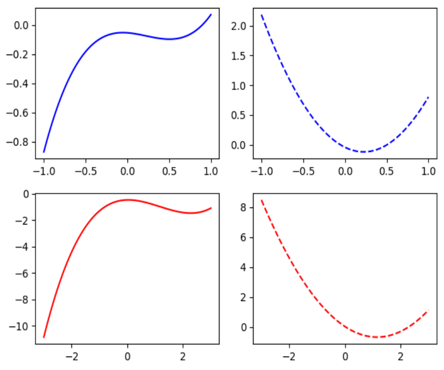
For illustration of the methods, we generate data from the following nonlinear model:

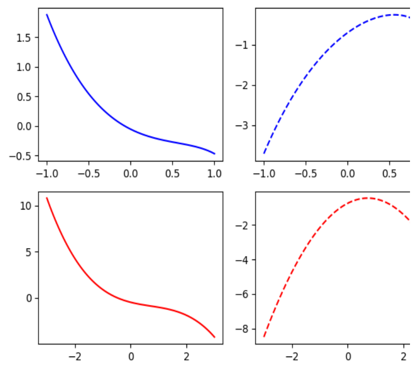
$$\begin{aligned}\mathbb{E}(Y_i|X_i) &= (0.5(X_{i1}+0.6)+0.8(X_{i2}-0.3)+X_{i4})^3+(0.5(-X_{i3}+1))^3\\ &\quad -4\sin(0.8X_{i1}+1)+0.4e^{1.2(-X_{i3}+0.5)}\\ &\quad +4\left(0.2\log(|\frac{1}{\cos(0.5(X_{i4}-0.6))}|)+6\arctan(0.5(X_{i5}-0.3))-3.1X_{i5}+1\right)\left(\frac{1.8}{X_{i3}-1.8}+2\right).\end{aligned}$$

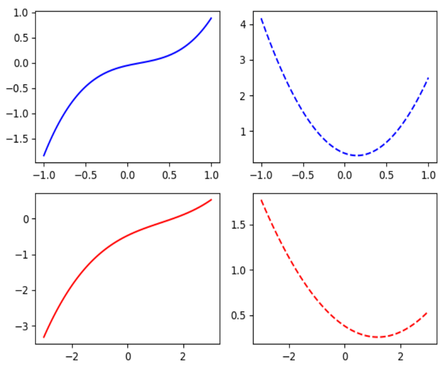
where $Y_i = E(Y_i \mid X_i) + \epsilon_i$ and ϵ_i are independently generated from the standard normal distribution for $1 \le i \le n$. $X_i = (X_{i1}, X_{i2}, ..., X_{i5})$, and the confounders X_{ij} are generated from $X_{ij} = 2(F(Z_{ij}) - 0.5)$, where $Z_i = (Z_{i1}, ..., Z_{ip})^\top \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{\Sigma})$, $\mathbf{\Sigma} = \{\sigma_{kk'}\}$, $\sigma_{kk'} = 0.5^{|k-k'|}$ for $1 \le k, k' \le 5$, and $F(\cdot)$ is the cumulative distribution function of standard normal distribution for $1 \le i \le n, 1 \le j \le p$. We consider $n = 2000$, 5000. For each setting, we run 200 replications.

This model is inspired from Airfoil Self-Noise data in Section 4.6. Figure 4.1 compares the mean response versus each covariate, partial derivative versus each covariate from the simulation model to those from estimated model of Airfoil Self-Noise data. The partial derivatives are the first partial derivatives with respect to each covariate. Both the mean response and the derivatives are plotted against each covariate while other covariates fixed at their mean values. We can see the simulation model has reproduced the nonlinear trends in both the mean response and the derivatives.


(a) Covariate 1


(b) Covariate 2


(c) Covariate 3


(d) Covariate 4

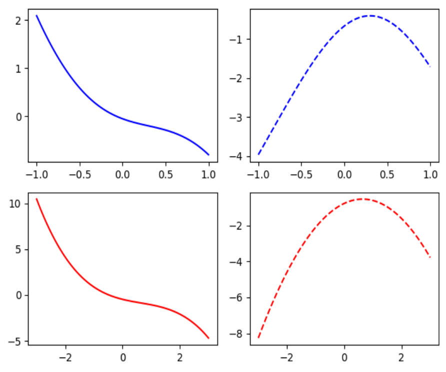

(e) Covariate 5

Figure 4.1: Mean response and partial derivatives from simulation model and Airfoil self-noise data for all 5 variables. For each subplot, the two top plots are from simulation model and the bottom plots are from Airfoil self-noise data. The solid lines are the mean response and the dashed line are the partial derivatives.

4.5.3 Simulation Results

To evaluate the estimation accuracy of the proposed mean response estimates and derivative estimates, we report the following three metrics, average squared bias (bias^2), average variance and average mean squared error (mse) :

$$\begin{aligned}
\text{average bias}^2 &= \frac{1}{n}\sum_{i=1}^{n}\{\frac{1}{n_{rep}}\sum_{j=1}^{n_{rep}}\hat{f}_j(x_i^*) - f(x_i^*)\}^2, \\
\text{average variance} &= \frac{1}{n}\sum_{i=1}^{n}\{\frac{1}{n_{rep}}\sum_{j=1}^{n_{rep}}\hat{f}_j(x_i^*)^2 - (\frac{1}{n_{rep}}\sum_{j=1}^{n_{rep}}\hat{f}_j(x_i^*))^2\}, \\
\text{average mse} &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n_{rep}}\sum_{j=1}^{n_{rep}}\{\hat{f}_j(x_i^*) - f(x_i^*)\}^2.
\end{aligned}$$

where $\hat{f}_j(\cdot)$ is the estimated function from the j-th replicate and $f(\cdot)$ is the target mean response function. We evaluate the estimated function based on the same set of the covariates x_i for all methods, which are generated in the same way as the covariates in each replicate.

For our simulation model with $d = 5$, table 4.3 reports the three evaluation metrics of the six methods for estimation of mean response obtained based on 200 simulation replications. Our proposed PRDN method has the lowest MSE values for both $n = 2000$ and $n = 5000$ settings. As n increases, the squared bias value and variance decrease. Compared to FNN estimates, while the squared bias values are close, the variance of PRDN estimates is smaller since PRDN method has less parameters to estimate due to its sparse structure. The GBM and RF estimates have inferior performance than the neural network based estimates. LM and GAM estimates have large bias due to model mis-specification.

Figure 4.2 and Figure 4.3 show the mean response curves and functional derivative curves from the estimated model and the true model while all other covariates are hold at

	n=2000						n=5000					
	PRDN	FNN	LM	GAM	GBM	RF	PRDN	FNN	LM	GAM	GBM	RF
bias^2	0.0200	0.0215	1.9403	1.3139	0.0625	0.1269	0.0072	0.0076	1.9398	1.3134	0.0408	0.0800
var	0.0548	0.0598	0.0087	0.1292	0.1668	0.1420	0.0234	0.0293	0.0033	0.0493	0.0945	0.0981
mse	0.0748	0.0813	1.9491	1.4431	0.2293	0.2689	0.0306	0.0371	1.9430	1.3627	0.1354	0.1781

Table 4.3: The average bias^2, variance and mse of the six methods for estimation of mean response obtained based on 200 simulation replications.

		n=2000					n=5000				
		x_1	x_2	x_3	x_4	x_5	x_1	x_2	x_3	x_4	x_5
	bias^2	0.0129	0.0144	0.0047	0.0147	0.0019	0.0039	0.0058	0.0024	0.0064	0.0009
mean response	var	0.0072	0.0076	0.0074	0.0084	0.0074	0.0037	0.0042	0.0038	0.0039	0.0035
	mse	0.0201	0.0220	0.0121	0.0231	0.0093	0.0076	0.0099	0.0063	0.0103	0.0044
	bias^2	0.0521	0.0226	0.0394	0.1060	0.0143	0.0129	0.0076	0.0248	0.0300	0.0112
partial derivative	var	0.0437	0.0536	0.0475	0.0487	0.0395	0.0204	0.0253	0.0224	0.0269	0.0228
	mse	0.0958	0.0762	0.0868	0.1548	0.0538	0.0333	0.0329	0.0472	0.0570	0.0340

Table 4.4: The average bias^2, variance and mse of the PRDN estimates of mean response and functional derivative.

their mean values, for $n = 2000$ and $n = 5000$, respectively. The derivatives are first order partial derivative with respect to each covariate. The light grey dashed lines are estimated from 200 replications and they have covered the red solid line, which is evaluated from the true model, which shows our estimates have captured the patterns in the relationships between each covariate and mean response. As n increases, the coverage band is narrower and reproduce the trend with a higher accuracy. The average squared bias (bias^2), average variance and average mean squared error (mse) are reported in Table 4.4.

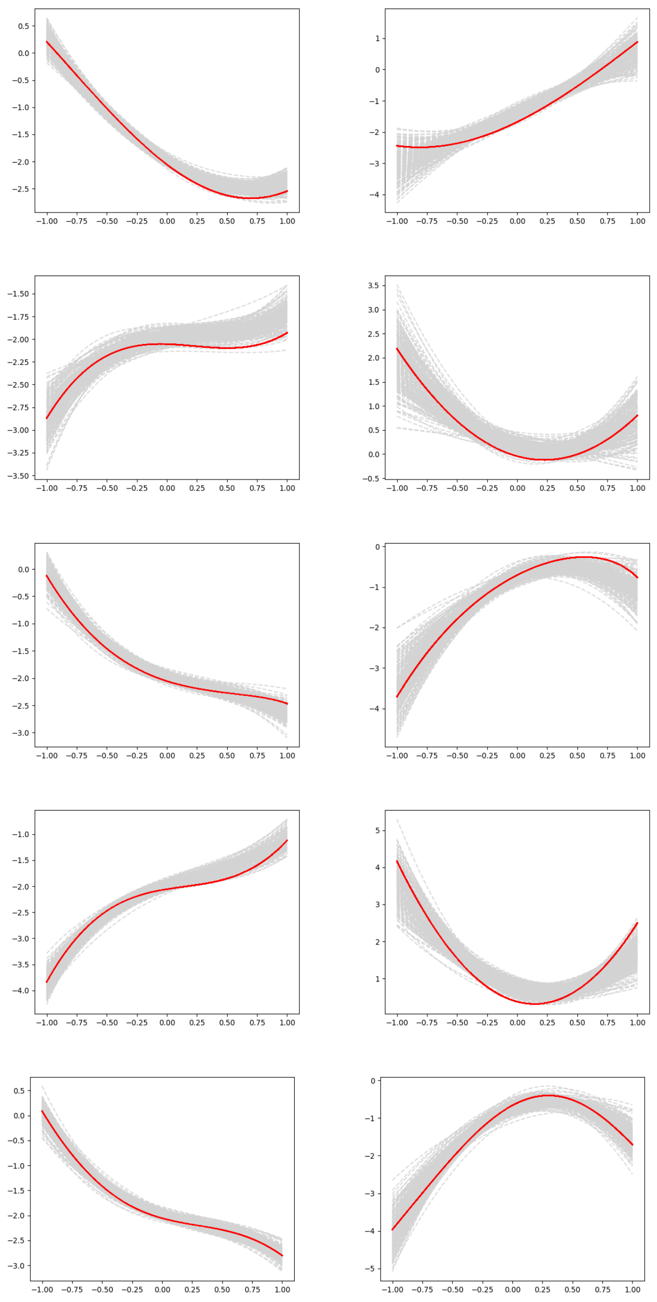


Figure 4.2: Mean response curves and functional derivative curves from estimated model and true model for all 5 variables (from top to bottom: Covariate 1 to Covariate 5). For each subplot, the left plot is the mean response and the right plot is the partial derivative. The red solid lines are from the true model and the grey dashed line(shade) are from the estimated models of all 200 replicates. $n = 2000$.

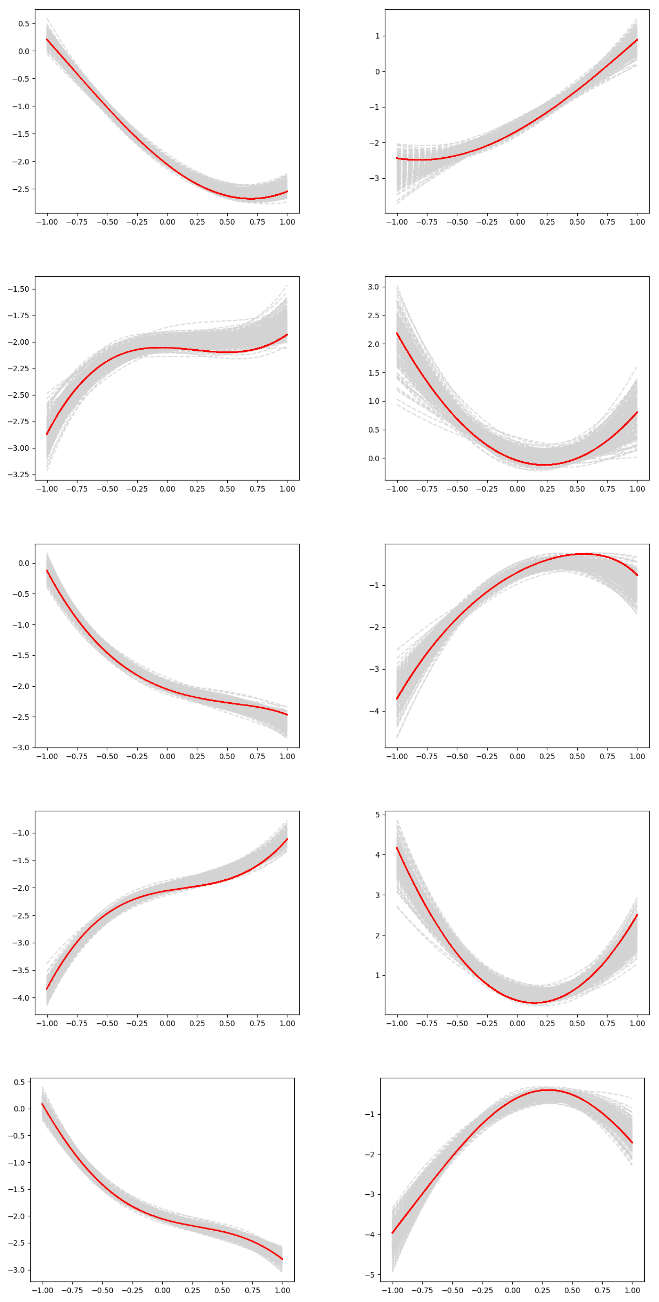


Figure 4.3: Mean response curves and functional derivative curves from estimated model and true model for all 5 variables (from top to bottom: Covariate 1 to Covariate 5). For each subplot, the left plot is the mean response and the right plot is the partial derivative. The red solid lines are from the true model and the grey dashed line(shade) are from the estimated models of all 200 replicates. $n = 5000$.

4.6 Real Applications

In this section, we illustrate our proposed method by using two datasets with continuous response variables (Abalone data and Airfoil Self-Noise data). Each dataset is randomly split into 80% training data and 20% test data. The training data is used to fit the model, whereas the test data is used to examine the prediction accuracy. Then, we compare our PDRN with five methods, including fully-connected feedforward neural networks (FNN), the linear models (LM), the generalized additive models (GAM), the gradient boosted machines (GBM) and the random forests (RF). For all methods, the tuning parameters are selected by 5-fold cross validations based on a grid search, and the software packages used are the same as simulation studies, see Section 4.5.1.

4.6.1 Abalone Data

The abalone dataset is available at the UCI Machine Learning Repository ([41]), which contains 4177 observations and 9 attributes. The attributes are: Sex (male, female and infant), Length (longest shell measurement), Diameter (perpendicular to length), Height (with meat in shell), Whole weight (whole abalone), Shucked weight (weight of meat), Viscera weight (gut weight after bleeding), Shell weight (after being dried) and Rings (+1.5 gives the age in years). The goal is to predict the age of abalone based on these physical measurements. Since the age depends on the Rings, we take the Rings as the response variable. Since Length and Diameter are highly correlated with the correlation coefficient 0.9868 and infant has no gender, we delete Length and Sex and we use the only six covariates, Diameter, Height, Whole weight, Shucked weight, Viscera weight and Shell weight, in

	PRDN	FNN	GLM	GAM	GBM	RF
MSPE	4.1023	4.1084	4.3910	4.3262	4.2419	4.2317

Table 4.5: The mean squared prediction error (MSPE) from the six different methods for the abalone data.

our analysis. In the dataset, there are two observations with zero value for Height, and two other observations are outliers, so we delete these four observations and use the remaining 4173 observations in our analysis.

In order to evaluate our method we report the mean squared prediction error (MSPE), which is defined as $\frac{1}{n}\sum(\hat{y}_i - y_i)^2$, where n is the test sample size, $\hat{y}_i$ and y_i are the estimated mean response and its observation value. Table 4.5 reports the MSPE values in the test data for six different methods. We can see our proposed PRDN estimate is comparable to FNN estimate and has the smallest MSPE compared to other methods.

Figure 4.4 depicts the fitted mean response curves (solid lines) and functional partial derivative curves (dashed lines) of the response Rings versus each of the four covariates obtained from our PDRN method, while other covariates are fixed at their mean values. We can see nonlinear trends exist in all plots, especially for Diameter and Whole weight. For Whole weight, the Rings seems to be increasing steadily as the Whole weight is increasing, but the functional derivative curve has a pattern that first rise then fall, which shows a changing speed of the increase in Rings. For Height, Whole weight and Shell weight, the fitted values of Rings are increasing as the covariate value increases. However, from the functional derivative curves we can see different patterns. The rate of increase of Rings keeps decreasing as Height is increasing, similar pattern can be found in the plots for Shell

	PRDN	FNN	GLM	GAM	GBM	RF
MSPE	0.1627	0.1623	0.4931	0.4123	0.3614	0.2739

Table 4.6: The mean squared prediction error (MSPE) from the six different methods for the airfoil data.

weight, however the rate of increase of Rings speeds up for a while that goes down after a certain point as the Whole weight increase. The function derivative curves show us more insights in understanding the relationship between the Rings and all the six covariates.

4.6.2 Airfoil Self-Noise Data

The abalone dataset is available at the UCI Machine Learning Repository ([41]), which contains 1503 observations and 5 attributes. The attributes are Frequency, in Hertzs; Angle of attack, in degrees; Chord length, in meters; Free-stream velocity, in meters per second; Suction side displacement thickness, in meters. The goal is to predict scaled sound pressure level, in decibels. Since each unit of measurement of covariate is different with the other, the original range of covariates are quite different, we standardize each covariate.

Table 4.6 reports the mean squared prediction error (MSPE) values in the test data for six different methods. We can see our proposed PRDN estimate is comparable to FNN estimate and has the smallest MSPE compared to other methods.

Figure 4.5 depicts the fitted mean response curves (solid lines) and functional partial derivative curves (dashed lines) of the response sound pressure level versus each of the five covariates obtained from our PDRN method, while other covariates are fixed at their mean values. Clear nonlinear trend exist in every plot and interestingly, each plot, for

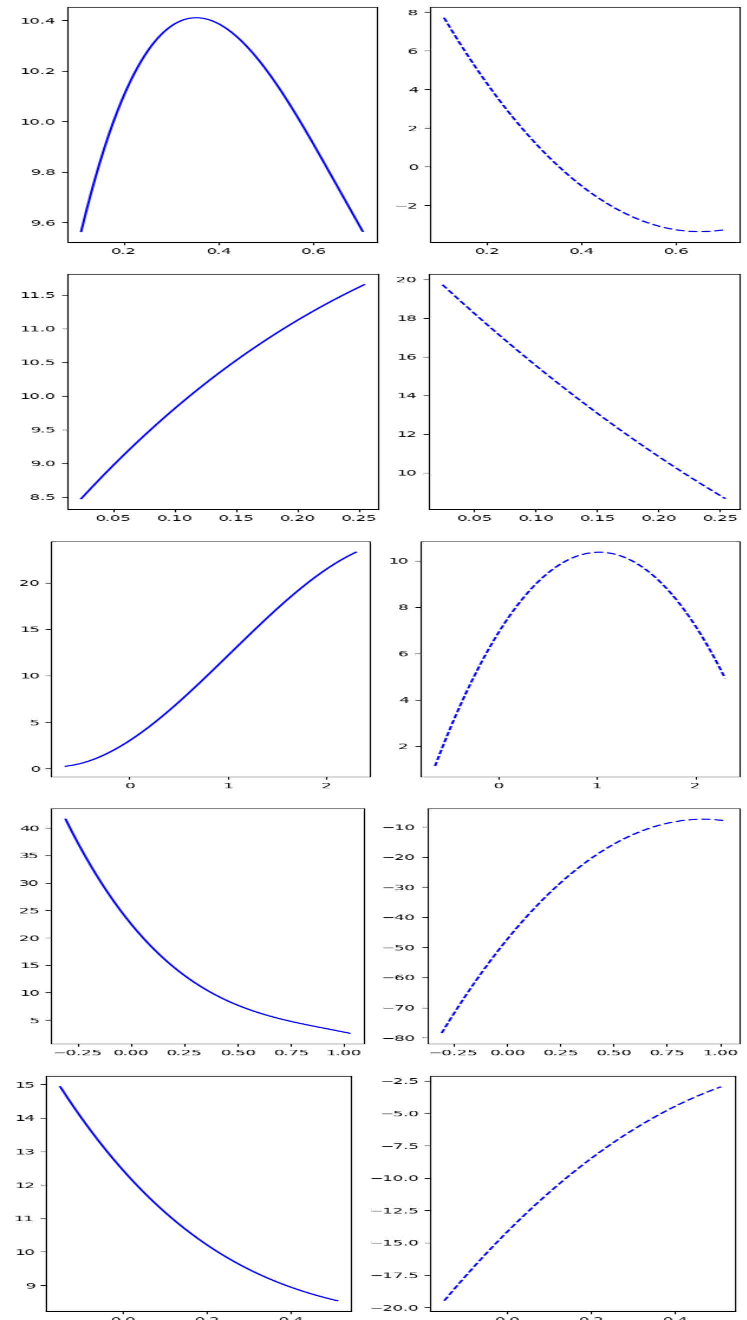


Figure 4.4: Mean response and functional derivatives against each covariate for Abalone data (from top to bottom: Diameter, Height, Whole weight, Shucked weight, Viscera weight, Shell weight). For each subplot, the left plot (solid line) is the mean response and the right plot (dashed line) is the partial derivative.

both mean response and function derivative, has a ”turning point”. Except for Frequency, the turning point in functional derivatives, very close to the origin, result in the flat middle of mean response curves.

4.7 Discussion

In this chapter, we propose estimating target function upon a network architecture of sparsely-connected deep neural networks with the Rectified Quadratic Unit (ReQU) activation function, through an empirical risk minimization framework and apply regularization to prevent possible over-fitting. Our proposed neural network bases on Jacobi polynomial approximation on the hyperbolic cross/sparse grid. Our framework can be applied to both regression and function derivative estimation. Our proposed method is a remedy to smooth functional derivative estimation since the estimators are all represented by smooth functions. Practically, we illustrate the proposed method through simulation studies and two real data applications. In general, our proposed method provides a reliable solution for mitigating the curse of dimensionality for modern large-scale data analysis. As a future work, we will investigate the statistical properties of our estimator.

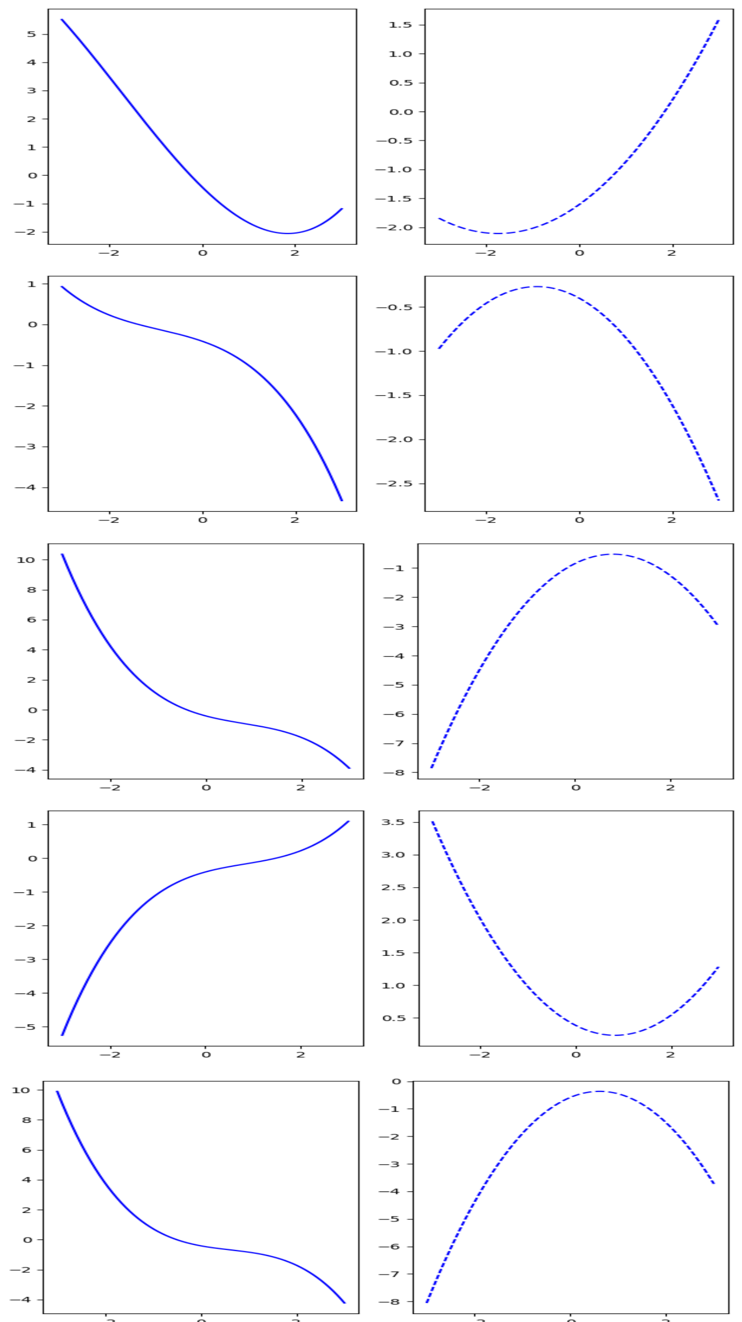


Figure 4.5: Mean response and functional derivatives against each covariate for airfoil data (from top to bottom: Frequency, Angle of attack, Chord length, Free-stream velocity, Suction side displacement thickness). For each subplot, the left plot (solid line) is the mean response and the right plot (dashed line) is the partial derivative.

Chapter 5

Conclusions

In this dissertation, we mainly illustrate artificial neural network techniques applied in causal inference (Chapter 3) and sparse deep neural network regression and functional derivative estimation (Chapter 4).

In Chapter 3, we provide a unified framework for efficient estimation of various types of TEs in observational data with a diverging number of covariates through a generalized optimization. The resulting TE estimator only involves the estimate of one nuisance function, which is approximated by fully connected ANNs with one hidden layer. Theoretically, we show that the number of confounders is allowed to increase with the sample size, and further investigate how fast it can grow with the sample size to ensure root-n consistency of the resulting TE estimator. Moreover, we establish asymptotic normality and semiparametric efficiency of the TE estimator. We have shown that the ANNs with one hidden layer can circumvent the "curse of dimensionality" and the resulting TE estimators enjoy root-n consistency under the condition that the target function is in a mixed

Sobolev space. Our new results advance the understanding of the required conditions and the statistical properties for ANNs in causal inference, and lay a theoretical foundation to demonstrate that ANNs are promising tools for causality analysis when the dimension is allowed to diverge, whereas most existing works on ANNs estimation still assume the dimension to be fixed. Practically, we illustrate the proposed method through simulation studies and a real data example. The numerical studies support our theoretical findings.

In Chapter 4, we focus on sparse deep neural network regression and functional derivative estimation. We propose a penalized deep ReQU network estimator (PDRN) obtained from empirical risk minimization framework. The proposed neural network bases on Jacobi polynomial approximation on the hyperbolic cross/sparse grid and alleviates the "curse of dimensionality". Our estimator provides a remedy to smooth functional derivative estimation. Practically, we illustrate the proposed method through simulation studies and two real data examples.

Bibliography

[1] Martin Abadi, Ashish Agarwal, Paul Barham, and et al. Tensorflow: Large-scale machine learning on heterogeneous systems. *http://arxiv.org/abs/1603.04467*, 2016.

[2] Alberto Abadie and Guido W Imbens. Large sample properties of matching estimators for average treatment effects. *econometrica*, 74(1):235–267, 2006.

[3] Chunrong Ai, Oliver Linton, Kaiji Motegi, and Zheng Zhang. A unified framework for efficient estimation of general treatment models. *arXiv preprint arXiv:1808.04936*, 2018.

[4] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):1–74, 2021.

[5] Donald W. K. Andrews. Asymptotic normality of series estimators for nonparametric and semiparametric regression models. *Econometrica: Journal of the Econometric Society*, 59:307–345, 1991.

[6] Donald WK Andrews. Empirical process methods in econometrics. *Handbook of Econometrics*, 4:2247–2294, 1994.

[7] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

[8] Martin Anthony, Peter L Bartlett, Peter L Bartlett, et al. *Neural network learning: Theoretical foundations*, volume 9. cambridge university press Cambridge, 1999.

[9] S. Athey, J. Tibshirani, and S. Wager. Generalized random forests. *Annals of Statistics*, 47(523):1148–1178, 2019.

[10] Susan Athey, Guido W Imbens, and Stefan Wager. Approximate residual balancing: debiased inference of average treatment effects in high dimensions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(4):597–623, 2018.

[11] H. Bang and J. M. Robins. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61:962–973, 2005.

[12] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[13] Andrew R Barron and Jason M Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv preprint arXiv:1809.03090*, 2018.

[14] Benedikt Bauer and Michael Kohler. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics*, 47(4):2261–2285, 2019.

[15] B. Bauker and M. Kohler. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics*, 47(4):2261–2285, 2019.

[16] R. Bellman. *Curse of dimensionality. Adaptive control processes: a guided tour.* Princeton, NJ, 1961.

[17] A Belloni, V Chernozhukov, and C Hansen. Inference on treatment effects after selection amongst high-dimensional controls. *The Review of Economic Studies*, 81(2):608–650, 2014.

[18] Alexandre Belloni, Daniel Chen, Victor Chernozhukov, and Christian Hansen. Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica*, 80(6):2369–2429, 2012.

[19] Alexandre Belloni, Victor Chernozhukov, Ivan Fernández-Val, and Christian Hansen. Program evaluation and causal inference with high-dimensional data. *Econometrica*, 85(1):233–298, 2017.

[20] Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on treatment effects after selection among high-dimensional controls. *The Review of Economic Studies*, 81(2):608–650, 2014.

[21] Alexandre Belloni, Victor Chernozhukov, and Lie Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[22] Peter J Bickel, Ya'acov Ritov, Alexandre B Tsybakov, et al. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[23] Alan S Blinder. Wage discrimination: reduced form and structural estimates. *Journal of Human Resources*, 8(4):436–455, 1973.

[24] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[25] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[26] W. Cao, A. A. Tsiatis, and M. Davidian. Improving efficiency and robustness of the doubly robust estimator for a population mean with incomplete data. *Biometrika*, 96:723–734, 2009.

[27] Matias D Cattaneo. Efficient semiparametric estimation of multi-valued treatment effects under ignorability. *Journal of Econometrics*, 155(2):138–154, 2010.

[28] K. C. G. Chan and S. C. P. Yam. Oracle, multiple robust and multipurpose calibration in a missing response problem. *Statistical Science*, 29:380–396, 2014.

[29] Kwun Chuen Gary Chan, Sheung Chi Phillip Yam, and Zheng Zhang. Globally efficient non-parametric inference of average treatment effects by empirical balancing calibration weighting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(3):673–700, 2016.

[30] Xiaohong Chen. Large sample sieve estimation of semi-nonparametric models. *Handbook of Econometrics*, 6(B):5549–5632, 2007.

[31] Xiaohong Chen, Han Hong, Alessandro Tarozzi, et al. Semiparametric efficiency in gmm models with auxiliary data. *The Annals of Statistics*, 36(2):808–843, 2008.

[32] Xiaohong Chen, Oliver Linton, and Ingrid Van Keilegom. Estimation of semiparametric models when the criterion function is not smooth. *Econometrica*, 71(5):1591–1608, 2003.

[33] Xiaohong Chen, Ying Liu, Shujie Ma, and Zheng Zhang. Efficient estimation of general treatment effects using neural networks with a diverging number of confounders. *arXiv preprint arXiv:2009.07055*, 2020.

[34] Xiaohong Chen and Xiaotong Shen. Sieve extremum estimates for weakly dependent data. *Econometrica*, pages 289–314, 1998.

[35] Xiaohong Chen and Halbert White. Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory*, 45(2):682–691, 1999.

[36] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters, 2018.

[37] Kyle Colangelo and Ying-Ying Lee. Double debiased machine learning nonparametric inference with continuous treatments, 2021.

[38] Felipe Cucker and Ding Xuan Zhou. *Learning theory: an approximation theory viewpoint*, volume 24. Cambridge University Press, 2007.

[39] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[40] P. Ding and F. Li. Causal inference: a missing data perspective. *Statistical Science*, 33:214–237, 2018.

[41] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[42] Alexander D'Amour, Peng Ding, Avi Feller, Lihua Lei, and Jasjeet Sekhon. Overlap in observational studies with high-dimensional covariates. *Journal of Econometrics*, 221(2):644–654, 2021.

[43] Hasan Fallahgoul, Vincentius Franstianto, and Gregoire Loeper. Towards explaining the relu feed-forward network. *Available at SSRN*, 2019.

[44] Jianqing Fan and Irene Gijbels. *Local polynomial modelling and its applications.* Routledge, 2018.

[45] Max H Farrell. Robust inference on average treatment effects with possibly more covariates than observations. *Journal of Econometrics*, 189(1):1–23, 2015.

[46] Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *arXiv preprint arXiv:1809.09953*, 2019.

[47] Sergio Firpo. Efficient semiparametric estimation of quantile treatment effects. *Econometrica*, 75(1):259–276, 2007.

[48] Yuri Fonseca, Marcelo Medeiros, Gabriel Vasconcelos, and Alvaro Veiga. Boost: Boosting smooth trees for partial effect estimation in nonlinear regressions. *arXiv preprint arXiv:1808.03698*, 2018.

[49] D. A. Freedman and R. A. Berk. Weighting regression by propensity scores. *Evaluation Review*, 32:392–409, 2008.

[50] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[52] Michael Griebel and Jan Hamaekers. Sparse grids for the schrödinger equation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(2):215–247, 2007.

[53] Michael Griebel and Stephan Knapek. Optimized tensor-product approximation spaces. *Constructive Approximation*, 16(4):525–540, 2000.

[54] J. Hahn. On the role of the propensity score in efficient semiparametric estimation of average treatment effects. *Econometrica*, 66(2):315–331, 1998.

[55] P. Han. "a general framework for quantile estimation with incomplete data. *Statistica Sinica*, 28:1307–1332, 2018.

[56] P. Han, L. Kong, and J. Zhao. A general framework for quantile estimation with incomplete data. *Journal of the Royal Statistical Society, Series B*, 81:305–333, 2019.

[57] Ben B Hansen. Full matching in an observational study of coaching for the sat. *Journal of the American Statistical Association*, 99(467):609–618, 2004.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[60] James J Heckman, Hidehiko Ichimura, and Petra Todd. Matching as an econometric evaluation estimator. *The Review of Economic Studies*, 65(2):261–294, 1998.

[61] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[62] K. Hirano, G. W. Imbens, and G. Ridder. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica*, 71(4):1161–1189, July 2003.

[63] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[64] Han Hong, Michael P Leung, and Jessie Li. Inference on finite-population treatment effects under limited overlap. *The Econometrics Journal*, 23(1):32–47, 2020.

[65] Kurt Hornik, Maxwell Stinchcombe, Halbert White, and Peter Auer. Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Computation*, 6(6):1262–1275, 1994.

[66] Joel L Horowitz and Enno Mammen. Rate-optimal estimation for a general class of nonparametric regression models with unknown link functions. *The Annals of Statistics*, 35(6):2589–2619, 2007.

[67] Ming-Yueh Huang and Kwun Chuen Gary Chan. Joint sufficient dimension reduction and estimation of conditional and average treatment effects. *Biometrika*, 104(3):583–596, 2017.

[68] Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *The Journal of Machine Learning Research*, 15(1):2869–2909, 2014.

[69] J. D.Y. Kang and J. L. Schafer. Demystifying double robustness: a comparison of alternative strategies for estimating a population mean from incomplete data. *Statistical Science*, 22:523–539, 2007.

[70] E. Kennedy, Z. Ma, M. McHugh, and D. Small. Nonparametric methods for doubly robust estimation of continuous treatment effects. *Journal of the Royal Statistical Society: Series B*, 79:1229–1245, 2017.

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

[72] Jason M Klusowski and Andrew R Barron. Approximation by combinations of relu and squared relu ridge functions with l_1 and l_0 controls. *IEEE Transactions on Information Theory*, 64(12):7649–7656, 2018.

[73] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[74] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[75] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*, volume 23. Springer Science & Business Media, 1991.

[76] Ying-Ying Lee. Efficient propensity score regression estimators of multivalued treatment effects for the treated. *Journal of Econometrics*, 204(2):207–222, 2018.

[77] Bo Li, Shanshan Tang, and Haijun Yu. Better approximations of high dimensional smooth functions by deep neural networks with rectified power units. *arXiv preprint arXiv:1903.05858*, 2019.

[78] Qi Li and Jeffrey Scott Racine. *Nonparametric econometrics: theory and practice.* Princeton University Press, 2007.

[79] Z. C. Lipton. The mythos of model interpretability. *Queue - Machine Learning*, 16(30), 2018.

[80] Yu Liu and Kris De Brabanter. Smoothed nonparametric derivative estimation using weighted difference quotients. *J. Mach. Learn. Res.*, 21:65–1, 2020.

[81] GG Lorentz. Approximation of functions, athena series. *Selected Topics in Mathematics*, 1966.

[82] W. Luo, Y. Zhu, and D. Ghosh. On estimating regression-based causal effects using sufficient dimension reduction. *Biometrika*, 104:51–65, 2017.

[83] Shujie Ma, Jeffrey S Racine, and Aman Ullah. Nonparametric estimation of marginal effects in regression-spline random effects models. *Econometric Reviews*, 39(8):792–825, 2020.

[84] Shujie Ma, Jeffrey S Racine, and Lijian Yang. Spline regression in the presence of categorical predictors. *Journal of Applied Econometrics*, 30(5):705–717, 2015.

[85] Shujie Ma, Liping Zhu, Zhiwei Zhang, Chih-Ling Tsai, and Raymond J Carroll. A robust and efficient approach to causal inference based on sparse sufficient dimension reduction. *Annals of statistics*, 47(3):1505, 2019.

[86] Xinwei Ma and Jingshen Wang. Robust inference using inverse probability weighting. *Journal of the American Statistical Association*, 115(532):1851–1860, 2020.

[87] Pascal Massart. About the constants in talagrand's concentration inequalities for empirical processes. *The Annals of Probability*, 28(2):863–884, 2000.

[88] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[89] Hrushikesh N Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8(1):164–177, 1996.

[90] Hadrien Montanelli and Qiang Du. New error bounds for deep relu networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92, 2019.

[91] Whitney K Newey and Daniel McFadden. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245, 1994.

[92] Whitney K Newey and James L Powell. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, 55:819–847, 1987.

[93] Whitney K Newey and James R Robins. Cross-fitting and fast remainder rates for semiparametric estimation. *arXiv preprint arXiv:1801.09138*, 2018.

[94] Y. Ning, S. Peng, and K. Imai. Robust estimation of causal effects via a high-dimensional covariate balancing propensity score. *Biometrika*, 107(3):533—554, 2020.

[95] Ronald Oaxaca. Male-female wage differentials in urban labor markets. *International economic review*, 14(3):693–709, 1973.

[96] Ariel Pakes and David Pollard. Simulation and the asymptotics of optimization estimators. *Econometrica: Journal of the Econometric Society*, pages 1027–1057, 1989.

[97] Cheolwoo Park and Kee-Hoon Kang. Sizer analysis for the comparison of regression curves. *Computational Statistics & Data Analysis*, 52(8):3954–3970, 2008.

[98] James M Robins, Andrea Rotnitzky, and Lue Ping Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American statistical Association*, 89(427):846–866, 1994.

[99] Vitaliana Rondonotti, JS Marron, and Cheolwoo Park. Sizer for time series: a new approach to the analysis of trends. *Electronic Journal of Statistics*, 1:268–289, 2007.

[100] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[101] A. Rotnitzky, Q. Lei, M. Sued, and J. M. Robins. Improved double-robust estimation in missing data and causal inference models. *Biometrika*, 99:439–456, 2012.

[102] J. Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *Annals of Statistics*, 48:1875–1897, 2020.

[103] Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral methods: algorithms, analysis and applications*, volume 41. Springer Science & Business Media, 2011.

[104] Jie Shen and Li-Lian Wang. Sparse spectral approximations of high-dimensional problems based on hyperbolic cross. *SIAM Journal on Numerical Analysis*, 48(3):1087–1109, 2010.

[105] Xiaotong Shen et al. On methods of sieves and penalization. *The Annals of Statistics*, 25(6):2555–2591, 1997.

[106] Xiaotong Shen and Jian Shi. Sieve likelihood ratio inference on general parameter space. *Science in China Series A: Mathematics*, 48(1):67–78, 2005.

[107] J Andrew Simpkin, María Luz Durbán Reguera, Debbie A Lawlor, Corrie Macdonald-Wallis, Margaret T May, Chris Metcalfe, and Kate Tilling. Derivative estimation for longitudinal data analysis. 2018.

[108] Charles J Stone. Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, pages 1040–1053, 1982.

[109] Charles J Stone. The use of polynomial splines and their tensor products in multivariate function estimation. *The Annals of Statistics*, 22(1):118–171, 1994.

[110] Z. Tan. Bounded, efficient and doubly robust estimation with inverse weighting. *Biometrika*, 97:661–682, 2010.

[111] Z. Tan. Model-assisted inference for treatment effects using regularized calibrated estimation with high-dimensional data. *Annals of Statistics*, 48:811–837, 2020.

[112] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[113] Anastasios Tsiatis. *Semiparametric Theory and Missing Data*. Springer Science & Business Media, 2007.

[114] Sara Van de Geer, Peter Bühlmann, Ya'acov Ritov, Ruben Dezeure, et al. On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics*, 42(3):1166–1202, 2014.

[115] M. J. van der Laan and J. M. Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer, New York., 2003.

[116] M. J. van der Laan and S. Rose. *Targeted Learning: Causal Inference for Observational and Experimental Data*. Springer, New York., 2011.

[117] A. W. van der Vaart. *Asymptotic statistics*. Cambridge University Press, 1998.

[118] S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

[119] Qingcan Wang et al. Exponential convergence of the deep neural network approximation for analytic functions. *arXiv preprint arXiv:1807.00297*, 2018.

[120] Larry Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[121] Halbert White. Learning in artificial neural networks: A statistical perspective. *Neural computation*, 1(4):425–464, 1989.

[122] Halbert White. *Artificial neural networks: approximation and learning theory*. Blackwell Publishers, Inc., 1992.

[123] Patrick Henry Winston. *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[124] Q. Yao and H. Tong. Asymmetric least squares regression estimation: A nonparametric approach. *Journal of Nonparametric Statistics*, 6:2–3, 1996.

[125] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[126] Dmitry Yarotsky. Optimal approximation of continuous functions by very deep relu networks. *arXiv preprint arXiv:1802.03620*, 2018.

[127] Hui Zou, Ji Zhu, and Trevor Hastie. The margin vector, admissible loss and multiclass marginbased classifiers. *School of Statistics, University of Minnesota, Technical report*, 2006.

Appendix A

Supplementary Materials for Chapter 3

A.1 Polynomial Approximation and Curse of Dimensionality

We consider to approximate a target function $f(\cdot) \in \mathcal{W}^{m,\delta_0,\infty}(\mathcal{X})$ using polynomial series. Denote $s_0 := m + \delta_0$. Since $f(\cdot) \in \mathcal{W}^{m,\delta_0,\infty}(\mathcal{X})$, the partial derivatives of $f(x)$ up to s_0-times are uniformly bounded, i.e. $\sup_{|\boldsymbol{\alpha}|_1 \leq s_0} \sup_{x \in \mathcal{X}} |D^{\boldsymbol{\alpha}} f(x)| \leq 1$. By [81], there exist polynomials $P_{n_1,\ldots,n_p}(x)$, of degree n_i in x_i, for which

$$\sup_{x \in \mathcal{X}} |f(x) - P_{n_1,\ldots,n_p}(x)| \leq C_p \cdot \sum_{i=1}^{p} \frac{1}{n_i^{s_0}},$$

where C_p is a constant depending on p.

Consider a K-dimensional polynomial sieve $\{u_K(x)\}$ of the form:

$$u_1(x) = 1,\ u_2(x) = (1, x_1)^\top, \ldots, u_{p+1}(x) = (1, x_1, ..., x_p)^\top,\ u_{p+2}(x) = (1, x_1, ..., x_p, x_1^2)^\top, ...$$

To ensure all degrees of $(x_1, ..., x_p)$ get up to s_0, we require $K = (s_0 + 1)^p$. Then

$$\inf_{\lambda_K \in \mathbb{R}^K} \sup_{x \in \mathcal{X}} |f(x) - \lambda_K^\top u_K(x)| \leq C_p \cdot p \cdot K^{-\frac{s_0}{p}}.$$

Since $\mathcal{X}$ is compact, which implies that the $L^2(dF_X)$ approximation error is also of $O(C_p \cdot p \cdot K^{-\frac{s_0}{p}})$. In comparison with the approximation rates using ANN, (3.5), the approximation based on the polynomial sieve suffers from the curse of dimensionality.

A.2 Proof of the Asymptotic Distribution of the Proposed TE Estimator

We first prove $\widehat{\beta}_d \xrightarrow{p} \beta_d^*$ for any $d \in \{0, 1, ..., J\}$. Because $\widehat{\beta}_d$ (resp. β_d^*) is a minimizer of $n^{-1} \sum_{i=1}^{n} D_{di} \mathcal{L}(Y_i - \beta_d) / \widehat{\pi}_d(\boldsymbol{X}_i)$ (resp. $\mathbb{E}[D_{di} \mathcal{L}(Y_i - \beta) / \pi_d^*(\boldsymbol{X}_i)]$), from the theory of M-estimation [117, Theorem 5.7], if the following uniform convergence holds:

$$\sup_{\beta\in\Theta}\left|\frac{1}{n}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}\left(Y_i-\beta\right)-\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}\left(Y_i-\beta\right)\right]\right|\xrightarrow{p}0.$$

then $\widehat{\beta}_d \xrightarrow{p} \beta_d^*$. We start to verify above condition. Using the triangular inequality, we have

$$\begin{aligned}&\sup_{\beta\in\Theta}\left|\frac{1}{n}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}\left(Y_i-\beta\right)-\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}\left(Y_i-\beta\right)\right]\right|\\ \leq&\sup_{\beta\in\Theta}\left|\frac{1}{n}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)\pi_d^*(\boldsymbol{X}_i)}\left\{\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)\right\}\mathcal{L}\left(Y_i-\beta\right)\right| \quad \text{(A.1)}\\ &+\sup_{\beta\in\Theta}\left|\frac{1}{N}\sum_{i=1}^{N}\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}\left(Y_i-\beta\right)-\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}\left(Y-\beta\right)\right]\right|. \quad \text{(A.2)}\end{aligned}$$

We first show (A.1) is of $o_P(1)$. By Theorem 1, Cauchy-Scharwz' inequality, and Assumptions 5 and 8, we have

$$\begin{aligned}|(\text{A.1})|\leq&O_p(1)\cdot\left\{\frac{1}{n}\sum_{i=1}^{n}\left\{\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)\right\}^2\right\}^{1/2}\cdot\sup_{\beta\in\Theta}\left\{\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\left(Y_i-\beta\right)^2\right\}^{1/2}\\ \leq&o_P(1)\cdot\left\{\sup_{\beta\in\Theta}\mathbb{E}\left[\mathcal{L}\left(Y-\beta\right)^2\right]+o_P(1)\right\}^{1/2}=o_P(1),\end{aligned}$$

where the first inequality holds because $\widehat{\pi}_d(x)$ is uniformly bounded away from 0 and 1 with probability approaching to 1 since $\widehat{\pi}_d(x)\xrightarrow{p}\pi_d^*(x)$ uniformly in $x\in\mathcal{X}$ and $0<\underline{c}\leq\pi_d^*(x)$ by Assumption 5. To show (A.2) is of $o_P(1)$, by [91, Lemma 2.4], it is sufficient to verify the following conditions holds true:

1. Θ is compact;
2. $\mathcal{L}'(Y_i-\beta)$ is continuous in β with probability one;
3. $\mathbb{E}\left[\sup_{\beta\in\Theta}|\mathcal{L}(Y-\beta)|\right]<\infty$;

which are imposed in Assumptions 4 and 8. Hence, $\widehat{\beta}_d\xrightarrow{P}\beta_d^*$ holds.

Next, we establish the asymptotic normality for $\sqrt{n}\{\widehat{\beta}_d-\beta_d^*\}$. Since the loss function $\mathcal{L}(\cdot)$ may not be smooth (e.g. $\mathcal{L}(v)=v\{\tau-\mathbb{1}(v\leq 0)\}$ in quantile regression), the Delta method for deriving the large sample property is not applicable in our case. To circumvent this problem, we apply the *nearness of arg mins* argument. Define

$$G_{d,n}(\beta,\widehat{\pi}_d):=\frac{1}{n}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}(Y_i-\beta). \quad \text{(A.3)}$$

By definition

$$\widehat{\beta}_d = \arg\min_{\beta\in\Theta} G_{d,n}(\beta, \widehat{\pi}_d) = \arg\min_{\beta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}(Y_i - \beta), \tag{A.4}$$

then

$$\begin{aligned}\widehat{\beta}_d &= \arg\min_{\beta\in\Theta} n\left\{G_{d,n}(\beta,\widehat{\pi}_d) - G_{d,n}(\beta_d^*,\widehat{\pi}_d)\right\} = \arg\min_{\beta\in\Theta}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\left\{\mathcal{L}(Y_i-\beta) - \mathcal{L}(Y_i - \beta_d^*)\right\}\\ &= \arg\min_{\beta\in\Theta}\sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\left[-\mathcal{L}'(Y_i-\beta_d^*)(\beta-\beta_d^*) + \left\{\mathcal{L}(Y_i-\beta) - \mathcal{L}(Y_i-\beta_d^*) + \mathcal{L}'(Y_i-\beta_d^*)(\beta-\beta_d^*)\right\}\right].\end{aligned}$$

By using change of variables and defining the following functions:

$$\begin{aligned}&\widehat{u}_d := \sqrt{n}(\widehat{\beta}_d - \beta_d^*),\ u := \sqrt{n}(\beta - \beta_d^*),\\ &R_d(Y_i, u) := \mathcal{L}\left(Y_i - \left\{\beta_d^* + \frac{u}{\sqrt{n}}\right\}\right) - \mathcal{L}(Y_i - \beta_d^*) + \mathcal{L}'(Y_i - \beta_d^*)\cdot\frac{u}{\sqrt{n}},\\ &Q_{d,n}(u,\widehat{\pi}_d) := \sum_{i=1}^{n}\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\left[-\mathcal{L}'(Y_i-\beta_d^*)\cdot\frac{u}{\sqrt{n}} + R_d(Y_i,u)\right].\end{aligned}$$

Then we get

$$\widehat{u}_d = \arg\min_u Q_{d,n}(u,\widehat{\pi}_d).$$

Next, we define the following quadratic function

$$\widetilde{Q}_{d,n}(u) := \frac{u}{\sqrt{n}}\sum_{i=1}^{n}\left[-\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}'(Y_i-\beta_d^*) + \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1\right)\mathcal{E}_d(\boldsymbol{X}_i,\beta_d^*)\right] - \partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]\cdot\frac{u^2}{2},$$

which does not depend on $\widehat{\pi}_d$, and its minimizer is defined by

$$\widetilde{u}_d := \arg\min_u \widetilde{Q}_{d,n}(u,\widehat{\pi}_d).$$

Since $\widetilde{Q}_{d,n}(u)$ strictly convex and $\partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)] > 0$, then the minimizer $\widetilde{u}_d$ is equal to

$$\widetilde{u}_d = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} S_d(Y_i, D_{di}, \boldsymbol{X}_i;\beta_d^*),$$

where

$$S_d(Y_i, D_{di}, \boldsymbol{X}_i;\beta_d^*) := H_d^{-1}\cdot\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}'(Y_i-\beta_d^*) - \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1\right)\mathcal{E}_d(\boldsymbol{X}_i,\beta_d^*)\right]$$

is the influence function of β_d^* and $H_d := -\partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]$.

We complete the proof via the following steps:

- Step I: showing $\widetilde{Q}_{d,n}(u) - Q_{d,n}(u, \widehat{\pi}_d) = o_p(1)$ for every fixed u;
- Step II: showing $|\widehat{u}_d - \widetilde{u}_d| = o_P(1)$;
- Step III: obtaining the desired result $\sqrt{n}\{\widehat{\beta}_d - \beta_d^*\} = \widetilde{u}_d + \{\widehat{u}_d - \widetilde{u}_d\} = n^{-1/2}\sum_{i=1}^n S_d(Y_i, D_{di}, \boldsymbol{X}_i; \beta_d^*) + o_P(1)$.

We begin to establish Step I by showing that $\widetilde{Q}_{d,n}(u)$ is a quadratic approximation to the objective function $Q_{d,n}(u, \widehat{\pi}_d)$. We write the absolute value of the difference as follows:

$$\begin{aligned}
&\left|\widetilde{Q}_{d,n}(u) - Q_{d,n}(u, \widehat{\pi}_d)\right| \\
=&\Bigg|\frac{u}{\sqrt{n}}\sum_{i=1}^n \left[\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}'(Y_i - \beta_d^*) - \frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}'(Y_i - \beta_d^*) + \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1\right)\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)\right] \\
&-\sum_{i=1}^n \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}R_d(Y_i, u) + \partial_\beta \mathbb{E}[\mathcal{L}'(Y_i^*(d) - \beta_d^*)] \cdot \frac{u^2}{2}\Bigg| \\
\le& |\xi_{1,n}(u, \widehat{\pi}_d)| + |\xi_{2,n}(u, \widehat{\pi}_d)|,
\end{aligned}$$

where

$$\begin{aligned}
\xi_{1,n}(u, \widehat{\pi}_d) &:= \frac{u}{\sqrt{n}}\sum_{i=1}^n \left[\frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}\mathcal{L}'(Y_i - \beta_d^*) - \frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}'(Y_i - \beta_d^*) + \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1\right)\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)\right], \\
\xi_{2,n}(u, \widehat{\pi}_d) &:= \sum_{i=1}^n \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)}R_d(Y_i, u) - \partial_\beta \mathbb{E}[\mathcal{L}'(Y_i^*(d) - \beta_d^*)] \cdot \frac{u^2}{2}.
\end{aligned}$$

Then Step I holds if we prove that for every u,

$$\xi_{1,n}(u, \widehat{\pi}_d) = o_P(1), \tag{A.5}$$
$$\xi_{2,n}(u, \widehat{\pi}_d) = o_P(1). \tag{A.6}$$

We begin to establish (A.5). By Taylor's expansion, we have

$$\begin{aligned}
\xi_{1,n}(u, \widehat{\pi}_d) =& -\frac{u}{\sqrt{n}}\sum_{i=1}^n \frac{D_{di}}{\{\pi_d^*(\boldsymbol{X}_i)\}^2}\mathcal{L}'\left(Y_i - \beta_d^*\right)\{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\} \\
&+\frac{u}{\sqrt{n}}\sum_{i=1}^n \frac{D_{di}}{\{\widetilde{\pi}_d(\boldsymbol{X}_i)\}^3}\mathcal{L}'\left(Y_i - \beta_d^*\right)\{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}^2 \\
&+\frac{u}{\sqrt{n}}\sum_{i=1}^n \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1\right)\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*) \\
=& -u \cdot \mu_n\left(\frac{D_{di}}{\{\pi_d^*(\boldsymbol{X}_i)\}^2}\mathcal{L}'\left(Y_i - \beta_d^*\right)\{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}\right) && \text{(A.7)} \\
&+\frac{u}{\sqrt{n}}\sum_{i=1}^n \frac{D_{di}}{\{\widetilde{\pi}_d(\boldsymbol{X}_i)\}^3}\mathcal{L}'\left(Y_i - \beta_d^*\right)\{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}^2 && \text{(A.8)} \\
&+\frac{u}{\sqrt{n}}\sum_{i=1}^n \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1\right)\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*) - u \cdot \sqrt{n} \cdot \mathbb{E}\left[\frac{\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)}{\pi_d^*(\boldsymbol{X}_i)}\{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}\right] \\
& && \text{(A.9)}
\end{aligned}$$

where $\widetilde{\pi}_d(\boldsymbol{X}_i)$ is between $\widehat{\pi}_d(\boldsymbol{X}_i)$ is $\pi_d^*(\boldsymbol{X}_i)$. For (A.7), similar to the proof of Lemma 1 in the online Supplementay Materials [33] and using Assumption 5 (ii), we have that (A.7) is of $o_P(1)$.

For (A.8), by Theorem 1, we have

$$
\begin{aligned}
|(\text{A.8})| = & \left| \frac{u}{\sqrt{n}} \sum_{i=1}^{n} \frac{D_{di}}{\{\widetilde{\pi}_d(\boldsymbol{X}_i)\}^3} \mathcal{L}'\left(Y_i - \beta\right) \{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}^2 \right| \\
\leq & O_P(\sqrt{n}) \cdot \frac{u}{n} \sum_{i=1}^{n} |\mathcal{L}'(Y_i - \beta_d^*)| \cdot \{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}^2 \\
= & O_P(\sqrt{n}) \cdot u \cdot \mathbb{E}\left[|\mathcal{L}'(Y_i - \beta_d^*)| \cdot \{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}^2\right] \cdot \{1 + o_P(1)\} \\
\leq & O_P(\sqrt{n}) \cdot u \cdot \sup_{x \in \mathcal{X}} \mathbb{E}[|\mathcal{L}'(Y - \beta_d^*)||\boldsymbol{X} = x] \cdot \mathbb{E}\left[\{\widehat{\pi}_d(\boldsymbol{X}_i) - \pi_d^*(\boldsymbol{X}_i)\}^2\right] \cdot \{1 + o_P(1)\} \\
= & O_P(\sqrt{n}) \cdot u \cdot O(1) \cdot o_P(n^{-1/2}) \cdot O_P(1) = o_P(1).
\end{aligned}
$$

For the term (A.9), we have

$$
\begin{aligned}
(\text{A.9}) = & \frac{u}{\sqrt{n}} \sum_{i=1}^{n} \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1 \right) \mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*) - u \cdot \sqrt{n} \cdot \mathbb{E}\left[\frac{\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)}{\pi_d^*(\boldsymbol{X}_i)} \{L\left(\widehat{g}_d(\boldsymbol{X}_i)\right) - L\left(g_d^*(\boldsymbol{X}_i)\right)\} \right] \\
= & \frac{u}{\sqrt{n}} \sum_{i=1}^{n} \left(\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)} - 1 \right) \mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*) - u \cdot \sqrt{n} \cdot \mathbb{E}\left[\frac{\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)}{\pi_d^*(\boldsymbol{X}_i)} L'\left(g_d^*(\boldsymbol{X}_i)\right) \{\widehat{g}_d(\boldsymbol{X}_i) - g_d^*(\boldsymbol{X}_i)\} \right] \qquad (\text{A.10}) \\
& - \frac{u}{2} \cdot \sqrt{n} \cdot \mathbb{E}\left[\frac{\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)}{\pi_d^*(\boldsymbol{X}_i)} L''\left(\widetilde{g}_d(\boldsymbol{X}_i)\right) \{\widehat{g}_d(\boldsymbol{X}_i) - g_d^*(\boldsymbol{X}_i)\}^2 \right], \qquad (\text{A.11})
\end{aligned}
$$

where $\widetilde{g}_d$ is between $\widehat{g}_d$ and g_d^*. We have that (A.10) is of $o_P(1)$ by Lemma 2 in the online Supplementay Materials [33] and the fact $L'\left(g_d^*(\boldsymbol{X}_i)\right) = L\left(g_d^*(\boldsymbol{X}_i)\right)\left(1 - L\left(g_d^*(\boldsymbol{X}_i)\right)\right) = \pi_d^*(\boldsymbol{X}_i)(1 - \pi_d^*(\boldsymbol{X}_i))$. For the term (A.11), we have

$$
\begin{aligned}
|(\text{A.11})| \leq & \frac{|u|}{2} \cdot \sqrt{n} \cdot \mathbb{E}\left[\frac{\mathcal{E}_d(\boldsymbol{X}_i, \beta_d^*)}{\pi_d^*(\boldsymbol{X}_i)} L\left(\widetilde{g}_d(\boldsymbol{X}_i)\right)\left(1 - L\left(\widetilde{g}_d(\boldsymbol{X}_i)\right)\right) \cdot |1 - 2L\left(\widetilde{g}_d(\boldsymbol{X}_i)\right)| \cdot \{\widehat{g}_d(\boldsymbol{X}_i) - g_d^*(\boldsymbol{X}_i)\}^2 \right] \\
\leq & |u| \cdot O(1) \cdot \sqrt{n} \cdot \mathbb{E}[\{\widehat{g}_d(\boldsymbol{X}_i) - g_d^*(\boldsymbol{X}_i)\}^2] = o_P(1),
\end{aligned}
$$

where the second inequality holds by Assumption 5 and the last equality holds by Lemma S.2 in the online Supplementay Materials [33]. Combining the results for (A.10) and (A.11), we obtain that (A.9) is of $o_P(1)$. Combining the results for (A.7), (A.8), and (A.9), we obtain (A.5).

Next, we prove (A.6). Note that

$$
\begin{aligned}
|\xi_{2,n}(u,\widehat{\pi}_d)| &= \left|\sum_{i=1}^n \frac{D_{di}}{\widehat{\pi}_d(\boldsymbol{X}_i)} R_d(Y_i,u) - \partial_\beta \mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]\cdot\frac{u^2}{2}\right| \\
&\le \left|\sum_{i=1}^n D_{di}R_d(Y_i,u)\frac{\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)}{\widehat{\pi}_d(\boldsymbol{X}_i)\pi_d^*(\boldsymbol{X}_i)}\right| \qquad \text{(A.12)} \\
&\quad + \left|\sum_{i=1}^n \frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}R_d(Y_i,u) - n\cdot\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}R_d(Y_i,u)\right]\right| \qquad \text{(A.13)} \\
&\quad + \left|n\cdot\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}R_d(Y_i,u)\right] - \partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]\cdot\frac{u^2}{2}\right|. \qquad \text{(A.14)}
\end{aligned}
$$

For (A.12), we have

$$
\begin{aligned}
&\left|\sum_{i=1}^n D_{di}R_d(Y_i,u)\frac{\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)}{\widehat{\pi}_d(\boldsymbol{X}_i)\pi_d^*(\boldsymbol{X}_i)}\right| \\
\le& n\cdot\sqrt{\frac{1}{n}\sum_{i=1}^n\left|\frac{D_{di}R_d(Y_i,u)}{\widehat{\pi}_d(\boldsymbol{X}_i)\pi_d^*(\boldsymbol{X}_i)}\right|^2}\cdot\sqrt{\frac{1}{n}\sum_{i=1}^n\left|\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)\right|^2} \\
=& n\cdot\sqrt{\mathbb{E}\left[\frac{D_{di}R_d(Y_i,u)^2}{\pi_d^*(\boldsymbol{X}_i)^4}\right]}\cdot\sqrt{\mathbb{E}\left[\left|\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)\right|^2\right]}\cdot\{1+o_P(1)\} \\
\le& n\cdot O(1)\cdot\sqrt{\mathbb{E}\left[R_d(Y_i^*(d),u)^2\right]}\cdot\sqrt{\mathbb{E}\left[\left|\widehat{\pi}_d(\boldsymbol{X}_i)-\pi_d^*(\boldsymbol{X}_i)\right|^2\right]}\cdot\{1+o_P(1)\} \\
=& n\cdot O(1)\cdot O\left(\frac{|u|^{3/2}}{n^{3/4}}\right)\cdot o_P(n^{-1/4})\cdot\{1+o_P(1)\} = |u|^{3/2}\cdot o_P(1),
\end{aligned}
$$

where the second inequality holds by assumption that $\pi_d^*(x)$ is uniformly bounded away from zero, the second equality holds because of Lemma 2 in the online Supplementay Materials [33], the first and second moments $R_d(Y_i^*(d),u)$ given by

$$
\begin{aligned}
\mathbb{E}\left[R_d(Y_i^*(d),u)\right] =& \mathbb{E}\left[\mathcal{L}\left(Y^*(d)-\left\{\beta_d^*+\frac{u}{\sqrt{n}}\right\}\right)\right] - \mathbb{E}\left[\mathcal{L}(Y^*(d)-\beta_d^*)\right] + \mathbb{E}[\mathcal{L}'(Y^*(d)-\beta_d^*)]\cdot\frac{u}{\sqrt{n}} \\
=& \frac{u^2}{2n}\cdot\partial_\beta\mathbb{E}\left[\mathcal{L}'\left(Y^*(d)-\left\{\beta_d^*+\frac{u^*}{\sqrt{n}}\right\}\right)\right]
\end{aligned}
$$

where u^* is between 0 and u, and

$$
\begin{aligned}
&\mathbb{E}\left[R_d(Y_i^*(d),u)^2\right] \\
=& \mathbb{E}\left[\left\{\mathcal{L}\left(Y_i^*(d)-\left\{\beta_d^*+\frac{u}{\sqrt{n}}\right\}\right) - \mathcal{L}(Y_i^*(d)-\beta_d^*) + \mathcal{L}'(Y_i^*(d)-\beta_d^*)\cdot\frac{u}{\sqrt{n}}\right\}^2\right] \\
=& \mathbb{E}\left[\left\{-\mathcal{L}'\left(Y_i^*(d)-\left\{\beta_d^*+\frac{\tilde{u}}{\sqrt{n}}\right\}\right)\cdot\frac{u}{\sqrt{n}} + \mathcal{L}'(Y_i^*(d)-\beta_d^*)\cdot\frac{u}{\sqrt{n}}\right\}^2\right] \\
\le& \text{const}\times\frac{|\overline{u}|}{\sqrt{n}}\frac{u^2}{n} \le \text{const}\times\frac{|u|^3}{n^{3/2}},
\end{aligned}
$$

where $\overline{u}$ is between 0 and u, and the first inequality holds by Assumption 8 (i).

For (A.13), by computing its second moment and using Chebyshev's inequality, we have

$$(\text{A.13}) = O_P\left(\sqrt{n}\cdot O\left(\frac{|u|^{3/2}}{n^{3/4}}\right)\right) = O_P(n^{-1/4}\cdot|u|^{3/2}) = o_P(|u|^{3/2}).$$

For (A.14), we have

$$\begin{aligned}(\text{A.14}) =& \left| n\cdot\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}R_d(Y_i,u)\right] - \partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]\cdot\frac{u^2}{2}\right| \\ =& \left| n\cdot\mathbb{E}\left[R_d(Y_i^*(d),u)\right] - \partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]\cdot\frac{u^2}{2}\right| \\ =& \left|\frac{u^2}{2}\cdot\partial_\beta\mathbb{E}\left[\mathcal{L}'\left(Y^*(d)-\left\{\beta_d^*+\frac{u^*}{\sqrt{n}}\right\}\right)\right] - \partial_\beta\mathbb{E}[\mathcal{L}'(Y_i^*(d)-\beta_d^*)]\cdot\frac{u^2}{2}\right| \\ =& o(u^2).\end{aligned}$$

Therefore, we obtain (A.6). So for fixed u,

$$\widetilde{Q}_{d,n}(u) - Q_{d,n}(u,\widehat{\pi}_d) = o_P(1). \tag{A.15}$$

To establish Step II, with (A.15), by using the same argument of proving Lemma A.5 in the supplemental material of [47], we can obtain $|\widehat{u}_d - \widetilde{u}_d| = o_P(1)$. Finally, we can conclude our desired result in Step III:

$$\sqrt{n}\{\widehat{\beta}_d - \beta_d^*\} = \widetilde{u} + \{\widehat{u}-\widetilde{u}\} = \frac{1}{\sqrt{n}}\sum_{i=1}^n S_d(Y_i, D_{di}, \boldsymbol{X}_i;\beta_d^*) + o_P(1).$$

A.3 Proof of Equation (3.12) for Variance Estimation

Using the tower property of conditional expectation, we rewrite H_d as:

$$\begin{aligned}H_d =& -\partial_\beta\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\cdot\mathbb{E}\left[\mathcal{L}'(Y_i-\beta)|D_{di},\boldsymbol{X}_i\right]\right]\Bigg|_{\beta=\beta_d^*} \\ =& -\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\cdot\partial_\beta\mathbb{E}\left[\mathcal{L}'(Y_i-\beta)|D_{di},\boldsymbol{X}_i\right]\Big|_{\beta=\beta_d^*}\right].\end{aligned}$$

Applying Leibniz integration rule, we obtain

$$
\begin{aligned}
&\partial_\beta \mathbb{E}\left[\mathcal{L}'(Y_i - \beta)|D_i = d, \boldsymbol{X}_i = x\right]\Big|_{\beta=\beta_d^*} \\
=&\partial_\beta \left[\int_{\mathbb{R}} \mathcal{L}'(y-\beta) f_{Y|D,X}(y|d,x)dy\right]\Bigg|_{\beta=\beta_d^*} \\
=&\partial_\beta \left[\int_{\mathbb{R}} \mathcal{L}'(z) f_{Y|D,X}(z+\beta|d,x)dz\right]\Big|_{\beta=\beta_d^*} \text{ (use } z = y-\beta)) \\
=&\int_{\mathbb{R}} \mathcal{L}'(z)\cdot \frac{\partial}{\partial y} f_{Y|D,X}(z+\beta_d^*|d,x)dz \\
=&\int_{\mathbb{R}} \mathcal{L}'(y-\beta_d^*)\cdot \frac{\partial}{\partial y} f_{Y|D,X}(y|d,x)dy \\
=&\int_{\mathbb{R}} \mathcal{L}'(y-\beta_d^*)\cdot \frac{\frac{\partial}{\partial y} f_{Y|D,X}(y|d,x)}{f_{Y|D,X}(y|d,x)} f_{Y|D,X}(y|d,x)dy \\
=&\int_{\mathbb{R}} \mathcal{L}'(y-\beta_d^*)\cdot \frac{\frac{\partial}{\partial y} f_{Y,X|D}(y,x|d)}{f_{Y,X|D}(y,x|d)} f_{Y|D,X}(y|d,x)dy \\
=&\mathbb{E}\left[\mathcal{L}'(Y-\beta_d^*)\frac{\frac{\partial}{\partial y} f_{Y,X|D}(Y_i,\boldsymbol{X}_i|d)}{f_{Y,X|D}(Y_i,X_{i,n}|d)}\middle| D_i = d, \boldsymbol{X}_i = x\right],
\end{aligned}
$$

and consequently

$$
H_d = -\mathbb{E}\left[\frac{D_{di}}{\pi_d^*(\boldsymbol{X}_i)}\mathcal{L}'(Y_i-\beta_d^*)\frac{\partial}{\partial y}\log f_{Y,X|D}(Y_i,\boldsymbol{X}_i|d)\right].
$$