Journal of Machine Learning Research 25 (2024) 1-75 Submitted 05/22; Revised 12/23; Published 03/24

Nonparametric Estimation of Non-Crossing Quantile Regression Process with Deep ReQU Neural Networks

Guohao Shen[*] GUOHAO.SHEN@POLYU.EDU.HK
Department of Applied Mathematics
The Hong Kong Polytechnic University
Hong Kong SAR, China

Yuling Jiao[*] YULINGJIAOMATH@WHU.EDU.CN
School of Mathematics and Statistics
and Hubei Key Laboratory of Computational Science
Wuhan University
Wuhan 430072, China

Yuanyuan Lin YLIN@STA.CUHK.EDU.HK
Department of Statistics
The Chinese University of Hong Kong
Hong Kong SAR, China

Joel L. Horowitz JOEL-HOROWITZ@NORTHWESTERN.EDU
Department of Economics
Northwestern University
Evanston, IL 60208, USA

Jian Huang J.HUANG@POLYU.EDU.HK
Department of Applied Mathematics
The Hong Kong Polytechnic University
Hong Kong SAR, China

Editor: Ryan Tibshirani

Abstract

We propose a penalized nonparametric approach to estimating the quantile regression process (QRP) in a nonseparable model using rectifier quadratic unit (ReQU) activated deep neural networks and introduce a novel penalty function to enforce non-crossing of quantile regression curves. We establish the non-asymptotic excess risk bounds for the estimated QRP and derive the mean integrated squared error for the estimated QRP under mild smoothness and regularity conditions. To establish these non-asymptotic risk and estimation error bounds, we also develop a new error bound for approximating C^s smooth functions with $s > 1$ and their derivatives using ReQU activated neural networks. This is a new approximation result for ReQU networks and is of independent interest and may be useful in other problems. Our numerical experiments demonstrate that the proposed method is competitive with or outperforms two existing methods, including methods using reproducing kernels and random forests for nonparametric quantile regression.

Keywords: Approximation error, quantile process, deep neural networks, monotonic constraints, non-asymptotic error bound

[*]. Guohao Shen and Yuling Jiao contributed equally to this work.

1. Introduction

Consider a nonparametric regression model

$$Y = f_0(X, U), \tag{1}$$

where $Y \in \mathbb{R}$ is a response variable, $X \in \mathcal{X} \subset \mathbb{R}^d$ is a d-dimensional vector of predictors, U is an unobservable random variable following the uniform distribution on $(0, 1)$ and independent of X. The function $f_0 : \mathcal{X} \times (0, 1) \to \mathbb{R}$ is an unknown regression function, and f_0 is increasing in its second argument. This is a non-separable quantile regression model, in which the specification $U \sim \text{Unif}(0, 1)$ is a normalization but not a restrictive assumption (Chernozhukov et al., 2007; Horowitz and Lee, 2007). Nonseparable quantile regression models are important in empirical economics (see, e.g., Blundell et al. (2017)). Based on (1), it can be seen that for any $\tau \in (0, 1)$, the conditional τ-th quantile $Q_{Y|x}(\tau)$ of Y given $X = x$ is

$$Q_{Y|x}(\tau) = f_0(x, \tau). \tag{2}$$

We refer to $f_0 = \{f_0(x, \tau) : (x, \tau) \in \mathcal{X} \times (0, 1)\}$ as a quantile regression process (QRP). A basic property of QRP is that it is nondecreasing with respect to τ for any given $x \in \mathcal{X}$, often referred to as the non-crossing property. We propose a novel penalized nonparametric method for estimating f_0 on a random discrete grid of quantile levels in $(0, 1)$ simultaneously, with a penalty designed to ensure the non-crossing property.

Quantile regression (Koenker and Bassett, 1978) is an important method for modeling the relationship between a response Y and a predictor X. Different from least squares regression that estimates the conditional mean of Y given X, quantile regression models the conditional quantiles of Y given X, so it fully describes the conditional distribution of Y given X. The non-separable model (1) can be transformed into a familiar quantile regression model with an additive error. For any $\tau \in (0, 1)$, we have $P\{Y - f_0(X, \tau) \le 0\} = \tau$ under (1). If we define $\epsilon = Y - f_0(X, \tau)$, then model (1) becomes

$$Y = g_0(X) + \epsilon, \tag{3}$$

where $g_0(X) = f_0(X, \tau)$ and $P(\epsilon \le 0 \mid X = x) = \tau$ for any $x \in \mathcal{X}$. An attractive feature of the nonseparable model (1) is that it explicitly includes the quantile level as a second argument of f_0, which makes it possible to construct a single objective function for estimating the whole quantile process simultaneously.

A general nonseparable quantile regression model that allows a vector random disturbance U was proposed by Chernozhukov and Hansen (2005). The model (1) in the presence of endogeneity was considered by Chernozhukov et al. (2007), who gave local identification conditions for the quantile regression function f_0 and provided sufficient conditions under which a series estimator is consistent. The convergence rate of the series estimator is unknown. The relationship between the nonseparable quantile regression model (1) and the usual separable quantile regression model was discussed in Horowitz and Lee (2007). A study of nonseparable bivariate quantile regression for nonparametric demand estimation using splines under shape constraints was given in Blundell et al. (2017).

There is a large body of literature on separable linear quantile regression in the fixed-dimension setting (Koenker and Bassett, 1978; Koenker, 2005) and in the high-dimensional settings (Belloni and Chernozhukov, 2011; Wang et al., 2012; Zheng et al., 2015). Nonparametric estimation of separable quantile regressions has also been studied. Examples include the methods using shallow neural networks (White, 1992), smoothing splines (Koenker et al., 1994; He and Shi, 1994; He and Ng, 1999) and reproducing kernels (Takeuchi et al., 2006; Sangnier et al., 2016). Semiparametric quantile regression has also been considered in the literature (Chao et al., 2016; Belloni et al., 2019). A popular semiparametric quantile regression model is

$$Q_{Y|x}(\tau) = Z(x)^\top \beta(\tau). \tag{4}$$

where $Q_{Y|x}(\tau)$ is defined in (2) and $Z(x) \in \mathbb{R}^m$ is usually a series representation of the predictor x. The goal is to estimate the coefficient process $\{\beta(\tau) : \tau \in (0,1)\}$ and derive the asymptotic distribution of the estimators. Such results can be used for conducting statistical inference about $\beta(\tau)$. However, they hinge on the model assumption (4). If this assumption is not satisfied, estimation and inference results based on a misspecified model can be misleading.

Quantile regression curves satisfy a monotonicity condition. At the population level, it holds that $f_0(x, \tau_2) \geq f_0(x; \tau_1)$ for any $0 < \tau_1 < \tau_2 < 1$ and every $x \in \mathcal{X}$. However, for an estimator $\hat{f}$ of f_0, there can be values of x for which the quantile curves cross, that is, $\hat{f}(x, \tau_2) < \hat{f}(x; \tau_1)$ due to finite sample size and sampling variation. Quantile crossing makes it challenging to interpret the estimated quantile curves (He, 1997). Therefore, it is desirable to avoid it in practice. Constrained optimization methods have been used to obtain non-crossing conditional quantile estimates in linear quantile regression and nonparametric quantile regression with a scalar covariate (He, 1997; Bondell et al., 2010). A method proposed by Chernozhukov et al. (2010) uses sorting to rearrange the original estimated non-monotone quantile curves into monotone curves without crossing. It is also possible to apply the isotonization method for qualitative constraints (Mammen, 1991) to the original estimated quantile curves to obtain quantile curves without crossing. Brando et al. (2022) proposed a deep learning algorithm for estimating conditional quantile functions that ensures quantile monotonicity. They first restrict the output of a deep neural network to be positive as the estimator of the derivative of the conditional quantile function, then by using truncated Chebyshev polynomial expansion, the estimated derivative is integrated and the estimator of conditional quantile function is obtained.

Recently, there has been active research on nonparametric least squares regression using deep neural networks (Bauer and Kohler, 2019; Schmidt-Hieber, 2020; Chen et al., 2019; Kohler et al., 2019; Nakada and Imaizumi, 2020; Farrell et al., 2021; Jiao et al., 2023). These studies show that, under appropriate conditions, least squares regression with neural networks can achieve the optimal rate of convergence up to a logarithmic factor for estimating a conditional mean regression function. Since the quantile regression problem considered in this work is quite different from the least squares regression, different treatments are needed in the present setting.

We propose a penalized nonparametric approach for estimating the nonseparable quantile regression model (1) using rectified quadratic unit (ReQU) activated deep neural networks. We introduce a penalty function for the derivative of the QRP with respect to the quantile level to avoid quantile crossing, which does not require numerical integration as in Brando et al. (2022).

Our main contributions are as follows.

1. We propose a novel loss function that is the expected quantile loss function with respect to a distribution over $(0,1)$ for the quantile level, instead of the quantile loss function at a single quantile level as in the usual quantile regression. An appealing feature of the proposed loss function is that it can be used to estimate quantile regression functions at an arbitrary number of quantile levels simultaneously.

2. We propose a new penalty function to enforce the non-crossing property for quantile curves at different quantile levels. This is achieved by encouraging the derivative of the quantile regression function $f(x,\tau)$ with respect to τ to be nonnegative. The use of ReQU activation ensures that the derivative exists. This penalty is easy to implement and computationally feasible for high-dimensional predictors.

3. We establish non-asymptotic excess risk bounds for the estimated QRP and derive the mean integrated squared error for the estimated QRP under the assumption that the underlying quantile regression process belongs to the C^s class of functions on $\mathcal{X} \times (0,1)$.

4. We derive novel approximation error bounds for C^s smooth functions with a positive smoothness index s and their derivatives using ReQU activated deep neural networks. The error bounds hold not only for the target function, but also its derivatives. This is a new approximation result for ReQU networks and is of independent interest and may be useful in other problems.

5. We conduct simulation studies to evaluate the finite sample performance of the proposed QRP estimation method and demonstrate that it is competitive or outperforms two existing nonparametric quantile regression methods, including kernel based quantile regression and quantile regression forests.

The remainder of the paper is organized as follows. In Section 2 we describe the proposed method for nonparametric estimation of QRP with a novel penalty function for avoiding quantile crossing. In Section 3 we state the main results of the paper, including bounds for the non-asymptotic excess risk and the mean integrated squared error for the proposed QRP estimator. In Section 4 we derive the stochastic error for the QRP estimator. In Section 5 we establish a novel approximation error bound for approximating C^s smooth functions and their derivatives using ReQU activated neural networks. Section 6 describes computational implementation of the proposed method. In Section 7 we conduct numerical studies to evaluate the performance of the QRP estimator. Conclusion remarks are given Section 8. Proofs and technical details are provided in the Appendix.

2. Deep quantile regression process estimation with non-crossing constraints

In this section, we describe the proposed approach for estimating a quantile regression process using deep neural networks with a novel penalty for avoiding non-crossing.

2.1 The standard quantile regression

We first recall the standard quantile regression method with the check loss function (Koenker and Bassett, 1978). For a given quantile level $\tau \in (0,1)$, the quantile check loss function is

$$\rho_\tau(x) = x\{\tau - I(x \leq 0)\},\ x \in \mathbb{R}.$$

For any $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ and $\tau \in (0,1)$, the τ-risk of f is defined by

$$\mathcal{R}^\tau(f) = \mathbb{E}_{X,Y}\{\rho_\tau(Y - f(X,\tau))\}. \tag{5}$$

Clearly, by the model assumption in (2), for each given $\tau \in (0,1)$, the function $f_0(\cdot,\tau)$ is the minimizer of $\mathcal{R}^\tau(f)$ over all the measurable functions from $\mathcal{X} \times (0,1) \to \mathbb{R}$, i.e., for

$$f_*^\tau = \arg\min_f \mathcal{R}^\tau(f) = \arg\min_f \mathbb{E}_{X,Y}\{\rho_\tau(Y - f(X,\tau))\},$$

we have $f_*^\tau \equiv f_0(\cdot,\tau)$ on $\mathcal{X} \times \{\tau\}$. This is a basic identification result for the standard quantile regression, where only a single conditional quantile function $f_0(\cdot,\tau)$ at a given quantile level τ is estimated.

2.2 Expected check loss with non-crossing constraints

Our goal is to estimate the whole quantile regression process $\{f_0(\cdot,\tau) : \tau \in (0,1)\}$. The existing method estimates the quantile curves $f_0(\cdot,\tau)$ at each τ separately and then monotonize them afterwards to ensure non-crossing. In this subsection, we present the proposed penalized estimation framework, where the entire quantile process is modelled nonparametrically. We construct a randomized objective function by treating the quantile level τ as a random variable whose distribution is supported on the unit interval. The resulting estimator is naturally non-crossing, computationally efficient, and easy-to-implement.

Let ξ be a random variable supported on $(0,1)$ with density function $\pi_\xi : (0,1) \to \mathbb{R}^+$. Consider the following randomized version of the check loss function

$$\rho_\xi(x) = x\{\xi - I(x \leq 0)\},\ x \in \mathbb{R}.$$

For a measurable function $f : \mathcal{X} \times (0,1) \to \mathbb{R}$, define the ξ-risk of f by

$$\mathcal{R}^\xi(f) = \mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y - f(X,\xi))\} = \int_0^1 \mathcal{R}^t(f)\pi_\xi(t)dt. \tag{6}$$

At the population level, let $f^* : \mathcal{X} \times (0,1) \to \mathbb{R}$ be a measurable function defined by

$$f^* \in \arg\min_f \mathcal{R}^\xi(f) = \arg\min_f \int_0^1 \mathcal{R}^t(f)\pi_\xi(t)dt.$$

Note that f^* may not be unique if (X,ξ) has zero density on some set $A_0 \subseteq \mathcal{X} \times (0,1)$ with positive Lebesgue measure. In this case, $f^*(x,\xi)$ can take any value for $(x,\xi) \in A_0$ since it does not affect the risk. Importantly, since the target quantile function $f_0(\cdot,\tau)$ defined in (1) minimizes the τ-risk $\mathcal{R}^\tau$ for each $\tau \in (0,1)$, f_0 is also the risk minimizer of $\mathcal{R}^\xi$ over all measurable functions. Then we have $f_0 \equiv f^*$ on $\mathcal{X} \times (0,1)$ almost everywhere given that (X, ξ) has nonzero density on $\mathcal{X} \times (0,1)$ almost everywhere.

In addition, the risk $\mathcal{R}^{\xi}$ depends on the distribution of ξ. Different distributions of ξ may lead to different $\mathcal{R}^{\xi}$. However, the target quantile process f_0 is still the risk minimizer of $\mathcal{R}^{\xi}$ over all measurable functions, regardless of the distribution of ξ. We state this property in the following proposition, whose proof is given in the Appendix.

Proposition 1 *For any random variable ξ supported on $(0,1)$, the target function f_0 minimizes the risk $\mathcal{R}^{\xi}(\cdot)$ defined in (6) over all measurable functions, i.e.,*

$$f_0 \in \arg\min_f \mathcal{R}^{\xi}(f) = \arg\min_f \mathbb{E}_{X,Y,\xi}\{\rho_{\xi}(Y - f(X,\xi))\}.$$

Furthermore, if (X, ξ) has non zero density almost everywhere on $\mathcal{X} \times (0,1)$ and the probability measure of (X, ξ) is absolutely continuous with respect to the Lebesgue measure, then f_0 is the unique minimizer of $\mathcal{R}^{\xi}(\cdot)$ over all measurable functions in the sense of almost everywhere(almost surely), i.e.,

$$f_0 = \arg\min_f \mathcal{R}^{\xi}(f) = \arg\min_f \mathbb{E}_{X,Y,\xi}\{\rho_{\xi}(Y - f(X,\xi))\},$$

up to a negligible set with respect to the probability measure of (X, η) on $\mathcal{X} \times (0,1)$.

The loss function in (6) can be viewed as a weighted quantile check loss function, where the distribution of ξ weights the importance of different quantile levels in the estimation. Proposition 1 implies that, though different distributions of ξ may result in different estimators with finite samples, these estimators can be shown to be consistent for the target function f_0 under mild conditions.

A natural and simple choice of the distribution of ξ is the uniform distribution over $(0,1)$ with density function $\pi_{\xi}(t) \equiv 1$ for all $t \in (0,1)$. In this paper we focus on the case that ξ is uniformly distributed on $(0,1)$, but we emphasize that the theoretical results presented in Section 3-5 hold for different choices of the distribution of ξ. More discussions can be found at the beginning of Section 3.

In practice, only a random sample $\{(X_i, Y_i)\}_{i=1}^n$ is available. We can only estimate the quantile process on a discrete grid of quantile levels. Moreover, the integral with respect to π_{ξ} in (6) does not have a closed form expression, so we approximate it using a random sample $\{\xi_i\}_{i=1}^n$ generated from π_{ξ}. Then, the empirical risk corresponding to the population risk $R^{\xi}(f)$ in (6) is

$$\mathcal{R}_n^{\xi}(f) = \frac{1}{n} \sum_{i=1}^{n} \rho_{\xi_i}(Y_i - f(X_i, \xi_i)). \tag{7}$$

By minimizing (7) over certain hypothesis function class, we can obtain estimates for the process $\{f_0(\cdot, \tau) : \tau \in (0,1)\}$ on a grid of random quantile levels that are increasingly dense as the sample size n increases.

Let $\mathcal{F}_n$ be a class of deep neural network (DNN) functions defined on $\mathcal{X} \times (0,1)$. We define the QRP estimator as the empirical risk minimizer

$$\hat{f}_n \in \arg\min_{f \in \mathcal{F}_n} \mathcal{R}_n^{\xi}(f). \tag{8}$$

The estimator $\hat{f}_n$ contains estimates of the quantile curves $\{\hat{f}_n(x,\xi_1),\dots,\hat{f}_n(x,\xi_n)\}$ at the quantile levels $\xi_1,\dots,\xi_n$. An attractive feature of this approach is that it estimates all these quantile curves simultaneously; since the estimator $\hat{f}_n$ based on neural networks takes the quantile level as an input and automatically interpolates the quantile curves at the points other than $\{\xi_1,\dots,\xi_n\}$.

By the basic properties of quantiles, the underlying quantile regression function $f_0(x,\tau)$ satisfies the monotonicity constraints

$$f_0(x,\xi_{(1)}) \le \cdots \le f_0(x;\xi_{(n)}),\ x \in \mathcal{X},$$

where $\xi_{(1)} < \cdots < \xi_{(n)}$ are the ordered values of $\xi_1,\dots,\xi_n$. It is desirable that the estimated quantile function also possess this monotonicity property. However, with finite samples and due to sampling variation, the estimated quantile function $\hat{f}_n(x,\tau)$ may violate this monotonicity property and cross for some values of x, leading to an improper distribution for the predicted response. To avoid quantile crossing, constraints are required in the estimation process. However, it is not a simple matter to impose monotonicity constraints directly on regression quantiles.

We use the fact that a regression quantile function $f_0(x,\tau)$ is nondecreasing in its second argument τ if its partial derivative with respect to τ is non-negative. For a quantile regression function $f:\mathcal{X}\times(0,1)\to\mathbb{R}$ with first order partial derivatives, we let $\partial f/\partial\tau$ denote the partial derivative operator for f with respect to its second argument. A natural way to impose the monotonicity on $f(x,\tau)$ with respect to τ is to constrain its partial derivative with respect to τ to be nonnegative. So it is natural to consider ways to constrain the derivative of $f(x;\tau)$ with respect to τ to be nonnegative.

We propose a penalty function based on the ReLU activation function, $\sigma_1(x)=\max\{x,0\}$ $x\in\mathbb{R}$, as follows,

$$\kappa(f)=\mathbb{E}_{X,\xi}\sigma_1\Big(-\frac{\partial}{\partial\tau}f(X,\xi)\Big)=\mathbb{E}_{X,\xi}\Big[\max\Big\{-\frac{\partial}{\partial\tau}f(X,\xi),0\Big\}\Big]. \tag{9}$$

Clearly, this penalty function encourages $\frac{\partial}{\partial\tau}f(x,\xi)\ge 0$. The empirical version of κ is

$$\kappa_n(f):=\frac{1}{n}\sum_{i=1}^{n}\Big[\max\Big\{-\frac{\partial}{\partial\tau}f(X_i,\xi_i),0\Big\}\Big]. \tag{10}$$

Based on the above discussion and combining (6) and (9), we propose the following population level penalized risk for the regression quantile functions

$$\mathcal{R}^{\xi}_{\lambda}(f)=\mathbb{E}_{X,Y,\xi}\Big[\rho_\xi(Y-f(X,\xi))+\lambda\max\Big\{-\frac{\partial}{\partial\tau}f(X,\xi),0\Big\}\Big], \tag{11}$$

where $\lambda\ge 0$ is a tuning parameter. Suppose that the partial derivative of the target quantile function f_0 with respect to its second argument exists. It then follows that $\frac{\partial}{\partial\tau}f_0(x,u)\ge 0$ for any $(x,u)\in\mathcal{X}\times(0,1)$, and thus f_0 is also the risk minimizer of $\mathcal{R}^{\xi}_{\lambda}(f)$ over all measurable functions on $\mathcal{X}\times(0,1)$.

The empirical risk with respect to (11) for estimating the quantile functions is

$$\mathcal{R}^{\xi}_{n,\lambda}(f)=\frac{1}{n}\sum_{i=1}^{n}\Big[\rho_{\xi_i}(Y_i-f(X_i,\xi_i))+\lambda\max\Big\{-\frac{\partial}{\partial\tau}f(X_i,\xi_i),0\Big\}\Big]. \tag{12}$$

The penalized empirical risk minimizer over a class of functions $\mathcal{F}_n$ is given by

$$\hat{f}_n^\lambda \in \arg\min_{f\in\mathcal{F}_n} \mathcal{R}^\xi_{n,\lambda}(f), \tag{13}$$

We refer to $\hat{f}_n^\lambda$ as a penalized deep quantile regression process (DQRP) estimator. The function class $\mathcal{F}_n$ plays an important role in (13). Next we give a detailed description of $\mathcal{F}_n$.

2.3 ReQU activated neural networks

Neural networks with nonlinear activation functions have proven to be a powerful approach for approximating multi-dimensional functions. Rectified linear unit (ReLU), defined as $\sigma_1(x) = \max\{x, 0\}, x \in \mathbb{R}$, is one of the most commonly used activation functions due to its attractive properties in computation and optimization. ReLU neural networks have received much attention in statistical machine learning (Schmidt-Hieber, 2020; Bauer and Kohler, 2019; Jiao et al., 2023) and applied mathematics (Yarotsky, 2017, 2018; Shen et al., 2020, 2019; Lu et al., 2021a). However, since partial derivatives are involved in our proposed objective function (12) and need to be approximated, piecewise linear ReLU networks may not be an ideal choice as the ReLU function is not differentiable at 0.

We will use the Rectified Quadratic Unit (ReQU) activation, which is smooth and has a continuous first derivative. The ReQU activation function, denoted as σ_2, is the squared ReLU,

$$\sigma_2(x) = \sigma_1^2(x) = [\max\{x, 0\}]^2,\ x \in \mathbb{R}. \tag{14}$$

With ReQU as the activation function, the network will be smooth and differentiable. Thus ReQU activated networks are suitable to approximate a target function and its derivative as in (12). Below in Table 1 we tabulate the comparison between ReLU and ReQU networks in several important aspects.

Table 1: A comparison between ReLU and ReQU activation functions. Both activation functions are continuous and non-saturated, which won't encounter the problem of "vanishing gradients" during the optimization as Sigmodal activations (e.g. Sigmoid, Tanh) do. ReQU activation is differentiable and can approximate the gradient of the target function while ReLU activation is not, especially for estimation involving high-order derivatives of the target function.

Activation	Continuous	Non-saturated	Differentiable	Gradient Estimation
ReLU	✓	✓	✗	✗
ReQU	✓	✓	✓	✓

We set the function class $\mathcal{F}_n$ in (13) to be $\mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$, a class of ReQU activated multilayer perceptrons $f : \mathbb{R}^{d+1} \to \mathbb{R}$ with depth $\mathcal{D}$, width $\mathcal{W}$, size $\mathcal{S}$, number of neurons $\mathcal{U}$ and f satisfying $\|f\|_\infty \le \mathcal{B}$ and $\|\frac{\partial}{\partial\tau} f\|_\infty \le \mathcal{B}'$ for some $1 \le \mathcal{B}, \mathcal{B}' < \infty$, where $\|f\|_\infty$ is the sup-norm of a function f. The network parameters may depend on the sample size n, but the dependence is omitted in the notation for simplicity. The boundedness conditions on the neural network functions and their derivatives can be implemented by truncating or clipping the network output and its derivative (Chen et al., 2020; Lee and Kifer, 2021).

The architecture of a multilayer perceptron can be expressed as a composition of a series of functions

$$f(x) = \mathcal{L}_{\mathcal{D}} \circ \sigma_2 \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma_2 \circ \cdots \circ \sigma_2 \circ \mathcal{L}_1 \circ \sigma_2 \circ \mathcal{L}_0(x),\ x \in \mathbb{R}^{p_0},$$

where $p_0 = d+1$, σ_2 is the rectified quadratic unit (ReQU) activation function defined in (14) (operating on x component-wise if x is a vector), and $\mathcal{L}_i$'s are linear functions

$$\mathcal{L}_i(x) = W_i x + b_i,\ x \in \mathbb{R}^{p_i}, i = 0, 1, \ldots, \mathcal{D},$$

with $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ a weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ a bias vector. Here p_i is the width (the number of neurons or computational units) of the i-th layer. The input data consisting of predictor values X is the first layer and the output is the last layer. Such a network f has $\mathcal{D}$ hidden layers and $(\mathcal{D}+2)$ layers in total. We use a $(\mathcal{D}+2)$-vector $(p_0, p_1, \ldots, p_{\mathcal{D}}, p_{\mathcal{D}+1})^\top$ to describe the width of each layer; particularly, $p_0 = d+1$ is the dimension of the input (X, ξ) and $p_{\mathcal{D}+1} = 1$ is the dimension of the response Y in model (2). The width $\mathcal{W}$ is defined as the maximum width of hidden layers, i.e., $\mathcal{W} = \max\{p_1, ..., p_{\mathcal{D}}\}$; the size $\mathcal{S}$ is defined as the total number of parameters in the network f_ϕ, i.e., $\mathcal{S} = \sum_{i=0}^{\mathcal{D}} \{p_{i+1} \times (p_i + 1)\}$; the number of neurons $\mathcal{U}$ is defined as the number of computational units in hidden layers, i.e., $\mathcal{U} = \sum_{i=1}^{\mathcal{D}} p_i$. Note that the neurons in consecutive layers are connected to each other via linear transformation matrices W_i, $i = 0, 1, \ldots, \mathcal{D}$.

The network parameters can depend on the sample size n, but the dependence is suppressed for notational simplicity, that is, $\mathcal{S} = \mathcal{S}_n$, $\mathcal{U} = \mathcal{U}_n$, $\mathcal{D} = \mathcal{D}_n$, $\mathcal{W} = \mathcal{W}_n$, $\mathcal{B} = \mathcal{B}_n$ and $\mathcal{B}' = \mathcal{B}'_n$. This makes it possible to approximate the target regression function by neural networks as n increases. The approximation and excess error rates will be determined in part by how these network parameters depend on n.

3. Main results

In this section, we state our main results on the bounds for the excess risk and estimation error of the penalized DQRP estimator. The excess risk of the penalized DQRP estimator is defined as

$$\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0) = \mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y - \hat{f}_n^\lambda(X, \xi)) - \rho_\xi(Y - f_0(X, \xi))\},$$

where (X, Y, ξ) is an independent copy of the random sample $\{(X_i, Y_i, \xi_i)\}_{i=1}^n$.

Since the definition of $\mathcal{R}^\xi(\cdot)$ depends on the distribution of ξ, the theoretical guarantees for $\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0)$ established in this section also depends on the distribution of ξ. For instance, if ξ is chosen to have a point mass distribution at a fixed $\tau \in (0, 1)$, i.e., $P(\xi = \tau) = 1$, then the excess risk

$$\begin{aligned}
\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0) &= \mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y - \hat{f}_n^\lambda(X, \xi))\} - \mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y - f_0(X, \xi))\} \\
&= \mathbb{E}_{X,Y}\{\rho_\tau(Y - \hat{f}_n^\lambda(X, \tau))\} - \mathbb{E}_{X,Y}\{\rho_\tau(Y - f_0(X, \tau))\} \\
&= \mathcal{R}^\tau(\hat{f}_n^\lambda) - \mathcal{R}^\tau(f_0),
\end{aligned}$$

which is precisely the excess risk for nonparametric quantile regression at a single quantile level τ. In this case, our theoretical guarantee for the estimated quantile process will be only in terms of the estimated quantile curve $\hat{f}_n^\lambda(\cdot,\tau)$ at quantile level τ but not any other quantile levels in $(0,1)\backslash\{\tau\}$.

If ξ is chosen to follow a discrete uniform distribution on a set $\{\tau_i\}_{i=1}^k$ with $0<\tau_1<\ldots<\tau_k<1$, then the excess risk

$$\begin{aligned}\mathcal{R}^\xi(\hat{f}_n^\lambda)-\mathcal{R}^\xi(f_0)&=\mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y-\hat{f}_n^\lambda(X,\xi))\}-\mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y-f_0(X,\xi))\}\\&=\frac{1}{k}\sum_{j=1}^k\Big[\mathbb{E}_{X,Y}\{\rho_{\tau_j}(Y-\hat{f}_n^\lambda(X,\tau_j))\}-\mathbb{E}_{X,Y}\{\rho_{\tau_j}(Y-f_0(X,\tau_j))\}\Big]\\&=\frac{1}{k}\sum_{j=1}^k\Big[\mathcal{R}^{\tau_j}(\hat{f}_n^\lambda)-\mathcal{R}^{\tau_j}(f_0)\Big],\end{aligned}$$

which reduces to the excess risk for nonparametric quantile regression at multiple quantile levels. In this case, our theoretical results will only guarantee the consistency of the estimated quantile process at quantile levels in $\{\tau_i\}_{i=1}^k$. In addition, if the distribution of ξ is chosen to concentrate more around the extreme quantiles close to 0 or 1, our theoretical results can lead to a tighter bound for the prediction errors at extreme quantiles; see Corollary 5 for details.

Next, we first state a basic lemma for bounding the excess risk.

Lemma 1 (Excess risk decomposition) *For the penalized empirical risk minimizer $\hat{f}_n^\lambda$ defined in (13), its excess risk can be upper bounded as follows:*

$$\begin{aligned}&\mathcal{R}^\xi(\hat{f}_n^\lambda)-\mathcal{R}^\xi(f_0)\le\mathcal{R}_\lambda^\xi(\hat{f}_n^\lambda)-\mathcal{R}_\lambda^\xi(f_0)\\&\qquad\le 2\sup_{f\in\mathcal{F}_n}\big|[\mathcal{R}_\lambda^\xi(f)-\mathcal{R}_\lambda^\xi(f_0)]-[\mathcal{R}_{n,\lambda}^\xi(f)-\mathcal{R}_{n,\lambda}^\xi(f_0)]\big|+\inf_{f\in\mathcal{F}_n}\Big[\mathcal{R}_\lambda^\xi(f)-\mathcal{R}_\lambda^\xi(f_0)\Big].\end{aligned}$$

Therefore, the bound for ξ-excess risk can be decomposed into two parts: the stochastic error $\sup_{f\in\mathcal{F}_n}|[\mathcal{R}_\lambda^\xi(f)-\mathcal{R}_\lambda^\xi(f_0)]-[\mathcal{R}_{n,\lambda}^\xi(f)-\mathcal{R}_{n,\lambda}^\xi(f_0)]|$ and the approximation error $\inf_{f\in\mathcal{F}_n}[\mathcal{R}_\lambda^\xi(f)-\mathcal{R}_\lambda^\xi(f_0)]$. Once bounds for the stochastic error and approximation error are available, we can immediately obtain an upper bound for the ξ-excess risk of the penalized DQRP estimator $\hat{f}_n^\lambda$.

3.1 Non-asymptotic excess risk bounds

We first state the conditions needed for establishing the excess risk bounds.

Definition 2 (Multivariate differentiability classes C^s) *A function $f:\mathbb{B}\subset\mathbb{R}^d\to\mathbb{R}$ defined on a subset $\mathbb{B}$ of $\mathbb{R}^d$ is said to be in class $C^s(\mathbb{B})$ on $\mathbb{B}$ for a positive integer s, if all partial derivatives*

$$D^\alpha f:=\frac{\partial^\alpha}{\partial x_1^{\alpha_1}\partial x_2^{\alpha_2}\cdots\partial x_d^{\alpha_d}}f$$

exist and are continuous on $\mathbb{B}$ for all non-negative integers $\alpha_1, \alpha_2, \dots, \alpha_d$ such that $\alpha := \alpha_1 + \alpha_2 + \cdots + \alpha_d \leq s$. In addition, we define the norm of f over $\mathbb{B}$ by

$$\|f\|_{C^s} := \sum_{|\alpha|_1 \leq s} \sup_{\mathbb{B}} |D^\alpha f|,$$

where $|\alpha|_1 := \sum_{i=1}^d \alpha_i$ for any vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_d) \in \mathbb{R}^d$.

We make the following smoothness assumption on the target regression quantile function f_0.

Assumption 3 *The target quantile regression function $f_0 : \mathcal{X} \times (0,1) \to \mathbb{R}$ defined in (2) belongs to $C^s(\mathcal{X} \times (0,1))$ for $s \in \mathbb{N}^+$ with $s > 1$, where $\mathbb{N}^+$ is the set of positive integers.*

Let $\mathcal{F}_n' := \{\frac{\partial}{\partial \tau} f : f \in \mathcal{F}_n\}$ denote the function class induced by $\mathcal{F}_n$. For a class $\mathcal{F}$ of functions: $\mathcal{X} \to \mathbb{R}$, its pseudo dimension, denoted by $\mathrm{Pdim}(\mathcal{F})$, is the largest integer m for which there exists $(x_1, \dots, x_m, y_1, \dots, y_m) \in \mathcal{X}^m \times \mathbb{R}^m$ such that for any $(b_1, \dots, b_m) \in \{0,1\}^m$ there exists $f \in \mathcal{F}$ such that $\forall i : f(x_i) > y_i \iff b_i = 1$ (Anthony and Bartlett, 1999; Bartlett et al., 2019).

Theorem 4 (Non-asymptotic ξ-excess risk bounds) *For any $N \in \mathbb{N}^+$, let $\mathcal{F}_n := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be the ReQU activated neural networks $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ with depth $\mathcal{D} \leq 2N - 1$, width $\mathcal{W} \leq 12N^d$, the number of neurons $\mathcal{U} \leq 15N^{d+1}$, the number of parameters $\mathcal{S} \leq 24N^{d+1}$. Under Assumption 3, suppose that $\mathcal{B} \geq \|f_0\|_{C^0}$ and $\mathcal{B}' \geq \|f_0\|_{C^1}$. Then for $n \geq \max\{\mathrm{Pdim}(\mathcal{F}_n), \mathrm{Pdim}(\mathcal{F}_n')\}$ and any $\lambda \geq 0$, the ξ-excess risk of the penalized DQRP estimator $\hat{f}_n^\lambda$ defined in (13) satisfies*

$$\mathbb{E}\{\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0)\} \leq C_0(\mathcal{B} + \lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}} N^{(d+3)/2} + C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s} N^{-(s-1)}, \tag{15}$$

where $C_0 > 0$ is a universal constant and $C_{s,d,\mathcal{X}}$ is a positive constant depending only on d, s and the diameter of the support $\mathcal{X} \times (0,1)$. If $\lambda > 0$, we also have

$$\mathbb{E}\max\left\{-\frac{\partial}{\partial \tau}\hat{f}_n^\lambda(X,\xi), 0\right\} \leq \frac{1}{\lambda}\left[C_0(\mathcal{B} + \lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}} N^{(d+3)/2} + C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s} N^{-(s-1)}\right]. \tag{16}$$

Further, if we set $\lambda = \log n$ and $N = \lfloor n^{1/(d+2s+1)} \rfloor$, then (15) and (16) imply the upper bounds

$$\mathbb{E}\{\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0)\} \leq C(\log n)^2 n^{-\frac{s-1}{d+2s+1}}, \quad \mathbb{E}\max\left\{-\frac{\partial}{\partial \tau}\hat{f}_n^\lambda(X,\xi), 0\right\} \leq C(\log n)^2 n^{-\frac{s-1}{d+2s+1}},$$

where $C > 0$ is a constant depending only on $\mathcal{B}, \mathcal{B}', s, d, \mathcal{X}$ and $\|f_0\|_{C^s}$.

The upper bound for the ξ-excess risk (15) holds for any $\lambda \geq 0$, and the upper bound for the penalty term (16) holds for $\lambda > 0$. Note that larger λ leads to larger upper bound in

(15) but smaller upper bound in (16). For each fixed sample size n, one can choose a proper positive integer N based on n to construct such a ReQU network to achieve fast convergence rate of the ξ-excess risk with respect to the sample size n. We recommend $\lambda = \log n$ and $N = \lfloor n^{1/(d+2s+1)} \rfloor$ in (15) and (16), and we can obtain upper bounds with convergence rate $(\log n)^2 n^{-(s-1)/(d+1+2s)}$ for the ξ-excess risk. The term $(s-1)$ in the exponent is due to the approximation of the first-order partial derivative of the target function. Of course, the smoothness of the target function f_0 is unknown in practice and how to determine the smoothness of an unknown function is a difficult problem.

The non-asymptotic upper bound in Theorem 4 is for the excess risk for the entire estimated quantile regression process. To the best of our knowledge, most existing studies on nonparametric quantile regression focus on the quantile curve estimation at a specific quantile level, and their convergence results are for a given quantile level. In contrast, our method and theoretical results concern the entire quantile process and its derivative with respect to the quantile level.

We can derive a pointwise upper bound for the excess risk at any specific quantile level of interest. To this end, for any specific $\tau \in (0,1)$ and $\delta < 1$, we denote the collection of neighborhoods of τ with radium δ by $\mathbb{B}(\tau, \delta) := \{[a,b] : b - a = \delta, \tau \in [a,b]\}$, and define

$$B_\xi^\tau(\delta) := \underset{B \in \mathbb{B}(\tau,\delta)}{\operatorname{argmax}} \int_B \pi_\xi(t) dt$$

as the δ-neighborhood of τ on which ξ has the largest probability. Note that $\mathbb{P}(\xi \in B_\xi^\tau) = \int_{B_\xi^\tau} \pi_\xi(t) dt$.

Corollary 5 (Pointwise excess risk bounds) *Suppose ξ is a random variable with non-zero density on $(0,1)$ almost everywhere. For any $N \in \mathbb{N}^+$, let $\mathcal{F}_n := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be the ReQU activated neural networks $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ with depth $\mathcal{D} \le 2N - 1$, width $\mathcal{W} \le 12N^d$, the number of neurons $\mathcal{U} \le 15N^{d+1}$, the number of parameters $\mathcal{S} \le 24N^{d+1}$. Under Assumption 3, suppose that $\mathcal{B} \ge \|f_0\|_{C^0}$ and $\mathcal{B}' \ge \|f_0\|_{C^1}$. Then, for $n \ge \max\{\mathrm{Pdim}(\mathcal{F}_n), \mathrm{Pdim}(\mathcal{F}_n')\}$, the pointwise excess risk of the penalized DQRP estimator $\hat{f}_n^\lambda$ at a specific quantile level $\tau \in (0,1)$ satisfies*

$$\begin{aligned}
&\mathbb{E}_{X,Y}\{\rho_\tau(Y - \hat{f}_n^\lambda(X,\tau)) - \rho_\tau(Y - f_0(X,\tau))\} \\
=&\mathbb{E}\{\mathcal{R}^\tau(\hat{f}_n^\lambda) - \mathcal{R}^\tau(f_0)\} \le \frac{1}{\mathbb{P}(\xi \in B_\xi^\tau(\delta))} \Big[\mathbb{E}\{\mathcal{R}(\hat{f}_n^\lambda) - \mathcal{R}(f_0)\}\Big] + 2\delta(\mathcal{B}' + 2\mathcal{B}) \\
\le&\frac{1}{\mathbb{P}(\xi \in B_\xi^\tau(\delta))} \left[C_0(\mathcal{B} + \lambda\mathcal{B}')\sqrt{\frac{d \log n}{n}} N^{(d+3)/2} + C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s} N^{-(s-1)}\right] \\
&\qquad + 2\delta(\mathcal{B}' + 2\mathcal{B}),
\end{aligned}$$

for $\delta \in [0,1)$, where $C_0 > 0$ is a universal constant and $C_{s,d,\mathcal{X}}$ is a positive constant depending only on d, s and the diameter of the support $\mathcal{X} \times (0,1)$. Especially, if ξ is uniformly *distributed on* $(0,1)$*, then*

$$\sup_{\tau\in(0,1)} \mathbb{E}_{X,Y}\{\rho_\tau(Y-\hat{f}_n^\lambda(X,\tau))-\rho_\tau(Y-f_0(X,\tau))\}$$
$$\leq \frac{1}{\delta}\left[C_0(\mathcal{B}+\lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}}N^{(d+3)/2}+C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s}N^{-(s-1)}\right]+2\delta(\mathcal{B}'+2\mathcal{B}). \quad (17)$$

By Corollary 5, for each fixed sample size n, letting $N=\lfloor n^{1/(d+2s+1)}\rfloor$, $\lambda=\log n$, $\delta=n^{-(s-1)/[2(d+2s+1)]}$, and if ξ is uniformly distributed on $(0,1)$, we can obtain an upper bound from (17):

$$\sup_{\tau\in(0,1)} \mathbb{E}\{\mathcal{R}^\tau(\hat{f}_n^\lambda)-\mathcal{R}^\tau(f_0)\}\leq C(\log n)^2 n^{-\frac{s-1}{2(d+2s+1)}},$$

where $C>0$ is a constant depending only on $\mathcal{B},\mathcal{B}',s,d,\mathcal{X}$ and $\|f_0\|_{C^s}$.

In addition, if the distribution of ξ is chosen to concentrate more around the extreme quantiles near 0 and 1, then for any given $\delta\in[0,1)$, the probability $\mathbb{P}(\xi\in B_\xi^\tau(\delta))$ will be larger for τ near 0 and 1. This will lead to a tighter bound for the prediction errors at extreme quantiles, i.e. $\mathbb{E}_{X,Y}\{\rho_\tau(Y-\hat{f}_n^\lambda(X,\tau))-\rho_\tau(Y-f_0(X,\tau))\}$ for τ near 0 and 1.

3.2 Non-asymptotic mean integrated error

The empirical risk minimization quantile estimator typically results in an estimator $\hat{f}_n^\lambda$ whose risk $\mathcal{R}(\hat{f}_n^\lambda)$ is close to the optimal risk $\mathcal{R}(f_0)$ in expectation or with high probability. However, small excess risk in general only implies in a weak sense that the penalized empirical risk minimizer $\hat{f}_n^\lambda$ is close to the target f_0 (Remark 3.18 Steinwart (2007)). We bridge the gap between the excess risk and the mean integrated error of the estimated quantile function. To this end, we need the following condition on the conditional distribution of Y given X.

Assumption 6 *There exist constants* $K>0$ *and* $k>0$ *such that for any* $|\delta|\leq K$*,*

$$|P_{Y|X}(f_0(x,\tau)+\delta\mid x)-P_{Y|X}(f_0(x,\tau)\mid x)|\geq k|\delta|,$$

for all $\tau\in(0,1)$ *and* $x\in\mathcal{X}$ *up to a negligible set, where* $P_{Y|X}(\cdot\mid x)$ *denotes the conditional distribution function of* Y *given* $X=x$*.*

Assumption 6 is a mild condition on the distribution of Y in the sense that, if Y has a density that is bounded away from zero on any compact interval, then Assumption 6 will hold. In particular, no moment assumptions are made on the distribution of Y. Similar conditions are assumed by Padilla and Chatterjee (2021) in studying nonparametric quantile trend filtering for a single quantile level $\tau\in(0,1)$. This condition is weaker than Condition 2 in He and Shi (1994) where the density function of response is required to be lower bounded every where by some positive constant. Assumption 6 is also weaker than Condition D.1 in Belloni and Chernozhukov (2011), which requires the conditional density of Y given $X=x$ to be continuously differentiable and bounded away from zero uniformly for all quantiles in $(0,1)$ and all x in the support $\mathcal{X}$.

Under Assumption 6, the following self-calibration condition can be established as stated below. This will lead to a bound on the mean integrated error of the estimated quantile process based on a bound for the excess risk.

Lemma 7 (Self-calibration) *Suppose that Assumption 6 holds. For any $f : \mathcal{X} \times (0,1) \to \mathbb{R}$, denote*

$$\Delta^2(f, f_0) = \mathbb{E}[\min\{|f(X,\xi) - f_0(X,\xi)|, |f(X,\xi) - f_0(X,\xi)|^2\}],$$

where X is the predictor vector and ξ is a uniform random variable on (0,1) independent of X. Then we have

$$\Delta^2(f, f_0) \le c_{K,k}\{\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f_0)\},$$

for any $f : \mathcal{X} \times (0,1) \to \mathbb{R}$, where $c_{K,k} = \max\{2/k, 4/(Kk)\}$ and $K, k > 0$ are defined in Assumption 6. Especially, if $\|f - f_0\|_\infty \le K$, then

$$\mathbb{E}|f(X,\xi) - f_0(X,\xi)|^2 \le \frac{2}{k}\{\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f_0)\}.$$

Additionally, if f, f_0 are both bounded by $\mathcal{B}$ and $K \ge 2\mathcal{B}$, then $\|f - f_0\|_\infty \le K$.

Theorem 8 (Mean integrated error bound) *Suppose Assumptions 3 and 6 hold. For any $N \in \mathbb{N}^+$, let $\mathcal{F}_n := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be the class of ReQU activated neural networks $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ with depth $\mathcal{D} \le 2N - 1$, width $\mathcal{W} \le 12N^d$, number of neurons $\mathcal{U} \le 15N^{d+1}$, number of parameters $\mathcal{S} \le 24N^{d+1}$ and satisfying $\mathcal{B} \ge \|f_0\|_{C^0}$ and $\mathcal{B}' \ge \|f_0\|_{C^1}$. Then for $n \ge \max\{\mathrm{Pdim}(\mathcal{F}_n), \mathrm{Pdim}(\mathcal{F}'_n)\}$, the mean integrated error of the penalized DQRP estimator $\hat{f}^\lambda_n$ defined in (13) satisfies*

$$\mathbb{E}\{\Delta^2(\hat{f}^\lambda_n, f_0)\} \le c_{K,k}\Big[C_0(\mathcal{B} + \lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}}N^{(d+3)/2} + C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s}N^{-(s-1)}\Big], \quad (18)$$

where $C_0 > 0$ is a universal constant, $c_{K,k}$ is defined in Lemma 7, $C_{s,d,\mathcal{X}}$ is a positive constant depending only on d, s and the diameter of the support $\mathcal{X} \times (0,1)$. Letting $N = \lfloor n^{1/\{(d+2s+1)\}}\rfloor$ and $\lambda = \log n$ in (18), we obtain an upper bound

$$\mathbb{E}\{\Delta^2(\hat{f}^\lambda_n, f_0)\} \le C_1(\log n)^2 n^{-\frac{s-1}{d+2s+1}},$$

where $C_1 > 0$ is a constant depending only on $\mathcal{B}, \mathcal{B}', s, d, K, k, \mathcal{X}$ and $\|f_0\|_{C^s}$. Additionally, if $2\mathcal{B} \le K$, the penalized DQRP estimator $\hat{f}^\lambda_n$ defined in (13) satisfies

$$\mathbb{E}|\hat{f}^\lambda_n(X,\xi) - f_0(X,\xi)|^2 \le \frac{2}{k}\Big[C_0(\mathcal{B} + \lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}}N^{(d+3)/2} + C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s}N^{-(s-1)}\Big]. \quad (19)$$

Letting $N = \lfloor n^{1/\{(d+2s+1)\}}\rfloor$ and $\lambda = \log n$ in (19), we obtain

$$\mathbb{E}|\hat{f}^\lambda_n(X,\xi) - f_0(X,\xi)|^2 \le C_2(\log n)^2 n^{-\frac{s-1}{d+2s+1}},$$

where $C_2 > 0$ is a constant depending only on $\mathcal{B}, \mathcal{B}', s, d, K, k, \mathcal{X}$ and $\|f_0\|_{C^s}$.

For the nonseparable quantile regression model $Y = f_0(X,U)$, where U is uniformly distributed on $(0,1)$ and $f_0 : \mathcal{X} \times (0,1) \to \mathbb{R}$ is a function with a $(d+1)$-dimensional input and increasing in its second argument U, $f_0(\cdot,\cdot)$ is actually the function describing the quantile process of Y. Given any weighting random variable ξ on $(0,1)$, according to Theorem 8 in the revised manuscript, the risk $\mathbb{E}_{X,\xi}|f(X,\xi)-f_0(X,\xi)|^2 := \|f-f_0\|^2_{L^2(\nu(X,\xi))}$, where $\nu(X,\xi)$ denotes the probability measure of (X,ξ). Thus, it can be viewed as the $L_2(\nu(X,\xi))$ distance between f and the target quantile process functions f_0.

Without the crossing penalty in the objective function, the estimation for the derivative function is not needed, thus the convergence rate can be improved. In this case, ReLU activated or other neural networks can be used to estimate the quantile regression process. For instance, Shen et al. (2021) showed that nonparametric quantile regression based on ReLU neural networks attains a convergence rate of $n^{-s/(d+s)}$ up to a logarithmic factor. This rate is slightly faster than the rate $n^{-(s-1)/(d+2s+1)}$ in Theorem 8 when the estimation of the derivative function is involved.

Corollary 9 (Pointwise mean integrated error bound) *Suppose the conditions in Theorem 8 hold, then the penalized DQRP estimator $\hat{f}_n^\lambda$ defined in (13) satisfies*

$$\begin{aligned}&\mathbb{E}[\min\{|\hat{f}_n^\lambda(X,\tau)-f_0(X,\tau)|,|\hat{f}_n^\lambda(X,\tau)-f_0(X,\tau)|^2\}]\\ \leq&\frac{C_{K,k}}{\mathbb{P}(\xi\in B_\xi^\tau(\delta))}\Big[C_0(\mathcal{B}+\lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}}N^{(d+3)/2}+C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s}N^{-(s-1)}\Big]+2C_{K,k}\delta(\mathcal{B}'+2\mathcal{B}),\end{aligned} \tag{20}$$

for a specific $\tau\in(0,1)$ and $\delta\in[0,1)$, where $C_0>0$ is a universal constant, $c_{K,k}$ is defined in Lemma 7 and $C_{s,d,\mathcal{X}}$ is a positive constant depending only on d,s and the diameter of the support $\mathcal{X}\times(0,1)$. If additionally $2\mathcal{B}\leq K$, for $\tau\in(0,1)$ and $\delta\in[0,1)$ we have

$$\begin{aligned}&\mathbb{E}|\hat{f}_n^\lambda(X,\tau)-f_0(X,\tau)|^2\\ \leq&\frac{2}{k\cdot\mathbb{P}(\xi\in B_\xi^\tau(\delta))}\Big[C_0(\mathcal{B}+\lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}}N^{(d+3)/2}+C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s}N^{-(s-1)}\Big]+\frac{4\delta}{k}(\mathcal{B}'+2\mathcal{B}).\end{aligned} \tag{21}$$

Especially, if ξ is uniformly distributed on $(0,1)$, we have

$$\begin{aligned}&\sup_{\tau\in(0,1)}\mathbb{E}|\hat{f}_n^\lambda(X,\tau)-f_0(X,\tau)|^2\\ &\quad\leq\frac{2}{k\delta}\Big[C_0(\mathcal{B}+\lambda\mathcal{B}')\sqrt{\frac{d\log n}{n}}N^{(d+3)/2}+C_{s,d,\mathcal{X}}(1+\lambda)\|f_0\|_{C^s}N^{-(s-1)}\Big]+\frac{4\delta}{k}(\mathcal{B}'+2\mathcal{B}).\end{aligned} \tag{22}$$

Letting $N=\lfloor n^{1/\{(d+2s+1)\}}\rfloor$, $\lambda=\log n$ and $\delta=n^{-(s-1)/\{2(d+2s+1)\}}$ in (22), we obtain

$$\sup_{\tau\in(0,1)}\mathbb{E}|\hat{f}_n^\lambda(X,\tau)-f_0(X,\tau)|^2\leq C_2(\log n)^2 n^{-\frac{s-1}{2(d+2s+1)}},$$

where $C_2>0$ is a constant depending only on $\mathcal{B},\mathcal{B}',s,d,K,k,\mathcal{X}$ and $\|f_0\|_{C^s}$.

3.3 Lower bounds

In this subsection, we derive lower bounds for the prediction errors of the quantile process estimate and the pointwise conditional quantile function in terms of different measurements. We show that our DQRP quantile process estimator can achieve the lower bound of the convergence for process estimation up to logarithmic factors, but fails to achieve the minimax optimal rate for pointwise quantile estimation.

Our proof is based on the application of Fano's inequality (Scarlett and Cevher, 2019) and Varshamov-Gilber lemma (Tsybakov, 2008). By constructing a finite subset of the target function space, we turn the lower bound problem to a multiple hypothesis testing problem, and apply the Fano's inequality to obtain the lower bound.

Theorem 10 (Lower Bound for Quantile Process Estimation) *Given $s \in \mathbb{N}^+$, let $C^s_+ := C^s_+(\mathcal{X} \times (0,1))$ denote the class of functions $f : \mathcal{X} \times (0,1) \to \mathbb{R}$, which is s times differentiable and increasing in its last argument. Under Assumption 6, we have*

$$\inf_{\hat{f}_n} \sup_{f_0 \in C^s_+} \mathbb{E}\|\hat{f}_n - f_0\|_{C^1} \geq C \times n^{-(s-1)/(d+1+2s)},$$

where $C > 0$ is a universal constant and S_n denotes a sample generated from the model (1) with $f_0 \in C^s_+$, and $\hat{f}_n$ is any estimator based on S_n.

The lower bound in Theorem 10 is obtained with respect to the $\|\cdot\|_{C^1}$ distance and under the non-separable quantile regression model $Y = f_0(X, U)$ where f_0 is a $(d+1)$-dimensional function with smoothness index s.

There are several implications of the lower bound results in Theorem 10. First, our upper bound for the expected penalized excess risk $\mathbb{E}[\mathcal{R}^\lambda_\xi(\hat{f}^\lambda_n) - \mathcal{R}^\lambda_\xi(f_0)]$ in Theorem 4 achieves the rate of the lower bound in Theorem 10. And the upper bound in Theorem 4 would match its lower bound as long as $\mathbb{E}[\mathcal{R}^\lambda_\xi(\hat{f}^\lambda_n) - \mathcal{R}^\lambda_\xi(f_0)] \geq c_1 \mathbb{E}\|\hat{f}^\lambda_n - f_0\|_{C^1}$ for some universal constant $c_1 > 0$. Apparently, the opposite of the inequality holds by the Lipschitz property of the penalized risk. While the existence of the constant c_1 relies on proper distributional assumptions on the data and model (e.g., Assumption 6 and Lemma 7) and a relative small error for derivative estimation, i.e., $\|\frac{\partial}{\partial\tau}\hat{f}^\lambda_n - \frac{\partial}{\partial\tau}f_0\|_{C^0} \leq c_2\|\hat{f}^\lambda_n - f_0\|_{C^0}$ for some universal constant $c_2 > 0$ or $\|\frac{\partial}{\partial\tau}\hat{f}^\lambda_n - \frac{\partial}{\partial\tau}f_0\|_{C^0} \leq c_3\lambda \max\{-\frac{\partial}{\partial\tau}\hat{f}^\lambda_n, 0\}$ for some universal constant $c_3 > 0$. Second, regarding the excess risk $\mathcal{R}^\xi(\hat{f}^\lambda_n) - \mathcal{R}^\xi(f_0)$ without the penalty term, distributional assumptions on the data and model (e.g., Assumption 6) can link the excess risk with the prediction error through $\mathcal{R}^\xi(\hat{f}^\lambda_n) - \mathcal{R}^\xi(f_0) \geq c_4\Delta^2(\hat{f}^\lambda_n, f_0)$, where $c_4 > 0$ is a universal constant and $\Delta(\cdot,\cdot)$ is a proper measurement (see Lemma 7). In this sense, our obtained upper bound for the excess risk in Theorem 4 loses a bit of efficiency in the exponent of the rate from the optimal s to $(s-1)$, due to the penalty term on the first-order derivative. Third, our obtained results in Theorem 4 differ from existing ones in several aspects. Many existing deep quantile regression methods focus on the point estimation of the conditional quantile curve at a given quantile level, while our proposed estimator is for estimating the entire quantile process. Our estimation involves derivatives to encourage monotonicity, which results in a slower convergence rate from s to $(s-1)$ in the exponent, while existing quantile regression estimation does not involve derivatives and can achieve a better rate in the exponent with respect to the smoothness index s, but facing a higher risk of quantile-crossing.

The convergence rate in Theorem 10 attains the minimax rate $n^{-(s-k)/(d+2s+1)}$ derived in Stone (1982) for the estimation of the kth derivative of the nonparametric regression function $\mathbb{E}\{Y \mid X\}$ under certain distributional assumptions. For least square estimation for the nonparametric regression model $Y = f_0(X) + \epsilon$ with proper ϵ, the linkage between excess risk and prediction error naturally holds: $\mathcal{R}(\hat{f}_n^\lambda) - \mathcal{R}(f_0) = \mathbb{E}_X\|\hat{f}_n^\lambda(X) - f_0(X)\|^2$, where $\mathcal{R}(f) = \mathbb{E}_{X,Y}\|Y - f(X)\|^2$ is the least square risk. According to the nonparametric regression framework in Stone (1982), several quantile regression methods have been shown to achieve the minimax rate under proper assumptions (Chaudhuri, 1991; He and Shi, 1994; Padilla et al., 2022). For example, the deep conditional quantile estimation at quantile level $\tau = 0.5$ in Padilla et al. (2022) can achieve the minimax rate in Schmidt-Hieber (2020) for nonparametric mean regression under the assumption that the errors are Gaussian and the covariates are uniformly distributed in $[0, 1]^d$.

By Fano's inequality, we also derive lower bounds for the pointwise prediction error of quantile process estimation.

Theorem 11 (Lower Bound for Pointwise Quantile Estimation) *Given $s \in \mathbb{N}^+$, let $C_+^s := C_+^s(\mathcal{X} \times (0,1))$ denote the class of functions $f : \mathcal{X} \times (0,1) \to \mathbb{R}$, which is s times differentiable and increasing in its last argument. Under Assumption 6, we have*

$$\inf_{\hat{f}_n} \sup_{f_0 \in C_+^s} \sup_{\tau \in (0,1)} \mathbb{E}|\hat{f}_n(X,\tau) - f_0(X,\tau)| \geq C \times n^{-s/(d+2s)},$$

where $C > 0$ is a universal constant and S_n denotes a sample generated from the model (1) with $f_0 \in C_+^s$, and $\hat{f}_n$ is any estimator based on S_n.

The lower bound in Theorem 11 is for pointwise prediction error of quantile regression, where the target quantile function is d-dimensional with smoothness index s. This rate attains the minimax rate $n^{-s/(d+2s)}$ in Stone (1982) for the d-dimensional nonparametric regression function with smoothness s. The upper bounds for the pointwise quantile estimation error of our DQRP estimator in Corollary 5 and Corollary 9 fall short of the lower bound in Theorem 11 for several reasons. First, our DQRP estimation incorporates derivatives to enforce monotonicity, which compromises the efficiency of the convergence rate on the exponent from s to $(s-1)$. Second, the target function in our work is a $(d+1)$-dimensional quantile process, which compromises the efficiency of the convergence rate on the exponent from d to $(d+1)$. Third, our proposed objective function in (11) depends on the distribution of the randomized quantile level ξ, but no adversarial training is involved. Thus, the pointwise results of DQRP do not attain the optimal convergence rate in terms of the supremum norm over $\tau \in (0,1)$.

4. Stochastic error

Now we derive non-asymptotic upper bound for the stochastic error given in Lemma 1. The main difficulty here is that the term

$$\sup_{f \in \mathcal{F}_n} \left| [\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)] - [\mathcal{R}_{n,\lambda}^\xi(f) - \mathcal{R}_{n,\lambda}^\xi(f_0)] \right|$$

involves the partial derivatives of the neural network functions in $\mathcal{F}_n$. Thus we also need to study the properties, especially, the complexity of the partial derivatives of the neural network functions in $\mathcal{F}_n$. Let

$$\mathcal{F}_n' := \Big\{ \frac{\partial}{\partial \tau} f(x,\tau) : f \in \mathcal{F}_n, (x,\tau) \in \mathcal{X} \times (0,1) \Big\}.$$

Note that the partial derivative operator is not a Lipschitz contraction operator, thus Talagrand's lemma (Ledoux and Talagrand, 1991) cannot be used to link the Rademacher complexity of $\mathcal{F}_n$ and $\mathcal{F}_n'$, and to obtain an upper bound of the Rademacher complexity of $\mathcal{F}_n'$. In view of this, we consider a new class of neural network functions whose complexity is convenient to compute. Then the complexity of $\mathcal{F}_n'$ can be upper bounded by the complexity of such a class of neural network functions.

The following lemma shows that $\mathcal{F}_n'$ is contained in the class of neural network functions with ReLU and ReQU mixed-activated multilayer perceptrons. In the following, we refer to the neural networks activated by the ReLU or the ReQU as ReLU-ReQU activated neural networks, i.e., the activation functions in each layer of ReLU-ReQU network can be ReLU or ReQU and the activation functions in different layers can be different.

Lemma 12 (Network for partial derivative) *Let $\mathcal{F}_n := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be a class of ReQU activated neural networks $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ with depth (number of hidden layer) $\mathcal{D}$, width (maximum width of hidden layer) $\mathcal{W}$, number of neurons $\mathcal{U}$, number of parameters (weights and bias) $\mathcal{S}$ and f satisfying $\|f\|_\infty \leq \mathcal{B}$ and $\|\frac{\partial}{\partial \tau} f\|_\infty \leq \mathcal{B}'$. Then for any $f \in \mathcal{F}_n$, the partial derivative $\frac{\partial}{\partial \tau} f$ can be implemented by a ReLU-ReQU activated multilayer perceptron with depth $3\mathcal{D}+3$, width $10\mathcal{W}$, number of neurons $17\mathcal{U}$, number of parameters $23\mathcal{S}$ and bound $\mathcal{B}'$.*

By Lemma 12, the partial derivative of a function in $\mathcal{F}_n$ can be implemented by a function in $\mathcal{F}_n'$. Consequently, for κ and κ_n given in (9) and (10),

$$\sup_{f \in \mathcal{F}_n} |\kappa(f) - \kappa_n(f)| \leq \sup_{f' \in \mathcal{F}_n'} |\tilde{\kappa}(f') - \tilde{\kappa}_n(f')|,$$

where $\tilde{\kappa}(f) = \mathbb{E}[\max\{-f(X,\xi), 0\}]$ and $\tilde{\kappa}_n(f) = \sum_{i=1}^n [\max\{-f(X_i,\xi_i), 0\}]/n$. Note that $\tilde{\kappa}$ and $\tilde{\kappa}_n$ are both 1-Lipschitz in f, thus an upper bound for $\sup_{f' \in \mathcal{F}_n'} |\tilde{\kappa}(f') - \tilde{\kappa}_n(f')|$ can be derived once the complexity of $\mathcal{F}_n'$ is known. The complexity of a function class can be measured in several ways, including Rademacher complexity, covering number, VC dimension and Pseudo dimension. These measures depict the complexity of a function class differently but are closely related to each other in many ways (a brief description of these measures can be found in Appendix B). Next, we give an upper bound on the Pseudo dimension of the function class $\mathcal{F}_n'$, which facilities our derivation of the upper bound for the stochastic error.

Lemma 13 (Pseudo dimension of ReLU-ReQU multilayer perceptrons) *Let $\mathcal{F}$ be a function class implemented by ReLU-ReQU activated multilayer perceptrons with depth no more than $\tilde{\mathcal{D}}$, width no more than $\tilde{\mathcal{W}}$, number of neurons (nodes) no more than $\tilde{\mathcal{U}}$ and size or number of parameters (weights and bias) no more than $\tilde{\mathcal{S}}$. Then the Pseudo dimension of $\mathcal{F}$ satisfies*

$$\mathrm{Pdim}(\mathcal{F}) \leq \min\{7\tilde{\mathcal{D}}\tilde{\mathcal{S}}(\tilde{\mathcal{D}} + \log_2 \tilde{\mathcal{U}}), 22\tilde{\mathcal{U}}\tilde{\mathcal{S}}\}.$$

Theorem 14 (Stochastic error bound) *Let $\mathcal{F}_n = \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be the ReQU activated multilayer perceptron and let $\mathcal{F}_n' = \{\frac{\partial}{\partial \tau} f : f \in \mathcal{F}_n\}$ denote the class of first order partial derivatives. Then for $n \geq \max\{\mathrm{Pdim}(\mathcal{F}_n), \mathrm{Pdim}(\mathcal{F}_n')\}$, the stochastic error satisfies*

$$\begin{aligned}\mathbb{E} \sup_{f \in \mathcal{F}_n} \big| [\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)] - [\mathcal{R}_{n,\lambda}^\xi(f) - \mathcal{R}_{n,\lambda}^\xi(f_0) \big| \\ \leq c_0 \sqrt{\big\{\mathcal{B}\mathrm{Pdim}(\mathcal{F}_n) + \lambda \mathcal{B}' \mathrm{Pdim}(\mathcal{F}_n')\big\}} \sqrt{\frac{\log(n)}{n}}, \end{aligned} \tag{23}$$

for some universal constant $c_0 > 0$. Also,

$$\begin{aligned}\mathbb{E} \sup_{f \in \mathcal{F}_n} \big| [\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)] - [\mathcal{R}_{n,\lambda}^\xi(f) - \mathcal{R}_{n,\lambda}^\xi(f_0)] \big| \\ \leq c_1 \big(\mathcal{B} + \lambda \mathcal{B}'\big) \sqrt{\min\{5796 \mathcal{D}\mathcal{S}(\mathcal{D} + \log_2 \mathcal{U}), 8602 \mathcal{U}\mathcal{S}\}} \sqrt{\frac{\log(n)}{n}},\end{aligned}$$

for some universal constant $c_1 > 0$.

The proofs of Lemma 13 and Theorem 14 are given in the Appendix.

5. Approximation error

In this section, we give an upper bound on the approximation error of the ReQU network for approximating functions in C^s defined in Definition 2.

The ReQU activation function has a continuous first order derivative and its first order derivative is the popular ReLU function. With ReQU as the activation function, the network is smooth and differentiable. Therefore, ReQU is a suitable choice for our problem since derivatives are involved in the penalty function.

An important property of ReQU is that it can represent the square function x^2 without error. In the study of ReLU network approximation properties (Yarotsky, 2017, 2018; Shen et al., 2020), the analyses rely essentially on the fact that x^2 can be approximated by deep ReLU networks to any error tolerance as long as the network is large enough. With ReQU activated networks, x^2 can be represented exactly with one hidden layer and 2 hidden neurons. ReQU can be more efficient in approximating smooth functions in the sense that it requires a smaller network size to achieve the same approximation error.

Now we state some basic approximation properties of ReQU networks. The analysis of the approximation power of ReQU networks in our work basically rests on the fact that given inputs $x, y \in \mathbb{R}$, the powers x, x^2 and the product xy can be exactly computed by simple ReQU networks. Let $\sigma_2(x) = [\max\{x, 0\}]^2$ denote the ReQU activation function. We first list the following basic properties of the ReQU approximation:

1. For any $x \in \mathbb{R}$, the square function x^2 can be computed by a ReQU network with 1 hidden layer and 2 neurons, i.e.,
$$x^2 = \sigma_2(x) + \sigma_2(-x).$$
2. For any $x, y \in \mathbb{R}$, the multiplication function xy can be computed by a ReQU network with 1 hidden layer and 4 neurons, i.e.,
$$xy = \frac{1}{4}\{\sigma_2(x+y) + \sigma_2(-x-y) - \sigma_2(x-y) - \sigma_2(-x+y)\}.$$

3. For any $x \in \mathbb{R}$, taking $y = 1$ in the above equation, then the identity map $x \mapsto x$ can be computed by a ReQU network with 1 hidden layer and 4 neurons, i.e.,

$$x = \frac{1}{4}\{\sigma_2(x+1) + \sigma_2(-x-1) - \sigma_2(x-1) - \sigma_2(-x+1)\}.$$

4. If both x and y are non-negative, the formulas for square function and multiplication can be simplified as follows:

$$x^2 = \sigma_2(x), \qquad xy = \frac{1}{4}\{\sigma_2(x+y) - \sigma_2(x-y) - \sigma_2(-x+y)\}.$$

The above equations can be verified using simple algebra. The realization of the identity map is not unique here, since for any $a \neq 0$, we have $x = \{(x+a)^2 - x^2 - a^2\}/(2a)$ which can be exactly realized by ReQU networks. In addition, the constant function 1 can be computed exactly by a 1-layer ReQU network with zero weight matrix and constant 1 bias vector. In such a case, the basis $1, x, x^2, \ldots, x^p$ of the degree $p \in \mathbb{N}_0$ polynomials in $\mathbb{R}$ can be computed by a ReQU network with proper size. Therefore, any p-degree polynomial can be approximated without error.

To approximate the square function in (1) with ReLU networks on bounded regions, the idea of using "sawtooth" functions was first raised in Yarotsky (2017), and it achieves an error $\mathcal{O}(2^{-L})$ with width 6 and depth $\mathcal{O}(L)$ for positive integer $L \in \mathbb{N}^+$. General construction of ReLU networks for approximating a square function can achieve an error N^{-L} with width $3N$ and depth L for any positive integers $N, L \in \mathbb{N}^+$ (Lu et al., 2021a). Based on this basic fact, the ReLU networks approximating multiplication and polynomials can be constructed correspondingly. However, the network complexity (cost) in terms of network size (depth and width) for a ReLU network to achieve precise approximation can be large compared to that of a ReQU network since ReQU network can compute polynomials exactly with fewer layers and neurons.

Theorem 15 (Approximation of Polynomials by ReQU networks) *For any non-negative integer $N \in \mathbb{N}_0$ and any positive integer $d \in \mathbb{N}^+$, if $f : \mathbb{R}^d \to \mathbb{R}$ is a polynomial of d variables with total degree N, then there exists a ReQU activated neural network that can compute f with no error. More exactly,*

1. *if $d = 1$ where $f(x) = \sum_{i=1}^N a_i x^i$ is a univariate polynomial with degree N, then there exists a ReQU neural network with $2N-1$ hidden layers , $5N-1$ number of neurons, $8N$ number of parameters (weights and bias) and network width 4 that computes f with no error.*

2. *If $d \geq 2$ where $f(x_1, \ldots, x_d) = \sum_{i_1+\ldots+i_d=0}^N a_{i_1,\ldots,i_d} x_1^{i_1} \cdots x_d^{i_d}$ is a multivariate polynomial of d variables with total degree N, then there exists a ReQU neural network with $2N-1$ hidden layers , $2(5N-1)N^{d-1} + (5N-1)\sum_{j=1}^{d-2} N^j \leq 15N^d$ number of neurons, $16N^d + 8N\sum_{j=1}^{d-2} N^j \leq 24N^d$ number of parameters (weights and bias) and network width $8N^{d-1} + 4\sum_{j=1}^{d-2} N^j \leq 12N^{d-1}$ that computes f with no error.*

Theorem 15 shows any d-variate multivariate polynomial with degree N on $\mathbb{R}^d$ can be represented with no error by a ReQU network with $2N-1$ hidden layers, no more than $15N^d$ neurons, no more than $24N^d$ parameters (weights and bias) and width less than $12N^{d-1}$. The approximation powers of ReQU networks (and RePU networks) on polynomials are studied in Li et al. (2019b,a), in which the representation of a d-variate multivariate polynomials with degree N on $\mathbb{R}^d$ needs a ReQU network with $d\lfloor \log_2 N \rfloor + d$ hidden layers, and no more than $\mathcal{O}(\binom{N+d}{d})$ neurons and parameters. Compared to the results in Li et al. (2019a,b), the orders of neurons and parameters for the constructed ReQU network in Theorem 15 are basically the same. The the number of hidden layers for the constructed ReQU network here is $2N-1$ depending only on the degree of the target polynomial and independent of the dimension of input d, which is different from the dimension depending $d\lfloor \log_2 N \rfloor + d$ hidden layers required in Li et al. (2019a). In addition, ReLU activated networks with width $\{9(W+1)+N-1\}N^d = \mathcal{O}(WN^d)$ and depth $7N^2L = \mathcal{O}(LN^2)$ can only approximate d-variate multivariate polynomial with degree N with an accuracy $9N(W+1)^{-7NL} = \mathcal{O}(NW^{-LN})$ for any positive integers $W, L \in \mathbb{N}^+$. Note that the approximation results on polynomials using ReLU networks are generally on bounded regions, while ReQU can exactly compute the polynomials on $\mathbb{R}^d$. In this sense, the approximation power of ReQU networks is generally greater than that of ReLU networks.

Next, we leverage the approximation power of multivariate polynomials to derive error bounds of approximating general multivariate smooth functions using ReQU activated neural networks. Here we focus on the approximation of multivariate smooth functions in C^s space for $s \in \mathbb{N}^+$ defined in Definition 2.

Theorem 16 *Let f be a real-valued function defined on $\mathcal{X} \times (0,1) \subset \mathbb{R}^{d+1}$ belonging to class C^s for $0 \le s < \infty$. For any $N \in \mathbb{N}^+$, there exists a ReQU activated neural network ϕ_N with width no more than $12N^d$, hidden layers no more than $2N-1$, number of neurons no more than $15N^{d+1}$ and parameters no more than $24N^{d+1}$ such that for each multi-index $\alpha \in \mathbb{N}_0^d$, we have $|\alpha|_1 \le \min\{s, N\}$,*

$$\sup_{\mathcal{X}\times(0,1)} |D^\alpha (f - \phi_N)| \le C_{s,d,\mathcal{X}}\, N^{-(s-|\alpha|_1)} \|f\|_{C^s},$$

where $C_{s,d,\mathcal{X}}$ is a positive constant depending only on d, s and the diameter of $\mathcal{X} \times (0,1)$.

In Li et al. (2019b,a), a similar rate of convergence $\mathcal{O}(N^{-(s-\alpha)})$ under the Jacobi-weighted L^2 norm was obtained for the approximation of α-th derivative of a univariate target function, where $\alpha \le s \le N+1$ and s denotes the smoothness of the target function belonging to Jacobi-weighted Sobolev space. The ReQU network in Li et al. (2019a) has a different shape from ours specified in Theorem 8. The results of Li et al. (2019a) achieved a $\mathcal{O}(N^{-(s-\alpha)})$ rate using a ReQU network with $\mathcal{O}(\log_2(N))$ hidden layers, $\mathcal{O}(N)$ neurons and nonzero weights and width $\mathcal{O}(N)$. Simultaneous approximation to the target function and its derivatives by a ReQU network was also considered in Duan et al. (2021) for solving partial differential equations for d-dimensional smooth target functions in C^2.

Now we assume that the target function $f_0 : \mathcal{X} \times (0,1) \to \mathbb{R}$ in our QRP estimation problem belongs to the smooth function class C^s for some $s \in \mathbb{N}^+$. The approximation error $\inf_{f\in\mathcal{F}_n} \Big[\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f_0) + \lambda\{\kappa(f) - \kappa(f_0)\}\Big]$ given in Lemma 1 can be handled correspondingly.

Corollary 17 (Approximation error bound) *Suppose that the target function f_0 defined in (2) belongs to C^s for some $s \in \mathbb{N}^+$. For any $N \in \mathbb{N}^+$, let $\mathcal{F}_n := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be the ReQU activated neural networks $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ with depth (number of hidden layer) $\mathcal{D} \leq 2N-1$, width $\mathcal{W} \leq 12N^d$, number of neurons $\mathcal{U} \leq 15N^{d+1}$, number of parameters (weights and bias) $\mathcal{S} \leq 24N^{d+1}$, satisfying $\mathcal{B} \geq \|f_0\|_{C^0}$ and $\mathcal{B}' \geq \|f_0\|_{C^1}$. Then the approximation error given in Lemma 1 satisfies*

$$\inf_{f \in \mathcal{F}_n} \Big[\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f_0) + \lambda\{\kappa(f) - \kappa(f_0)\}\Big] \leq C_{s,d,\mathcal{X}}(1+\lambda)N^{-(s-1)}\|f_0\|_{C^s},$$

where $C_{s,d,\mathcal{X}}$ is a positive constant depending only on d, s and the diameter of the support $\mathcal{X} \times (0,1)$.

6. Computation

In this section, we describe the training algorithms for the proposed penalized DQRP estimator, including a generic algorithm and an improved algorithm.

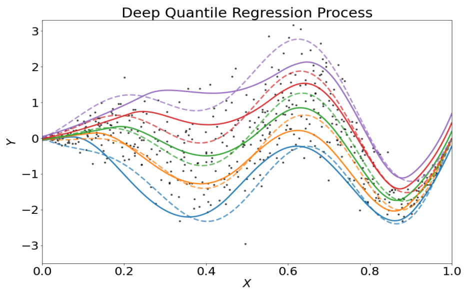
Algorithm 1 An stochastic gradient descent algorithm for the penalized DQRP estimator

```
Require: Sample data $\{(X_i, Y_i)\}_{i=1}^n$ with $n \geq 1$; Minibatch size $m \leq n$.
  Generate $n$ random values $\{\xi_i\}_{i=1}^n$ uniformly from $(0,1)$
  for number of training iterations do
      Sample minibatch of $m$ data $\{(X^{(j)}, Y^{(j)}, \xi^{(j)})\}_{j=1}^m$ form the data $\{(X_i, Y_i, \xi_i)\}_{i=1}^n$
      Update the ReQU network $f$ parameterized by $\theta$ by descending its stochastic gradient:
          $\nabla_\theta \frac{1}{m}\sum_{j=1}^m \Big[\rho_{\xi^{(j)}}(Y^{(j)} - f(X^{(j)}, \xi^{(j)})) + \lambda \max\Big\{ -\frac{\partial}{\partial \tau} f(X^{(j)}, \xi^{(j)}), 0\Big\}\Big]$
  end for
  The gradient-based updates can use any standard gradient-based algorithm. We used Adam in our experiments.
```

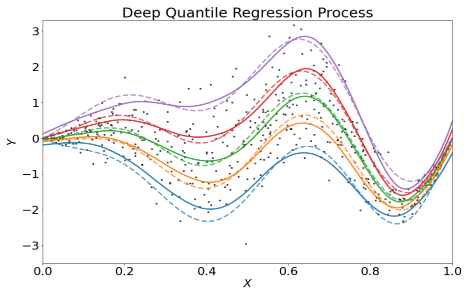
In Algorithm 1, the number of random values $\{\xi_i\}_{i=1}^n$ is set to be the same as the sample size n and each ξ_i is coupled with the sample (X_i, Y_i) for $i = 1, \ldots, n$ during the training process. This may degrade the efficiency of the learning DQRP $\hat{f}_n^\lambda$ since each data (X_i, Y_i) has only been used to train the ReQU network $f(\cdot, \xi_i)$ at a single value (quantile) ξ_i. Hence, we proposed an improved algorithm.

Algorithm 2 An improved stochastic gradient descent algorithm for the penalized DQRP estimator

```
Require: Sample data $\{(X_i, Y_i)\}_{i=1}^n$ with $n \geq 1$; Minibatch size $m \leq n$.
  for number of training iterations do
    Sample minibatch of $m$ data $\{(X^{(j)}, Y^{(j)})\}_{j=1}^m$ form the data $\{(X_i, Y_i)\}_{i=1}^n$
    Generate $m$ random values $\{\xi_j\}_{j=1}^m$ uniformly from $(0,1)$
    Update the ReQU network $f$ parameterized by $\theta$ by descending its stochastic gradient:
      $$\nabla_\theta \frac{1}{m}\sum_{j=1}^{m}\Big[\rho_{\xi_j}(Y^{(j)} - f(X^{(j)}, \xi_j)) + \lambda \max\Big\{-\frac{\partial}{\partial \tau} f(X^{(j)}, \xi_j), 0\Big\}\Big]$$
  end for
  The gradient-based updates can use any standard gradient-based algorithm. We used Adam in our experiments.
```



(a) Trained by Algorithm 1



(b) Trained by Algorithm 2

Figure 1: A comparison of Algorithms 1 and 2. The 512 training data generated from the "Wave" model are depicted as black dots. The target quantile functions at quantile levels 0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as dashed curves, and the estimated quantile functions are the solid curves with the same color. In the left panel, the estimator is trained by Algorithm 1. In the right panel, the estimator is trained by the improved Algorithm 2. Both trainings stop after 200 epochs.

In Algorithm 2, at each minibatch training iteration, m random values $\{\xi_j\}_{j=1}^m$ are generated uniformly from $(0,1)$ and coupled with the minibatch sample $\{(X^{(j)}, Y^{(j)})\}_{j=1}^m$ for the gradient-based updates. In this case, each sample (X_i, Y_i) gets involved in the training of ReQU network $f(\cdot, \xi)$ at multiple values (quantiles) of $\xi = \xi_i^{(1)}, \ldots, \xi_i^{(t)}$ where t denotes the number of minibatch iterations and $\xi_i^{(j)}, j = 1, \ldots, t$ denotes the random value generated at iteration t that is coupled with the sample (X_i, Y_i). In such a way, the utilization of each sample (X_i, Y_i) is greatly improved while the computation complexity does not increase compared to the generic Algorithm 1.

We use an example to demonstrate the advantage of Algorithm 2 over Algorithm 1. Figure 1 displays a comparison between Algorithm 1 and Algorithm 2 with the same simulated dataset generated from the "Wave" model (see section 7 for detailed introduction of the simulated model). The sample size $n = 512$, and the tuning parameter is chosen as

$\lambda = \log(n)$. Two ReQU neural networks with the same architecture (width of hidden layers $(256, 256, 256)$) are trained for 200 epochs by Algorithm 1 and Algorithm 2, respectively. The example and the simulation studies in section 7 show that Algorithm 2 has a better and more stable performance than Algorithm 1 without additional computational complexity.

7. Numerical studies

In this section, we compare the proposed penalized deep quantile regression with the following nonparametric quantile regression methods:

- Kernel-based nonparametric quantile regression (Sangnier et al., 2016), denoted by *kernel QR*. This is a joint quantile regression method based on a vector-valued reproducing kernel Hilbert space (RKHS), which enjoys fewer quantile crossings and enhanced performance compared to the estimation of the quantile functions separately. In our implementation, the radial basis function (RBF) kernel is chosen and a coordinate descent primal-dual algorithm (Fercoq and Bianchi, 2019) is used via the Python package *qreg*.

- Quantile regression forests (Meinshausen and Ridgeway, 2006), denoted by *QR Forest*. Conditional quantiles can be estimated using quantile regression forests, a method based on random forests. Quantile regression forests can nonparametrically estimate quantile regression functions with high-dimensional predictor variables. This method is shown to be consistent in Meinshausen and Ridgeway (2006).

- Deep quantile regression (Padilla et al., 2022; Shen et al., 2021), denoted by *DQR*. Especially, Padilla et al. (2022) discussed the extension to multiple (finitely many) quantiles estimation with non-crossing constraints using ReLU neural networks. We implement it in Python via *Pytorch* and use *Adam* (Kingma and Ba, 2014) algorithm with default learning rate 0.01 and default $\beta = (0.9, 0.99)$ (coefficients used for computing running averages of gradients and their squares).

- Penalized DQRP estimator as described in Section 2 using ReQU networks, denoted by *DQRP*. We implement it in Python via *Pytorch* and use *Adam* (Kingma and Ba, 2014) as the optimization algorithm with default learning rate 0.01 and default $\beta = (0.9, 0.99)$.

- Penalized DQRP estimator as described in Section 2 using ReLU networks, denoted by *DQRP**. We implement it in Python via *Pytorch* and use the same optimizer and parameters as those for *DQRP* using ReQU networks.

7.1 Estimation and Evaluation

For the proposed penalized DQRP (DQRP*) estimator, we set the tuning parameter $\lambda = \log(n)$ in the simulation stuides. We restrict the norm of neural network output and derivative to be bounded by $\mathcal{B}$ and $\mathcal{B}'$ respectively, and set $\mathcal{B} = \mathcal{B}' = 10 \times (\log n)^2$. The norm bound restriction is achieved by scaling the weights in the last layer of neural network during the optimization. We use rectangle networks with 3 hidden layers and width 128 for estimating univariate target functions, and we use rectangle networks with 3 hidden layers and width 256 for estimating multivariate target functions. We apply *Kernel QR* and *QR Forest* to estimate the quantile curves at 5 different levels for each simulated model, i.e., $\tau \in \{0.05, 0.25, 0.5, 0.75, 0.95\}$.

For each target f_0, according to model (1) we generate the training data $(X_i^{train}, Y_i^{train})_{i=1}^n$ with sample size n to train the empirical risk minimizer at $\tau \in \{0.05, 0.25, 0.5, 0.75, 0.95\}$ using Kernel QR and QR Forest, i.e.

$$\hat{f}_n^\tau \in \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \rho_\tau(Y_i^{train} - f(X_i^{train})),$$

where $\mathcal{F}$ is the class of RKHS, the class of functions for *QR forest*, or the class of ReQU neural network functions.

For each f_0, we also generate the testing data $(X_t^{test}, Y_t^{test})_{t=1}^T$ with sample size T from the same distribution of the training data. For the proposed method and for each obtained estimate $\hat{f}_n$, we denote $\hat{f}_n^\tau(\cdot) = \hat{f}_n(\cdot, \tau)$ for notational simplicity. For DQRP, Kernel QR and QR Forest, we calculate the testing error on $(X_t^{test}, Y_t^{test})_{t=1}^T$ at different quantile levels τ. For quantile level $\tau \in (0, 1)$, we calculate the L_1 distance between $\hat{f}_n^\tau$ and the corresponding risk minimizer $f_0^\tau(\cdot) := f_0(\cdot, \tau)$ by

$$\|\hat{f}_n^\tau - f_0^\tau\|_{L^1(\nu)} = \frac{1}{T} \sum_{t=1}^T \left| \hat{f}_n(X_t^{test}, \tau) - f_0^\tau(X_t^{test}, \tau) \right|,$$

and we also calculate the L_2^2 distance between $\hat{f}_n^\tau$ and the f_0^τ, i.e.

$$\|\hat{f}_n^\tau - f_0^\tau\|_{L^2(\nu)}^2 = \frac{1}{T} \sum_{t=1}^T \left| \hat{f}_n(X_t^{test}, \tau) - f_0^\tau(X_t^{test}, \tau) \right|^2.$$

The specific forms of f_0 are given in the data generation models below.

In the simulation studies, the size of testing data $T = 10^5$ for each data generation model. We report the mean and standard deviation of the L_1 and L_2^2 distances over $R = 100$ replications under different scenarios.

7.2 Univariate models

We consider three basic univariate models, including "Linear", "Wave" and "Triangle", which corresponds to different specifications of the target function f_0. The formulae are given below.

(a) Linear: $f_0(x, \tau) = 2x + F_t^{-1}(\tau)$,
(b) Wave: $f_0(x, \tau) = 2x \sin(4\pi x) + |\sin(\pi x)|\Phi^{-1}(\tau)$,
(c) Triangle: $f_0(x, \tau) = 4(1 - |x - 0.5|) + \exp(4x - 2)\Phi^{-1}(\tau)$,

where $F_t(\cdot)$ is the cumulative distribution function of the standard Student's t random variable, $\Phi(\cdot)$ is the cumulative distribution function of the standard normal random variable. We use the linear model as a baseline model in our simulations and expect all the methods perform well under the linear model. The "Wave" is a nonlinear smooth model and the "Triangle" is a nonlinear, continuous but non-differentiable model. These models are chosen so that we can evaluate the performance of *DQRP*, *DQRP**, *kernel QR* and *QR Forest* under different types of models.

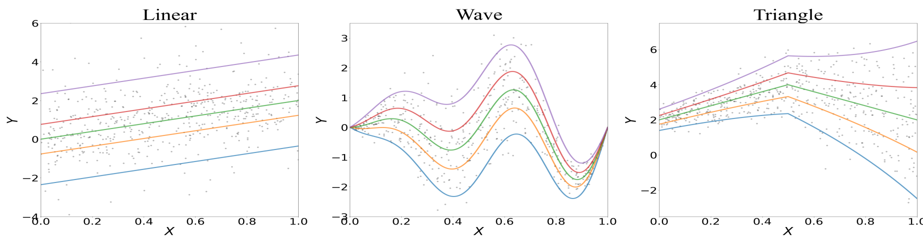


Figure 2: The target quantiles curves. From the left to the right, each column corresponds a data generation model, "Linear", "Wave" and "Triangle". The sample data with size $n = 512$ is depicted as grey dots.The target quantile functions at the quantile levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as solid curves.

For these models, we generate X uniformly from the unit interval $[0, 1]$. The τ-th conditional quantile of the response Y given $X = x$ can be calculated directly based on the expression of $f_0(x, \tau)$. Figure 2 shows all these univariate data generation models and their corresponding conditional quantile curves at $\tau = 0.05, 0.25, 0.50, 0.75, 0.95$.

Figures 3 and 4 show an instance of the estimated quantile curves for the "Wave" and "Triangle" models. The plot for the "Linear" model is included in the Appendix. In these plots, the training data is depicted as grey dots. The target quantile functions at the quantile levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as dashed curves, and the estimated quantile functions are represented by solid curves with the same color. For each figure, from the top to the bottom, the rows correspond to the sample size $n = 512, 2048$. From the left to the right, the columns correspond to the methods DQRP, DQRP*, kernel QR and QR Forest.

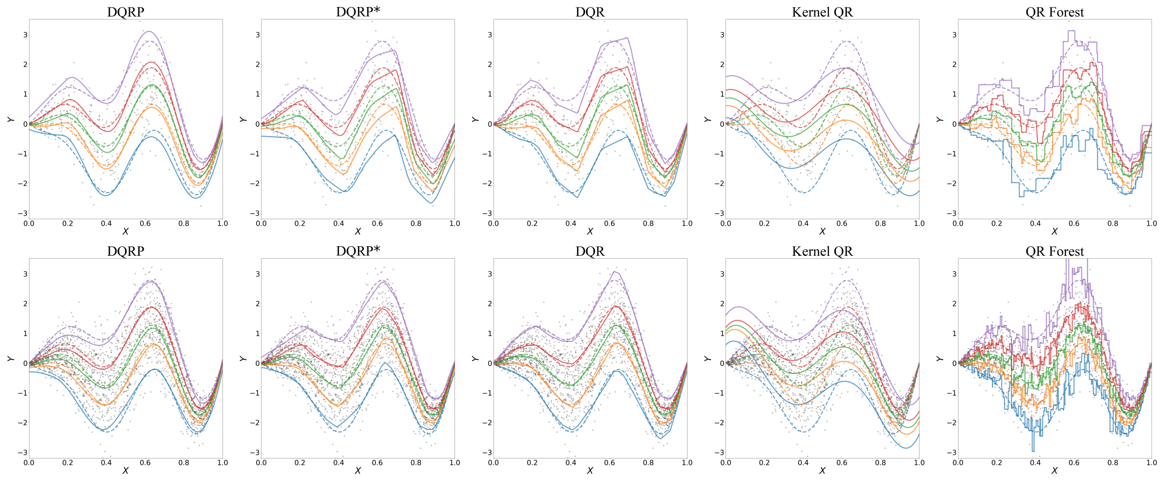


Figure 3: The fitted quantile curves under the univariate “Wave” model. The training data is depicted as grey dots. The target quantile functions at the quantile levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as dashed curves, and the estimated quantile functions are represented by solid curves with the same color. From the top to the bottom, the rows correspond to the sample size $n = 512, 2048$. From the left to the right, the columns correspond to the methods DQRP, DQRP*, DQR, kernel QR and QR Forest.

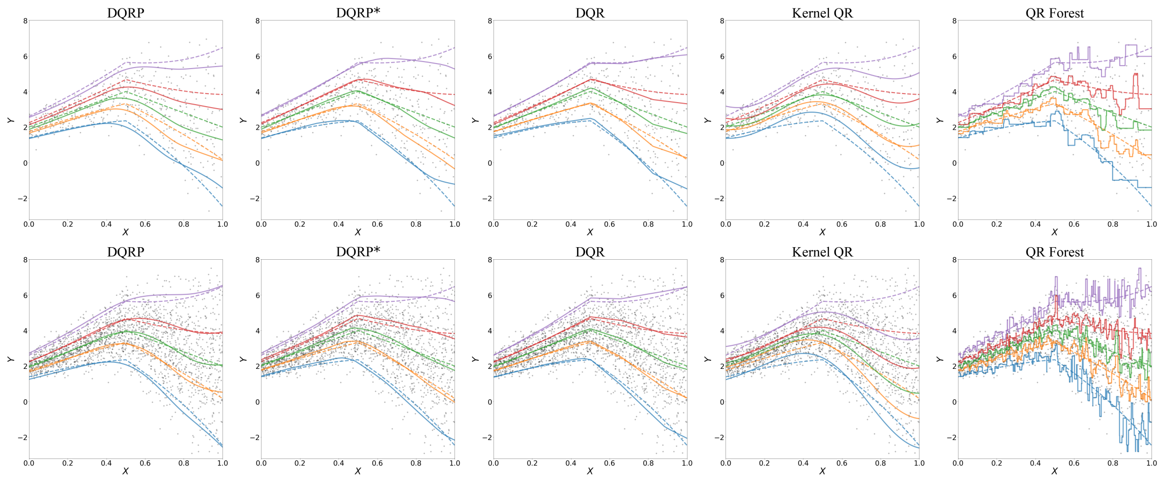


Figure 4: The fitted quantile curves under the univariate “Triangle” model. The training data is depicted as grey dots. The target quantile functions at the quantile levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as dashed curves, and the estimated quantile functions are represented by solid curves with the same color. From the top to the bottom, the rows correspond to the sample sizes $n = 512, 2048$. From the left to the right, the columns correspond to the methods DQRP, DQRP*, DQR, kernel QR and QR Forest.

Table 2: Data is generated from the "Wave" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	Sample size	$n = 512$		$n = 2048$	
τ	Method	L_1	L_2^2	L_1	L_2^2
0.05	DQRP	0.163(0.056)	0.046(0.034)	**0.093(0.027)**	**0.015(0.009)**
	DQRP*	0.178(0.046)	0.053(0.028)	0.113(0.024)	0.021(0.012)
	DQR	**0.161(0.041)**	**0.039(0.021)**	0.107(0.030)	0.016(0.010)
	Kernel QR	0.461(0.072)	0.377(0.125)	0.599(0.224)	0.600(0.470)
	QR Forest	0.228(0.030)	0.092(0.024)	0.195(0.017)	0.071(0.013)
0.25	DQRP	**0.111(0.030)**	**0.022(0.012)**	**0.075(0.026)**	**0.010(0.007)**
	DQRP*	0.133(0.035)	0.033(0.016)	0.087(0.028)	0.013(0.008)
	DQR	0.123(0.032)	0.022(0.012)	0.084(0.028)	0.011(0.006)
	Kernel QR	0.441(0.064)	0.298(0.109)	0.571(0.225)	0.545(0.460)
	QR Forest	0.166(0.024)	0.051(0.015)	0.143(0.012)	0.039(0.007)
0.5	DQRP	**0.104(0.027)**	**0.019(0.01)**	**0.073(0.023)**	**0.009(0.005)**
	DQRP*	0.124(0.035)	0.026(0.014)	0.080(0.029)	0.011(0.007)
	DQR	0.116(0.026)	0.020(0.009)	0.083(0.027)	0.010(0.006)
	Kernel QR	0.440(0.058)	0.289(0.105)	0.555(0.226)	0.530(0.461)
	QR Forest	0.157(0.024)	0.045(0.014)	0.137(0.010)	0.036(0.005)
0.75	DQRP	0.126(0.039)	0.027(0.016)	**0.084(0.027)**	0.014(0.008)
	DQRP*	0.134(0.039)	0.030(0.017)	0.091(0.031)	0.012(0.008)
	DQR	**0.123(0.029)**	**0.022(0.010)**	0.088(0.028)	**0.011(0.006)**
	Kernel QR	0.462(0.055)	0.322(0.107)	0.560(0.219)	0.546(0.462)
	QR Forest	0.168(0.022)	0.050(0.014)	0.146(0.013)	0.041(0.008)
0.95	DQRP	0.182(0.06)	0.057(0.038)	0.118(0.043)	0.023(0.017)
	DQRP*	0.180(0.049)	0.051(0.027)	0.121(0.039)	0.024(0.017)
	DQR	**0.158(0.041)**	**0.035(0.020)**	**0.102(0.030)**	**0.015(0.009)**
	Kernel QR	0.552(0.064)	0.469(0.120)	0.615(0.200)	0.648(0.462)
	QR Forest	0.224(0.030)	0.090(0.026)	0.198(0.018)	0.074(0.014)

Table 3: Data is generated from the "Triangle" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	Sample size	$n = 512$		$n = 2048$	
τ	Method	L_1	L_2^2	L_1	L_2^2
0.05	DQRP	**0.202(0.078)**	0.089(0.080)	0.145(0.072)	0.052(0.070)
	DQRP*	0.202(0.078)	0.089(0.070)	0.131(0.050)	0.034(0.026)
	DQR	0.216(0.072)	**0.089(0.068)**	**0.135(0.045)**	**0.037(0.026)**
	Kernel QR	0.533(0.147)	0.520(0.268)	0.515(0.200)	0.490(0.459)
	QR Forest	0.364(0.061)	0.282(0.115)	0.359(0.031)	0.264(0.053)
0.25	DQRP	0.164(0.063)	0.046(0.031)	0.092(0.034)	0.016(0.012)
	DQRP*	0.131(0.061)	**0.036(0.040)**	**0.090(0.039)**	**0.015(0.014)**
	DQR	**0.145(0.060)**	0.039(0.033)	0.099(0.038)	0.017(0.012)
	Kernel QR	0.249(0.084)	0.110(0.084)	0.308(0.138)	0.179(0.217)
	QR Forest	0.272(0.044)	0.155(0.061)	0.259(0.021)	0.140(0.026)
0.5	DQRP	0.170(0.066)	0.047(0.035)	0.097(0.038)	0.016(0.012)
	DQRP*	0.134(0.058)	0.032(0.025)	0.088(0.033)	0.013(0.009)
	DQR	**0.130(0.054)**	**0.030(0.030)**	**0.086(0.032)**	**0.012(0.008)**
	Kernel QR	0.164(0.063)	0.052(0.044)	0.231(0.134)	0.125(0.158)
	QR Forest	0.251(0.040)	0.131(0.053)	0.252(0.021)	0.133(0.026)
0.75	DQRP	0.198(0.082)	0.068(0.053)	0.117(0.052)	0.023(0.019)
	DQRP*	0.163(0.065)	0.047(0.032)	0.094(0.043)	0.015(0.013)
	DQR	**0.137(0.0528)**	**0.036(0.038)**	**0.090(0.035)**	**0.013(0.011)**
	Kernel QR	0.252(0.087)	0.109(0.081)	0.334(0.123)	0.192(0.166)
	QR Forest	0.261(0.043)	0.142(0.059)	0.266(0.024)	0.149(0.031)
0.95	DQRP	0.295(0.134)	0.145(0.106)	0.168(0.078)	0.052(0.048)
	DQRP*	0.282(0.133)	0.183(0.165)	0.174(0.063)	0.074(0.046)
	DQR	**0.198(0.095)**	**0.084(0.098)**	**0.122(0.039)**	**0.026(0.014)**
	Kernel QR	0.540(0.123)	0.522(0.242)	0.541(0.175)	0.527(0.363)
	QR Forest	0.359(0.057)	0.256(0.090)	0.357(0.033)	0.259(0.053)

Tables 2 and 3 summarize the results for the models "Wave" and "Triangle", respectively. For Kernel QR, QR Forest, Deep Quantile Regression (DQR) and our proposed DQRP (DQRP*) estimator, the corresponding L_1 and L_2^2 errors (standard deviation in parentheses) between the estimates and the target are reported at different quantile levels $\tau = 0.05, 0.25, 0.50, 0.75, 0.95$. For each column, using bold text we highlight the best method which produces the smallest risk among these three methods. For the smooth "Wave" model, the DQR and proposed DQRP, DQRP* perform similarly, and they outperforms Kernel QR and QR Forest in all the scenarios. For the nonlinear and nonsmooth

"Triangle" model, with ReLU activation function, DQRP* and DQR also tends to perform better than DQRP, Kernel QR and QR Forest. DQR performs slightly better than our proposed DQRP* where DQR focuses on the estimation of curves at finite quantiles ($\tau = 0.05, 0.25, 0.5, 0.75, 0.95$), and DQRP* focuses on the estimation of the whole quantile process. For the "Linear" model, the results from the three methods are comparable, all methods except QR Forest have similar performance, among which DQR and Kernel QR tends to have slightly better performance. The results for the "Linear" model are given in Table 11 in the Appendix.

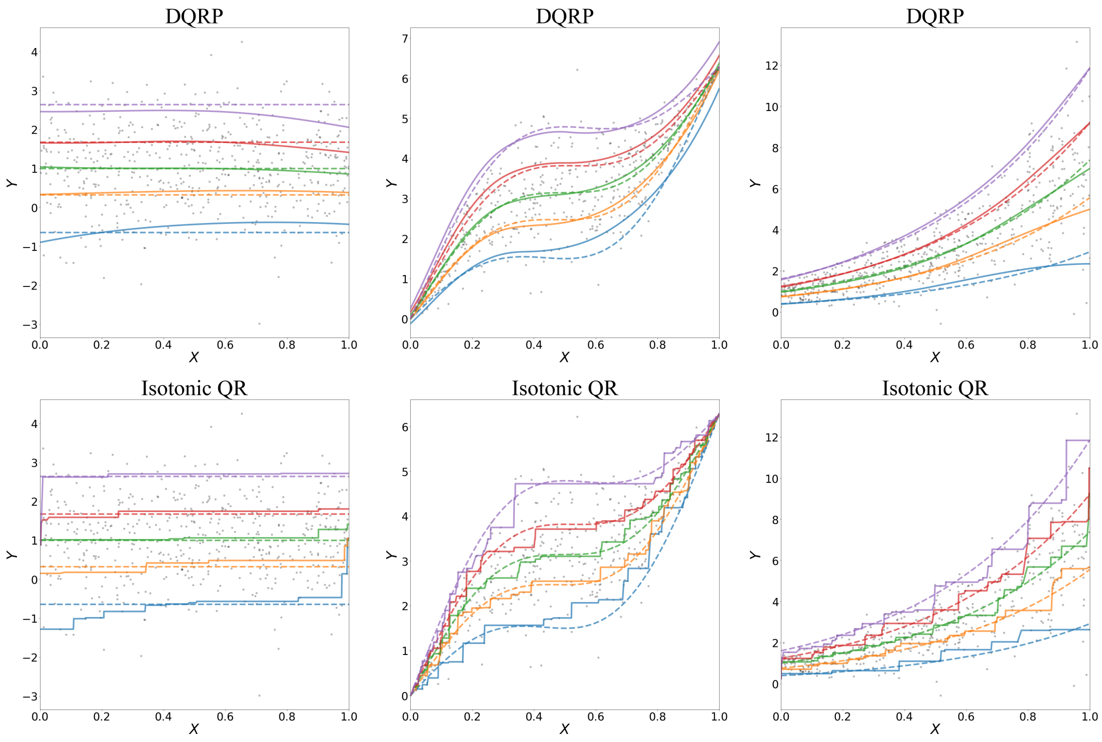


Figure 5: The fitted quantile curves under the univariate isotonic models: "Constant", "Swing" and "Exp" with sample size $n = 512$. The training data is depicted as grey dots. The target quantile functions at the quantile levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as dashed curves, and the estimated quantile functions are represented by solid curves with the same color. From left to the right, the columns correspond to the model "Constant", "Swing" and "Exp".

We note that DQR and Kernel QR are point quantile estimations, which have different focus compared to our proposed DQRP quantile process estimator. In this regard, we compare the performance of DQRP, DQRP*, and the interpolated DQR and QR Forest for quantile process estimation. In particular, to obtain the quantile process estimation from DQR and QR Forest, we use a two-step procedure: firstly, we run DQR and QR Forest at quantile level $\tau = 0.05, 0.25, 0.5, 0.75, 0.95$ and then interpolate these conditional quantile functions with respect to quantile level τ using a 3 hidden-layers ReLU neural networks with width 256. We denote the interpolated DQR estimator by DQR* and the interpolated

QR Forest estimator by QR Forest*. The simulation results are summarized in Tables 4-5.

Table 4: Data is generated from the "Wave" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The DQR* and QRF* estimators are obtained by interpolating DQR and QRF estimations at quantile level $\tau = 0.05, 0.25, 0.5, 0.75, 0.95$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	$n = 512$				$n = 2048$			
Method	DQRP	DQRP*	DQR*	QR Forest*	DQRP	DQRP*	DQR*	QR Forest*
τ		L_1				L_1		
0.05	0.162(0.058)	0.183(0.051)	**0.154(0.039)**	0.535(0.115)	0.125(0.065)	**0.123(0.034)**	0.148(0.088)	0.562(0.122)
0.1	**0.125(0.037)**	0.142(0.039)	0.158(0.048)	0.515(0.100)	0.109(0.065)	**0.108(0.045)**	0.184(0.096)	0.528(0.107)
0.15	**0.112(0.030)**	0.133(0.039)	0.141(0.047)	0.407(0.088)	0.098(0.056)	**0.096(0.041)**	0.160(0.084)	0.409(0.096)
0.2	**0.105(0.027)**	0.126(0.039)	0.120(0.038)	0.273(0.079)	0.091(0.046)	**0.088(0.034)**	0.119(0.066)	0.273(0.088)
0.25	**0.099(0.025)**	0.118(0.038)	0.111(0.033)	0.178(0.065)	0.087(0.037)	**0.083(0.030)**	0.094(0.042)	0.176(0.076)
0.3	**0.096(0.024)**	0.113(0.038)	0.108(0.033)	0.151(0.055)	0.083(0.030)	**0.080(0.028)**	0.089(0.032)	0.143(0.056)
0.35	**0.093(0.025)**	0.111(0.038)	0.110(0.033)	0.143(0.052)	0.080(0.025)	**0.077(0.028)**	0.086(0.029)	0.130(0.045)
0.4	**0.092(0.026)**	0.111(0.038)	0.110(0.033)	0.139(0.048)	0.078(0.022)	**0.075(0.028)**	0.083(0.027)	0.120(0.040)
0.45	**0.093(0.029)**	0.112(0.038)	0.111(0.032)	0.135(0.044)	0.076(0.021)	**0.075(0.027)**	0.078(0.027)	0.114(0.038)
0.5	**0.096(0.032)**	0.114(0.040)	0.113(0.031)	0.133(0.041)	**0.075(0.022)**	0.075(0.027)	0.076(0.026)	0.114(0.037)
0.55	**0.100(0.034)**	0.117(0.042)	0.114(0.030)	0.136(0.040)	0.076(0.024)	**0.075(0.027)**	0.078(0.026)	0.118(0.037)
0.6	**0.105(0.037)**	0.120(0.043)	0.116(0.030)	0.143(0.040)	0.078(0.027)	**0.076(0.026)**	0.082(0.024)	0.126(0.042)
0.65	**0.110(0.039)**	0.123(0.044)	0.121(0.033)	0.152(0.040)	0.081(0.028)	**0.077(0.026)**	0.084(0.023)	0.137(0.048)
0.7	**0.115(0.040)**	0.127(0.044)	0.127(0.038)	0.165(0.042)	0.085(0.028)	**0.079(0.026)**	0.082(0.024)	0.149(0.060)
0.75	**0.121(0.040)**	0.133(0.045)	0.133(0.044)	0.198(0.057)	0.089(0.028)	**0.081(0.026)**	0.083(0.027)	0.189(0.078)
0.8	**0.129(0.041)**	0.141(0.044)	0.151(0.057)	0.278(0.093)	0.094(0.029)	**0.082(0.026)**	0.102(0.038)	0.286(0.103)
0.85	**0.142(0.042)**	0.150(0.045)	0.186(0.071)	0.396(0.134)	0.100(0.032)	**0.087(0.026)**	0.144(0.054)	0.421(0.115)
0.9	0.165(0.049)	**0.160(0.044)**	0.210(0.076)	0.504(0.158)	0.105(0.034)	**0.102(0.031)**	0.174(0.061)	0.537(0.119)
0.95	0.227(0.075)	0.202(0.058)	**0.186(0.066)**	0.532(0.174)	**0.119(0.044)**	0.121(0.029)	0.138(0.047)	0.570(0.122)
τ		L_2^2				L_2^2		
0.05	0.042(0.028)	0.056(0.027)	**0.040(0.017)**	0.394(0.157)	0.028(0.028)	**0.026(0.012)**	0.037(0.046)	0.410(0.158)
0.1	**0.025(0.013)**	0.033(0.015)	0.043(0.023)	0.357(0.129)	0.023(0.026)	**0.020(0.015)**	0.053(0.054)	0.363(0.137)
0.15	**0.020(0.010)**	0.029(0.015)	0.034(0.021)	0.227(0.088)	0.018(0.020)	**0.016(0.014)**	0.041(0.042)	0.226(0.100)
0.2	**0.018(0.009)**	0.026(0.014)	0.024(0.016)	0.110(0.054)	0.015(0.014)	**0.013(0.011)**	0.023(0.026)	0.107(0.062)
0.25	**0.016(0.009)**	0.023(0.014)	0.021(0.012)	0.054(0.036)	0.013(0.010)	**0.012(0.009)**	0.014(0.012)	0.050(0.036)
0.3	**0.015(0.009)**	0.022(0.013)	0.020(0.012)	0.041(0.029)	0.012(0.008)	**0.011(0.008)**	0.012(0.008)	0.034(0.024)
0.35	**0.015(0.010)**	0.021(0.014)	0.021(0.012)	0.037(0.024)	0.011(0.006)	**0.010(0.008)**	0.012(0.008)	0.028(0.018)
0.4	**0.015(0.011)**	0.021(0.014)	0.021(0.013)	0.034(0.021)	**0.010(0.005)**	0.010(0.007)	0.011(0.007)	0.025(0.015)
0.45	**0.015(0.012)**	0.022(0.015)	0.021(0.012)	0.031(0.018)	**0.009(0.005)**	0.009(0.007)	0.010(0.006)	0.023(0.014)
0.5	**0.016(0.013)**	0.022(0.016)	0.021(0.011)	0.030(0.017)	**0.009(0.005)**	0.009(0.007)	0.009(0.006)	0.022(0.013)
0.55	**0.017(0.013)**	0.023(0.016)	0.022(0.011)	0.031(0.017)	**0.009(0.005)**	0.009(0.006)	0.010(0.006)	0.024(0.014)
0.6	**0.019(0.014)**	0.024(0.016)	0.023(0.012)	0.034(0.019)	0.010(0.006)	**0.010(0.006)**	0.011(0.006)	0.027(0.016)
0.65	**0.020(0.015)**	0.025(0.017)	0.025(0.014)	0.037(0.020)	0.011(0.007)	**0.010(0.006)**	0.011(0.006)	0.030(0.020)
0.7	**0.022(0.015)**	0.026(0.017)	0.028(0.017)	0.043(0.022)	0.011(0.007)	**0.010(0.007)**	0.011(0.006)	0.036(0.025)
0.75	**0.024(0.016)**	0.028(0.017)	0.031(0.020)	0.060(0.031)	0.012(0.008)	0.011(0.007)	**0.011(0.007)**	0.054(0.036)
0.8	**0.027(0.016)**	0.031(0.018)	0.039(0.030)	0.112(0.060)	0.014(0.009)	**0.011(0.007)**	0.017(0.012)	0.112(0.070)
0.85	**0.032(0.019)**	0.036(0.021)	0.057(0.043)	0.219(0.116)	0.016(0.011)	**0.012(0.007)**	0.031(0.021)	0.226(0.113)
0.9	0.043(0.025)	**0.041(0.024)**	0.071(0.051)	0.345(0.176)	0.018(0.013)	**0.016(0.009)**	0.042(0.025)	0.356(0.145)
0.95	0.078(0.047)	0.063(0.036)	**0.057(0.043)**	0.391(0.209)	0.024(0.017)	**0.023(0.010)**	0.029(0.017)	0.400(0.156)

Table 5: Data is generated from the "Triangle" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The DQR* and QRF* estimators are obtained by interpolating DQR and QRF estimations at quantile level $\tau = 0.05, 0.25, 0.5, 0.75, 0.95$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	$n = 512$				$n = 2048$			
Method	DQRP	DQRP*	DQR*	QR Forest*	DQRP	DQRP*	DQR*	QR Forest*
τ		L_1				L_1		
0.05	0.232(0.121)	0.241(0.107)	**0.214(0.122)**	0.525(0.175)	0.133(0.061)	**0.126(0.036)**	0.167(0.062)	0.652(0.168)
0.1	0.197(0.105)	**0.172(0.070)**	0.212(0.134)	0.536(0.129)	0.120(0.053)	**0.095(0.030)**	0.223(0.074)	0.634(0.146)
0.15	0.180(0.094)	**0.162(0.073)**	0.169(0.098)	0.422(0.103)	0.110(0.048)	**0.088(0.026)**	0.179(0.064)	0.494(0.124)
0.2	0.169(0.080)	0.154(0.073)	**0.144(0.060)**	0.292(0.102)	0.104(0.045)	**0.083(0.026)**	0.119(0.050)	0.339(0.106)
0.25	0.161(0.069)	0.145(0.070)	**0.134(0.050)**	0.213(0.083)	0.102(0.044)	**0.079(0.023)**	0.086(0.039)	0.225(0.089)
0.3	0.158(0.063)	0.137(0.067)	**0.123(0.049)**	0.179(0.062)	0.102(0.044)	**0.075(0.022)**	0.075(0.034)	0.172(0.079)
0.35	0.158(0.061)	0.129(0.063)	**0.118(0.049)**	0.160(0.054)	0.101(0.045)	0.072(0.024)	**0.070(0.030)**	0.140(0.070)
0.4	0.160(0.063)	0.123(0.059)	**0.117(0.050)**	0.147(0.050)	0.100(0.045)	0.070(0.025)	**0.067(0.028)**	0.121(0.061)
0.45	0.164(0.065)	0.121(0.056)	**0.119(0.052)**	0.139(0.050)	0.100(0.045)	0.071(0.026)	**0.066(0.027)**	0.109(0.053)
0.5	0.168(0.067)	0.121(0.055)	**0.120(0.056)**	0.139(0.046)	0.103(0.046)	0.073(0.026)	**0.068(0.027)**	0.104(0.046)
0.55	0.171(0.069)	0.125(0.055)	**0.121(0.058)**	0.141(0.045)	0.107(0.047)	0.074(0.026)	**0.071(0.027)**	0.104(0.042)
0.6	0.176(0.070)	0.130(0.057)	**0.123(0.058)**	0.145(0.050)	0.111(0.048)	0.076(0.026)	**0.075(0.025)**	0.112(0.042)
0.65	0.181(0.071)	0.136(0.059)	**0.126(0.056)**	0.156(0.060)	0.114(0.050)	0.078(0.026)	**0.078(0.025)**	0.125(0.049)
0.7	0.188(0.072)	0.144(0.061)	**0.131(0.056)**	0.178(0.073)	0.117(0.050)	0.082(0.028)	**0.080(0.031)**	0.146(0.060)
0.75	0.198(0.075)	0.152(0.062)	**0.141(0.060)**	0.222(0.093)	0.122(0.049)	0.089(0.032)	**0.086(0.049)**	0.196(0.081)
0.8	0.211(0.083)	**0.158(0.059)**	0.169(0.076)	0.304(0.129)	0.129(0.050)	**0.103(0.036)**	0.124(0.079)	0.332(0.104)
0.85	0.227(0.095)	**0.159(0.047)**	0.222(0.107)	0.437(0.146)	0.141(0.055)	**0.116(0.038)**	0.193(0.096)	0.514(0.121)
0.9	0.242(0.107)	**0.169(0.041)**	0.260(0.121)	0.546(0.168)	0.157(0.059)	**0.122(0.037)**	0.229(0.107)	0.664(0.146)
0.95	0.263(0.114)	0.272(0.093)	**0.227(0.105)**	0.532(0.191)	0.191(0.081)	0.172(0.048)	**0.164(0.099)**	0.675(0.186)
τ		L_2^2				L_2^2		
0.05	0.128(0.143)	0.142(0.130)	**0.111(0.145)**	0.555(0.513)	0.038(0.048)	**0.035(0.025)**	0.053(0.042)	0.654(0.332)
0.1	0.085(0.104)	**0.062(0.058)**	0.101(0.147)	0.527(0.336)	0.029(0.028)	**0.016(0.010)**	0.087(0.060)	0.619(0.278)
0.15	0.068(0.080)	**0.053(0.047)**	0.063(0.078)	0.319(0.171)	0.023(0.021)	**0.014(0.008)**	0.059(0.045)	0.382(0.182)
0.2	0.056(0.058)	0.048(0.043)	**0.040(0.029)**	0.159(0.104)	0.020(0.018)	**0.012(0.008)**	0.027(0.023)	0.184(0.101)
0.25	0.049(0.042)	0.044(0.038)	**0.037(0.024)**	0.088(0.069)	0.019(0.017)	**0.011(0.007)**	0.015(0.013)	0.088(0.056)
0.3	0.045(0.034)	0.039(0.035)	**0.031(0.021)**	0.059(0.048)	0.018(0.018)	**0.010(0.006)**	0.011(0.010)	0.052(0.039)
0.35	0.044(0.030)	0.034(0.031)	**0.028(0.022)**	0.048(0.038)	0.018(0.018)	**0.009(0.006)**	0.009(0.009)	0.035(0.031)
0.4	0.045(0.031)	0.031(0.028)	**0.027(0.025)**	0.041(0.032)	0.018(0.018)	0.009(0.006)	**0.008(0.008)**	0.027(0.025)
0.45	0.047(0.032)	0.029(0.026)	**0.028(0.028)**	0.038(0.027)	0.018(0.017)	0.009(0.006)	**0.008(0.007)**	0.021(0.020)
0.5	0.049(0.034)	**0.029(0.025)**	0.030(0.031)	0.037(0.025)	0.019(0.016)	**0.009(0.007)**	0.009(0.007)	0.019(0.017)
0.55	0.051(0.035)	**0.030(0.025)**	0.030(0.031)	0.038(0.024)	0.020(0.016)	0.010(0.007)	**0.009(0.007)**	0.019(0.014)
0.6	0.054(0.037)	0.033(0.027)	**0.031(0.031)**	0.041(0.028)	0.022(0.017)	**0.010(0.007)**	0.010(0.007)	0.020(0.014)
0.65	0.058(0.041)	0.036(0.029)	**0.031(0.030)**	0.049(0.037)	0.023(0.017)	0.011(0.007)	**0.011(0.007)**	0.025(0.017)
0.7	0.063(0.046)	0.039(0.032)	**0.033(0.029)**	0.065(0.052)	0.024(0.018)	0.012(0.007)	**0.011(0.007)**	0.035(0.025)
0.75	0.071(0.054)	0.042(0.034)	**0.036(0.028)**	0.099(0.077)	0.026(0.018)	0.014(0.009)	**0.013(0.015)**	0.065(0.043)
0.8	0.081(0.066)	**0.044(0.031)**	0.055(0.046)	0.179(0.130)	0.029(0.021)	**0.019(0.012)**	0.029(0.035)	0.176(0.095)
0.85	0.096(0.083)	**0.044(0.024)**	0.100(0.084)	0.328(0.208)	0.035(0.025)	**0.023(0.013)**	0.064(0.062)	0.404(0.174)
0.9	0.112(0.108)	**0.055(0.030)**	0.134(0.111)	0.478(0.294)	0.045(0.030)	**0.027(0.013)**	0.087(0.079)	0.649(0.281)
0.95	0.146(0.156)	0.164(0.100)	**0.102(0.080)**	0.453(0.327)	0.071(0.055)	0.077(0.044)	**0.047(0.056)**	0.666(0.363)

With the Isotonic Quantile Regression (denoted by Isotonic QR) in Mösching and Dümbgen (2020), we conduct simulation studies for isotonic univariate models, where the conditional quantile of response is increasing in the covariate (f_0 is in increasing in its first argument in our model setup). We simulate data from three models described below, including "Constant", "Swing" and "Exp", which corresponds to different specifications of the target function f_0.

(a) Constant: $f_0(x, \tau) = 1 + \Phi^{-1}(\tau)$;

(b) Swing: $f_0(x, \tau) = 2\pi x + \sin(2\pi x) + |\sin(\pi x)|\Phi^{-1}(\tau)$;

(c) Exp: $f_0(x,\tau) = \exp(2x) + \exp(2x-1)\Phi^{-1}(\tau)$,

Table 6: Data are generated from the isotonic models with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

Model	τ	Sample size / Method	$n=512$ L_1	$n=512$ L_2^2	$n=2048$ L_1	$n=2048$ L_2^2
Constant	0.05	DQRP	0.171(0.097)	0.047(0.059)	0.153(0.081)	0.036(0.036)
		Isotonic QR	0.144(0.068)	0.057(0.037)	0.079(0.035)	0.018(0.009)
	0.25	DQRP	0.080(0.041)	0.011(0.009)	0.059(0.032)	0.005(0.005)
		Isotonic QR	0.089(0.034)	0.022(0.012)	0.048(0.022)	0.006(0.004)
	0.5	DQRP	0.083(0.037)	0.011(0.008)	0.072(0.043)	0.008(0.009)
		Isotonic QR	0.086(0.032)	0.020(0.010)	0.046(0.018)	0.006(0.003)
	0.75	DQRP	0.126(0.066)	0.025(0.025)	0.119(0.065)	0.020(0.019)
		Isotonic QR	0.100(0.046)	0.025(0.015)	0.047(0.020)	0.006(0.003)
	0.95	DQRP	0.207(0.116)	0.067(0.064)	0.198(0.109)	0.056(0.054)
		Isotonic QR	0.132(0.059)	0.054(0.029)	0.070(0.032)	0.015(0.007)
Swing	0.05	DQRP	0.191(0.050)	0.060(0.028)	0.127(0.056)	0.033(0.037)
		Isotonic QR	1.488(0.076)	4.168(0.348)	1.478(0.035)	4.140(0.155)
	0.25	DQRP	0.101(0.029)	0.016(0.010)	0.098(0.052)	0.018(0.022)
		Isotonic QR	1.473(0.068)	3.695(0.283)	1.461(0.031)	3.673(0.132)
	0.5	DQRP	0.097(0.046)	0.015(0.013)	0.097(0.062)	0.017(0.022)
		Isotonic QR	1.471(0.063)	3.604(0.257)	1.459(0.029)	3.583(0.122)
	0.75	DQRP	0.131(0.083)	0.029(0.038)	0.126(0.076)	0.027(0.030)
		Isotonic QR	1.475(0.064)	3.703(0.268)	1.459(0.030)	3.673(0.126)
	0.95	DQRP	0.203(0.100)	0.070(0.073)	0.187(0.096)	0.060(0.056)
		Isotonic QR	1.513(0.072)	4.243(0.336)	1.471(0.034)	4.129(0.143)
Exp	0.05	DQRP	0.200(0.109)	0.086(0.086)	0.153(0.094)	0.050(0.060)
		Isotonic QR	0.886(0.073)	1.434(0.265)	0.832(0.046)	1.166(0.146)
	0.25	DQRP	0.159(0.094)	0.056(0.060)	0.130(0.082)	0.039(0.042)
		Isotonic QR	1.522(0.065)	3.787(0.325)	1.523(0.032)	3.727(0.164)
	0.5	DQRP	0.176(0.105)	0.065(0.080)	0.150(0.099)	0.049(0.059)
		Isotonic QR	1.994(0.081)	6.427(0.483)	2.013(0.039)	6.484(0.250)
	0.75	DQRP	0.212(0.142)	0.094(0.120)	0.183(0.132)	0.073(0.107)
		Isotonic QR	2.486(0.103)	9.972(0.754)	2.507(0.050)	10.045(0.394)
	0.95	DQRP	0.308(0.210)	0.197(0.278)	0.278(0.189)	0.157(0.197)
		Isotonic QR	3.151(0.132)	16.01(1.235)	3.212(0.073)	16.487(0.725)

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal random variable. The "Constant" is a linear isotonic model, and the "Swing" and "Exp" are nonlinear, continuous isotonic models. We set sample size $n = 512, 2048$ and replicate for 100 times.

The results are summarized in Table 6 and a visualization is presented in Figure 5. One can see that our DQRP method performs reasonably well and comparable to the isotonic quantile regression in Mösching and Dümbgen (2020) in the isotonic univariate case.

7.3 Multivariate models

We consider three basic multivariate models, including linear model ("Linear"), single index model ("SIM") and additive model ("Additive"), which correspond to different specifications of the target function f_0. The formulae are given below.

(a) Linear:

$$f_0(x, \tau) = 2A^\top x + F_t^{-1}(\tau),$$

(b) Single index model:

$$f_0(x, \tau) = \exp(0.1 \times A^\top x) + |\sin(\pi B^\top x)|\Phi^{-1}(\tau),$$

(c) Additive model:

$$f_0(x, \tau) = 3x_1 + 4(x_2 - 0.5)^2 + 2\sin(\pi x_3) - 5|x_4 - 0.5| + \exp\{0.1(B^\top x - 0.5)\}\Phi^{-1}(\tau),$$

where $F_t(\cdot)$ denotes the cumulative distribution function of the standard Student's t random variable, $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal random variable and the parameters (d-dimensional vectors)

$$\begin{aligned} A =& (0.409, 0.908, 0, 0, -2.061, 0.254, 3.024, 1.280)^\top, \\ B =& (1.386, -0.902, 5.437, 0, 0, -0.482, 4.611, 0)^\top. \end{aligned}$$

Table 7: Data is generated from the "Single index model" with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	Sample size	$n = 512$		$n = 2048$	
τ	Method	L_1	L_2^2	L_1	L_2^2
0.05	DQRP	0.529(0.037)	0.382(0.059)	0.433(0.016)	0.255(0.021)
	DQRP*	0.482(0.036)	0.326(0.052)	**0.421(0.011)**	**0.243(0.010)**
	DQR	0.635(0.050)	0.606(0.101)	0.564(0.032)	0.379(0.061)
	Kernel QR	0.641(0.043)	0.596(0.078)	0.620(0.030)	0.561(0.059)
	QR Forest	**0.460(0.012)**	**0.318(0.031)**	0.450(0.004)	0.305(0.013)
0.25	DQRP	0.228(0.024)	0.084(0.020)	0.195(0.013)	0.057(0.008)
	DQRP*	0.247(0.027)	0.097(0.022)	**0.191(0.010)**	**0.054(0.008)**
	DQR	0.415(0.040)	0.281(0.054)	0.338(0.028)	0.185(0.029)
	Kernel QR	0.462(0.043)	0.330(0.054)	0.486(0.043)	0.361(0.062)
	QR Forest	**0.214(0.010)**	**0.065(0.006)**	0.207(0.005)	0.058(0.003)
0.5	DQRP	0.179(0.021)	0.058(0.016)	0.098(0.011)	0.021(0.005)
	DQRP*	0.167(0.033)	0.048(0.019)	0.082(0.017)	0.011(0.004)
	DQR	0.329(0.044)	0.184(0.048)	0.241(0.028)	0.100(0.022)
	Kernel QR	0.339(0.035)	0.188(0.036)	0.346(0.048)	0.193(0.053)
	QR Forest	**0.081(0.010)**	**0.010(0.003)**	**0.058(0.005)**	**0.005(0.001)**
0.75	DQRP	0.269(0.021)	0.120(0.024)	0.209(0.013)	0.067(0.008)
	DQRP*	0.235(0.028)	0.086(0.022)	**0.183(0.010)**	**0.049(0.006)**
	DQR	0.422(0.043)	0.281(0.057)	0.341(0.031)	0.184(0.032)
	Kernel QR	0.453(0.047)	0.317(0.059)	0.492(0.044)	0.368(0.063)
	QR Forest	**0.213(0.011)**	**0.064(0.007)**	0.207(0.005)	0.058(0.003)
0.95	DQRP	**0.403(0.020)**	0.360(0.046)	0.459(0.029)	0.287(0.035)
	DQRP*	0.405(0.032)	0.338(0.070)	0.453(0.025)	**0.267(0.030)**
	DQR	0.643(0.049)	0.614(0.096)	0.561(0.027)	0.468(0.044)
	Kernel QR	0.637(0.044)	0.589(0.080)	0.627(0.033)	0.572(0.066)
	QR Forest	0.459(0.010)	**0.317(0.028)**	**0.451(0.0025)**	0.306(0.014)

Table 8: Data is generated from the "Additive" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	Sample size	$n = 512$		$n = 2048$	
τ	Method	L_1	L_2^2	L_1	L_2^2
0.05	DQRP	0.928(0.130)	1.367(0.305)	**0.542(0.095)**	**0.473(0.134)**
	DQRP*	**0.752(0.142)**	0.958(0.372)	0.544(0.148)	0.462(0.225)
	DQR	0.934(0.167)	1.400(0.467)	0.636(0.105)	0.665(0.207)
	Kernel QR	1.231(0.106)	2.116(0.324)	0.950(0.060)	1.177(0.134)
	QR Forest	0.752(0.059)	**0.871(0.129)**	0.568(0.026)	0.508(0.044)
0.25	DQRP	0.600(0.047)	0.660(0.112)	**0.374(0.032)**	0.243(0.039)
	DQRP*	**0.517(0.045)**	**0.442(0.082)**	0.383(0.066)	**0.242(0.084)**
	DQR	0.796(0.096)	1.038(0.245)	0.567(0.066)	0.533(0.119)
	Kernel QR	0.686(0.055)	0.771(0.124)	0.503(0.031)	0.397(0.046)
	QR Forest	0.586(0.038)	0.536(0.066)	0.431(0.018)	0.296(0.025)
0.5	DQRP	0.578(0.043)	0.606(0.091)	**0.364(0.038)**	0.226(0.044)
	DQRP*	**0.524(0.052)**	**0.453(0.093)**	0.365(0.066)	**0.219(0.080)**
	DQR	0.749(0.066)	0.914(0.158)	0.539(0.050)	0.483(0.088)
	Kernel QR	0.578(0.039)	0.566(0.074)	0.389(0.018)	0.256(0.024)
	QR Forest	0.553(0.037)	0.480(0.063)	0.404(0.018)	0.260(0.023)
0.75	DQRP	0.628(0.064)	0.718(0.140)	0.410(0.063)	0.289(0.084)
	DQRP*	0.577(0.076)	**0.498(0.138)**	**0.377(0.080)**	**0.231(0.099)**
	DQR	0.799(0.094)	1.034(0.239)	0.558(0.064)	0.520(0.118)
	Kernel QR	0.670(0.051)	0.745(0.106)	0.506(0.035)	0.412(0.055)
	QR Forest	**0.577(0.037)**	0.530(0.069)	0.427(0.019)	0.290(0.025)
0.95	DQRP	0.897(0.152)	1.343(0.381)	0.602(0.149)	0.593(0.264)
	DQRP*	0.873(0.175)	1.121(0.383)	0.600(0.187)	0.539(0.281)
	DQR	0.920(0.140)	1.340(0.389)	0.611(0.090)	0.616(0.174)
	Kernel QR	1.201(0.099)	2.025(0.296)	0.975(0.064)	1.243(0.151)
	QR Forest	**0.761(0.057)**	**0.925(0.132)**	**0.576(0.024)**	**0.523(0.042)**

The simulation results under multivariate "SIM" and "Additive" models are summarized in Tables 7-8 for point estimation and Tables 9-10 for process estimation. For Kernel QR, QR Forest, QR Forest*, DQR, DQR*, our proposed DQRP and DQRP*, the corresponding L_1 and L_2^2 distances (standard deviation in parentheses) between the estimates and the target are reported at different quantile levels. For each column or row, using bold text we highlight the best method which produces the smallest risk among these three methods.

Table 9: Data is generated from the "Single index model" with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The DQR* and QRF* estimators are obtained by interpolating DQR and QRF estimations at quantile level $\tau = 0.05, 0.25, 0.5, 0.75, 0.95$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	$n = 512$				$n = 2048$			
Method	DQRP	DQRP*	DQR*	QR Forest*	DQRP	DQRP*	DQR*	QR Forest*
τ	L_1				L_1			
0.05	0.529(0.026)	**0.466(0.022)**	0.468(0.020)	0.657(0.105)	0.476(0.028)	**0.433(0.015)**	0.452(0.007)	0.687(0.080)
0.1	0.383(0.020)	**0.343(0.014)**	0.403(0.037)	0.582(0.097)	0.347(0.014)	**0.322(0.007)**	0.374(0.016)	0.595(0.078)
0.15	0.299(0.016)	**0.274(0.010)**	0.338(0.028)	0.455(0.071)	0.272(0.009)	**0.258(0.005)**	0.297(0.010)	0.441(0.062)
0.2	0.242(0.014)	**0.225(0.009)**	0.285(0.017)	0.331(0.041)	0.219(0.008)	**0.211(0.005)**	0.243(0.008)	0.303(0.035)
0.25	0.202(0.013)	**0.188(0.010)**	0.251(0.020)	0.242(0.023)	0.179(0.009)	**0.172(0.005)**	0.206(0.009)	0.216(0.015)
0.3	0.173(0.013)	**0.158(0.011)**	0.216(0.020)	0.186(0.014)	0.147(0.010)	**0.138(0.005)**	0.166(0.009)	0.166(0.010)
0.35	0.156(0.015)	**0.133(0.013)**	0.188(0.022)	0.148(0.012)	0.121(0.011)	**0.108(0.005)**	0.129(0.010)	0.130(0.011)
0.4	0.146(0.016)	**0.114(0.015)**	0.165(0.022)	0.120(0.013)	0.103(0.013)	**0.083(0.006)**	0.098(0.010)	0.102(0.015)
0.45	0.143(0.018)	**0.102(0.018)**	0.148(0.022)	0.103(0.016)	0.093(0.016)	**0.065(0.007)**	0.075(0.010)	0.083(0.018)
0.5	0.145(0.019)	0.099(0.020)	0.144(0.022)	**0.097(0.018)**	0.091(0.017)	**0.059(0.008)**	0.068(0.010)	0.075(0.020)
0.55	0.151(0.018)	0.105(0.021)	0.151(0.022)	**0.103(0.017)**	0.098(0.015)	**0.067(0.008)**	0.079(0.009)	0.083(0.018)
0.6	0.164(0.018)	**0.118(0.022)**	0.170(0.023)	0.120(0.016)	0.113(0.013)	**0.085(0.007)**	0.104(0.010)	0.103(0.015)
0.65	0.182(0.017)	**0.137(0.020)**	0.196(0.023)	0.146(0.014)	0.136(0.011)	**0.109(0.005)**	0.136(0.010)	0.131(0.012)
0.7	0.206(0.017)	**0.161(0.018)**	0.229(0.023)	0.182(0.015)	0.163(0.010)	**0.137(0.004)**	0.172(0.010)	0.167(0.012)
0.75	0.236(0.018)	**0.189(0.014)**	0.266(0.023)	0.231(0.021)	0.195(0.010)	**0.170(0.003)**	0.211(0.009)	0.216(0.019)
0.8	0.272(0.018)	**0.223(0.010)**	0.310(0.021)	0.309(0.039)	0.232(0.010)	**0.209(0.003)**	0.253(0.007)	0.296(0.043)
0.85	0.316(0.018)	**0.268(0.009)**	0.357(0.021)	0.429(0.065)	0.277(0.011)	**0.257(0.005)**	0.306(0.009)	0.422(0.075)
0.9	0.372(0.018)	**0.332(0.013)**	0.418(0.022)	0.580(0.093)	0.336(0.015)	**0.325(0.008)**	0.377(0.016)	0.577(0.100)
0.95	0.462(0.020)	**0.447(0.023)**	0.508(0.023)	0.710(0.113)	**0.436(0.025)**	0.444(0.014)	0.466(0.012)	0.702(0.111)
τ	L_2^2				L_2^2			
0.05	0.375(0.040)	**0.284(0.029)**	0.323(0.042)	0.696(0.193)	0.298(0.034)	**0.242(0.016)**	0.314(0.021)	0.749(0.140)
0.1	0.197(0.022)	**0.156(0.013)**	0.265(0.051)	0.527(0.150)	0.158(0.012)	**0.138(0.005)**	0.235(0.025)	0.545(0.115)
0.15	0.123(0.014)	**0.103(0.008)**	0.181(0.034)	0.329(0.092)	0.099(0.007)	**0.092(0.004)**	0.142(0.015)	0.313(0.077)
0.2	0.084(0.010)	**0.073(0.007)**	0.121(0.016)	0.178(0.046)	0.066(0.006)	**0.063(0.004)**	0.082(0.005)	0.152(0.037)
0.25	0.062(0.008)	**0.052(0.006)**	0.095(0.016)	0.094(0.021)	0.046(0.005)	**0.043(0.003)**	0.058(0.005)	0.075(0.014)
0.3	0.048(0.008)	**0.038(0.006)**	0.073(0.014)	0.055(0.010)	0.032(0.005)	**0.028(0.002)**	0.038(0.005)	0.043(0.007)
0.35	0.040(0.008)	**0.028(0.006)**	0.057(0.013)	0.034(0.006)	0.024(0.005)	**0.018(0.002)**	0.024(0.004)	0.026(0.006)
0.4	0.036(0.008)	**0.021(0.005)**	0.045(0.012)	0.023(0.005)	0.018(0.004)	**0.011(0.002)**	0.015(0.003)	0.017(0.005)
0.45	0.034(0.008)	**0.017(0.005)**	0.038(0.011)	0.017(0.005)	0.015(0.004)	**0.007(0.002)**	0.009(0.002)	0.011(0.005)
0.5	0.035(0.008)	0.016(0.006)	0.035(0.010)	**0.015(0.005)**	0.014(0.004)	**0.006(0.002)**	0.008(0.002)	0.009(0.005)
0.55	0.038(0.009)	0.018(0.007)	0.038(0.011)	**0.017(0.006)**	0.016(0.004)	**0.007(0.002)**	0.010(0.002)	0.011(0.005)
0.6	0.045(0.009)	**0.022(0.008)**	0.047(0.013)	0.023(0.006)	0.021(0.004)	**0.011(0.002)**	0.016(0.003)	0.017(0.005)
0.65	0.054(0.010)	**0.030(0.009)**	0.061(0.015)	0.033(0.007)	0.029(0.005)	**0.018(0.002)**	0.027(0.004)	0.026(0.006)
0.7	0.069(0.012)	**0.040(0.010)**	0.081(0.018)	0.051(0.010)	0.041(0.006)	**0.029(0.003)**	0.042(0.004)	0.043(0.008)
0.75	0.089(0.015)	**0.054(0.011)**	0.108(0.020)	0.084(0.019)	0.058(0.007)	**0.042(0.003)**	0.062(0.005)	0.074(0.017)
0.8	0.116(0.019)	**0.074(0.010)**	0.145(0.021)	0.155(0.042)	0.081(0.008)	**0.061(0.002)**	0.092(0.005)	0.143(0.043)
0.85	0.153(0.023)	**0.101(0.008)**	0.193(0.025)	0.297(0.083)	0.113(0.010)	**0.089(0.002)**	0.143(0.014)	0.286(0.089)
0.9	0.207(0.025)	**0.149(0.010)**	0.266(0.033)	0.527(0.142)	0.162(0.012)	**0.138(0.006)**	0.227(0.029)	0.515(0.144)
0.95	0.306(0.027)	**0.260(0.026)**	0.382(0.040)	0.798(0.210)	0.258(0.028)	**0.252(0.016)**	0.338(0.034)	0.773(0.192)

Table 10: Data is generated from the "Additive model" with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The DQR* and QRF* estimators are obtained by interpolating DQR and QRF estimations at quantile level $\tau = 0.05, 0.25, 0.5, 0.75, 0.95$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	$n = 512$				$n = 2048$			
Method	DQRP	DQRP*	DQR*	QR Forest*	DQRP	DQRP*	DQR*	QR Forest*
τ	L_1				L_1			
0.05	0.923(0.132)	0.901(0.209)	**0.69(0.086)**	1.278(0.238)	0.538(0.093)	0.541(0.149)	**0.415(0.060)**	1.213(0.182)
0.1	0.730(0.084)	0.712(0.129)	**0.659(0.053)**	1.233(0.222)	0.423(0.037)	**0.410(0.081)**	0.456(0.083)	1.154(0.179)
0.15	0.655(0.064)	**0.651(0.086)**	0.653(0.049)	1.021(0.182)	0.393(0.025)	**0.389(0.063)**	0.432(0.066)	0.915(0.161)
0.2	**0.618(0.054)**	0.625(0.064)	0.655(0.053)	0.810(0.129)	**0.381(0.029)**	0.386(0.063)	0.397(0.044)	0.680(0.121)
0.25	**0.597(0.047)**	0.613(0.054)	0.652(0.055)	0.682(0.086)	**0.373(0.033)**	0.384(0.064)	0.382(0.038)	0.541(0.080)
0.3	**0.585(0.043)**	0.605(0.050)	0.649(0.053)	0.620(0.061)	**0.367(0.035)**	0.380(0.064)	0.377(0.039)	0.479(0.060)
0.35	**0.578(0.040)**	0.600(0.049)	0.648(0.051)	0.584(0.051)	**0.363(0.036)**	0.375(0.064)	0.375(0.039)	0.445(0.050)
0.4	0.575(0.039)	0.596(0.048)	0.649(0.049)	**0.558(0.047)**	**0.361(0.036)**	0.370(0.063)	0.374(0.037)	0.418(0.041)
0.45	0.574(0.039)	0.593(0.048)	0.648(0.048)	**0.541(0.045)**	**0.361(0.037)**	0.367(0.063)	0.373(0.035)	0.400(0.033)
0.5	0.576(0.042)	0.591(0.048)	0.649(0.046)	**0.535(0.044)**	**0.362(0.037)**	0.366(0.065)	0.373(0.034)	0.393(0.032)
0.55	0.580(0.045)	0.589(0.047)	0.650(0.045)	**0.540(0.049)**	0.367(0.039)	**0.366(0.069)**	0.373(0.034)	0.400(0.041)
0.6	0.586(0.048)	0.587(0.047)	0.653(0.046)	**0.559(0.059)**	0.373(0.043)	**0.368(0.073)**	0.372(0.035)	0.422(0.057)
0.65	0.595(0.053)	**0.585(0.046)**	0.657(0.049)	0.590(0.074)	0.383(0.048)	0.372(0.077)	**0.371(0.035)**	0.458(0.077)
0.7	0.608(0.058)	**0.583(0.046)**	0.663(0.053)	0.641(0.097)	0.395(0.054)	0.375(0.079)	**0.369(0.035)**	0.513(0.104)
0.75	0.624(0.064)	**0.583(0.050)**	0.670(0.058)	0.735(0.130)	0.409(0.061)	0.377(0.080)	**0.368(0.039)**	0.607(0.142)
0.8	0.647(0.070)	**0.587(0.061)**	0.671(0.056)	0.900(0.176)	0.426(0.067)	0.379(0.077)	**0.377(0.052)**	0.778(0.183)
0.85	0.681(0.080)	**0.605(0.087)**	0.667(0.050)	1.128(0.221)	0.447(0.073)	**0.387(0.077)**	0.400(0.070)	1.005(0.209)
0.9	0.740(0.099)	**0.664(0.134)**	0.672(0.055)	1.342(0.254)	0.483(0.087)	0.424(0.102)	**0.414(0.077)**	1.200(0.227)
0.95	0.890(0.152)	0.864(0.209)	**0.720(0.084)**	1.405(0.274)	0.597(0.148)	0.597(0.185)	**0.388(0.050)**	1.228(0.240)
τ	L_2^2				L_2^2			
0.05	1.346(0.312)	1.245(0.471)	**0.777(0.195)**	2.209(0.700)	0.466(0.129)	0.456(0.225)	**0.277(0.077)**	1.872(0.492)
0.1	0.923(0.192)	0.816(0.261)	**0.706(0.112)**	2.041(0.628)	0.310(0.045)	**0.275(0.108)**	0.330(0.109)	1.696(0.459)
0.15	0.770(0.151)	**0.687(0.168)**	0.693(0.102)	1.466(0.459)	0.271(0.034)	**0.249(0.083)**	0.298(0.084)	1.140(0.345)
0.2	0.693(0.128)	**0.636(0.124)**	0.695(0.110)	0.975(0.287)	0.253(0.038)	**0.245(0.081)**	0.254(0.052)	0.683(0.207)
0.25	0.648(0.110)	**0.612(0.104)**	0.689(0.116)	0.711(0.168)	**0.241(0.041)**	0.242(0.081)	0.243(0.046)	0.451(0.116)
0.3	0.621(0.097)	0.598(0.096)	0.683(0.111)	**0.592(0.108)**	**0.232(0.042)**	0.237(0.079)	0.235(0.048)	0.360(0.081)
0.35	0.604(0.089)	0.589(0.093)	0.681(0.108)	**0.528(0.085)**	**0.226(0.043)**	0.230(0.077)	0.227(0.047)	0.312(0.065)
0.4	0.595(0.085)	0.581(0.091)	0.681(0.106)	**0.485(0.077)**	**0.223(0.043)**	0.225(0.075)	0.226(0.045)	0.278(0.050)
0.45	0.593(0.085)	0.576(0.090)	0.681(0.105)	**0.458(0.071)**	0.222(0.043)	**0.221(0.076)**	0.225(0.042)	0.254(0.039)
0.5	0.596(0.088)	0.571(0.089)	0.682(0.101)	**0.449(0.070)**	0.224(0.044)	**0.219(0.078)**	0.225(0.041)	0.246(0.040)
0.55	0.604(0.094)	0.567(0.088)	0.684(0.098)	**0.460(0.078)**	0.229(0.046)	**0.220(0.083)**	0.224(0.040)	0.255(0.055)
0.6	0.619(0.102)	0.563(0.088)	0.689(0.098)	**0.493(0.098)**	0.237(0.051)	**0.222(0.089)**	0.223(0.040)	0.282(0.079)
0.65	0.639(0.113)	0.560(0.087)	0.698(0.103)	**0.551(0.132)**	0.250(0.058)	0.226(0.095)	**0.222(0.040)**	0.330(0.112)
0.7	0.668(0.126)	**0.557(0.089)**	0.710(0.111)	0.647(0.184)	0.266(0.068)	0.229(0.098)	**0.220(0.040)**	0.406(0.158)
0.75	0.706(0.141)	**0.558(0.098)**	0.725(0.121)	0.831(0.269)	0.286(0.080)	0.232(0.098)	**0.220(0.044)**	0.547(0.234)
0.8	0.758(0.158)	**0.567(0.120)**	0.728(0.116)	1.190(0.399)	0.311(0.093)	**0.234(0.093)**	0.235(0.061)	0.835(0.341)
0.85	0.833(0.181)	**0.603(0.168)**	0.721(0.105)	1.756(0.564)	0.344(0.107)	**0.242(0.090)**	0.256(0.085)	1.289(0.458)
0.9	0.967(0.229)	**0.718(0.264)**	0.731(0.116)	2.380(0.732)	0.399(0.136)	0.288(0.124)	**0.273(0.094)**	1.764(0.566)
0.95	1.321(0.383)	1.137(0.466)	**0.835(0.187)**	2.628(0.831)	0.585(0.261)	0.534(0.276)	**0.241(0.057)**	1.879(0.614)

7.4 Distribution of ξ

In this subsection, we investigate how the distribution of ξ affect the performance of the proposed method. We compute our proposed DQRP estimator under the data-generating models and the same configurations as in Section 7.2 and 7.3, but with different choices of distribution of ξ. We set ξ follow Beta distributions $Beta(\alpha, \beta)$ with four different parameters (α, β), i.e. $(\alpha, \beta) = (0.5, 0.5), (1, 1), (2, 2)$ and $(0.5, 1.5)$. The probability density functions and cumulative distribution functions for the four Beta distributions are presented in Figure 6.

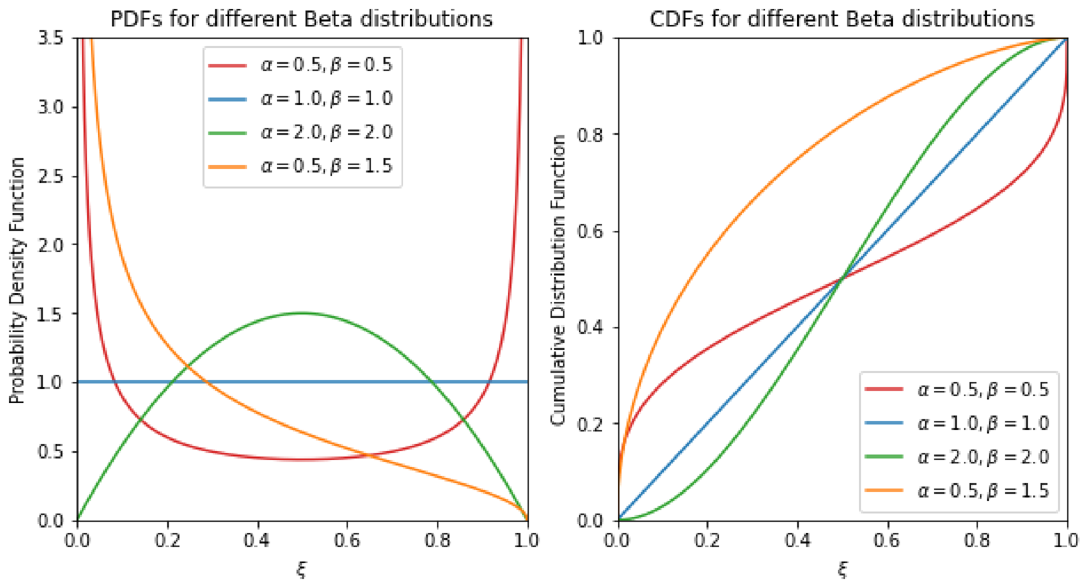


Figure 6: The probability density functions and cumulative distribution functions for Beta distributions $Beta(\alpha, \beta)$ of ξ with different parameters (α, β).

It is known that $Beta(1, 1)$ is a uniform distribution on $(0, 1)$. One can also see from Figure 6 that $Beta(0.5, 0.5)$ concentrates more around 0 and 1; $Beta(2, 2)$ concentrates more near 0.5 and $Beta(0.5, 1.5)$ is right-skewed.

We simulated for univariate models "Linear", "Wave" and "Triangle" with $n = 512$ specified in Section 7.2 to examine how the distribution of ξ affect the estimation. For each model, we replicate for 20 times for each distribution of ξ and report the mean L_1 distance between estimates and the target quantile curves at all levels in $(0, 1)$. The results are visualized in Figures 7-9. There is evidence that the distribution of ξ concentrating around 0 and 1 can improve the estimation at extreme quantiles. And our proposed method performs reasonably well with uniform distributed ξ under different models.

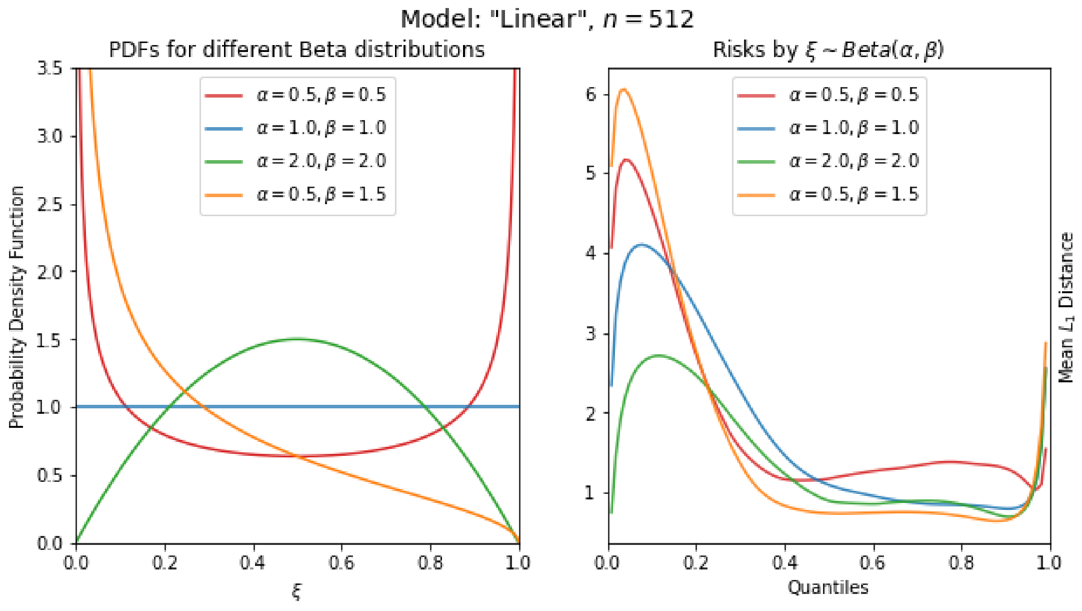


Figure 7: Univariate "Linear" model: The probability density functions of different choices of ξ are depicted in the left. The according mean L_1 distance between estimates and the target quantile curves at all levels in $(0, 1)$ over 20 replications are depicted in the right.

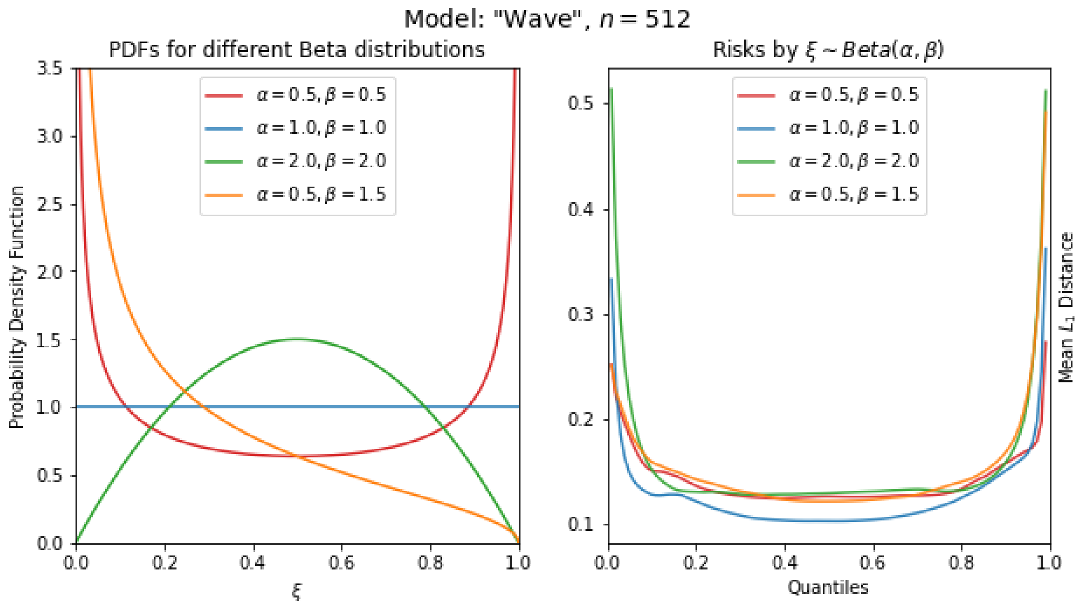


Figure 8: Univariate "Wave" model: The probability density functions of different choices of ξ are depicted in the left. The according mean L_1 distance between estimates and the target quantile curves at all levels in $(0, 1)$ over 20 replications are depicted in the right.

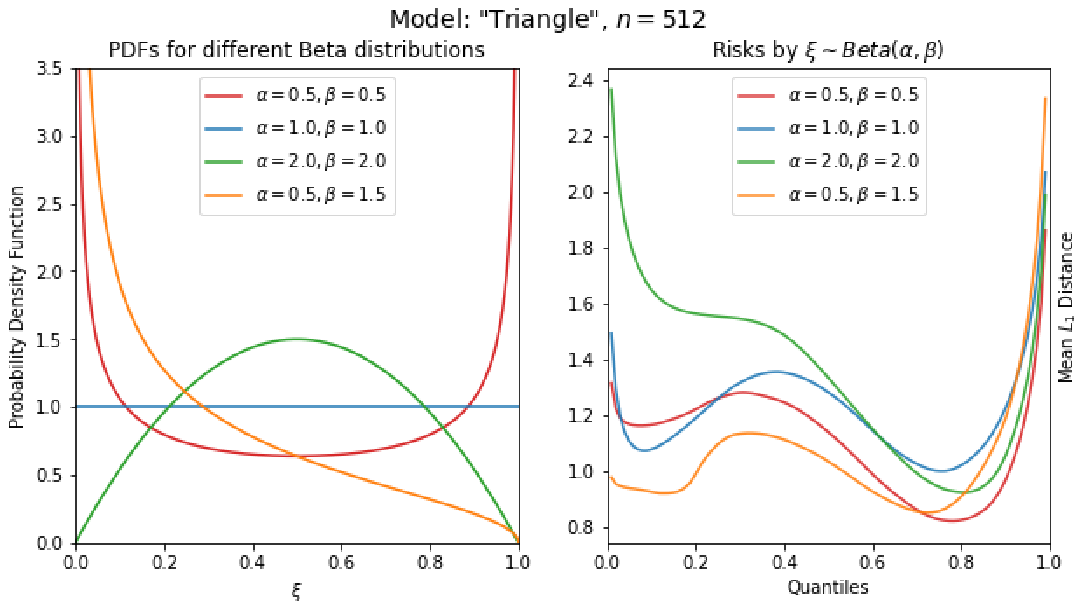


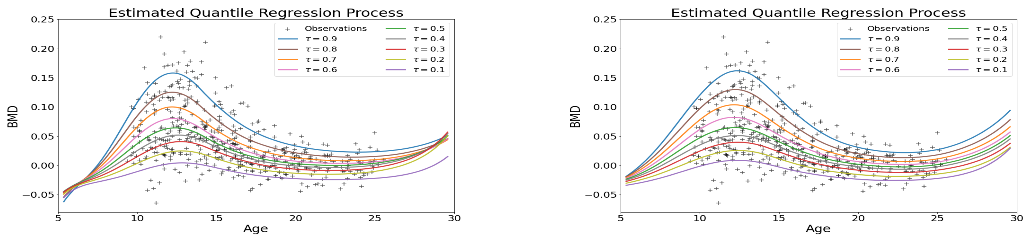
Figure 9: Univariate "Triangle" model: The probability density functions of different choices of ξ are depicted in the left. The according mean L_1 distance between estimates and the target quantile curves at all levels in $(0, 1)$ over 20 replications are depicted in the right.

7.5 Tuning Parameter

In this subsection, we study the effects of the tuning parameter λ on the proposed method. First, we demonstrate that the "quantile crossing" phenomenon can be mitigated. We apply our method to the bone mineral density (BMD) dataset. This dataset is originally reported in Bachrach et al. (1999) and analyzed in Takeuchi et al. (2006); Hastie et al. (2009)[1]. The dataset collects the bone mineral density data of 485 North American adolescents ranging from 9.4 years old to 25.55 years old. Each response value is the difference of the bone mineral density taken on two consecutive visits, divided by the average. The predictor age is the averaged age over the two visits.

In Figure 10, we show the estimated quantile regression processes without ($\lambda = 0$) or with ($\lambda = \log(n)$) the proposed non-crossing penalty. We use the Adam optimizer with the same parameters (for the optimization process) to train a fixed-shape ReQU network with three hidden layers and width $(128, 128, 128)$. The estimated quantile curves at $\tau = 0.1, 0.2, \ldots, 0.9$ and the observations are displayed in Figure 10. It can be seen that the proposed non-crossing penalty is effective to avoid quantile crossing, even in the area outside the range of the training data.

1. The data is also available from the website http://www-stat.stanford.edu/ElemStatlearn.



(a) Without non-crossing penalty (b) With non-crossing penalty

Figure 10: An example of quantile crossing problem in BMD data set. The estimated quantile curves at $\tau = 0.1, 0.2, \ldots, 0.9$ and the observations are depicted. In the left panel, the estimation is conducted without non-crossing penalty and there are crossings at both edges of the graph. In the right figure, the estimation is conducted with non-crossing penalty. There is no quantile crossing even in the area outside the range of the training data.

Second, we study how the value of tuning parameter λ affects the risk of the estimated quantile regression process and how it helps avoiding crossing. Given a sample with size n, we train a series of the DQRP estimators at different values of the tuning parameter λ. For each DQRP estimator, we record its risk and penalty values and the track of these values are plotted in Figures 11-12. For each obtained DQRP estimator $\hat{f}_n^\lambda$, the statistics "Risk" is calculated according to the formula

$$\mathcal{R}(\hat{f}_n^\lambda) = \mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y - f(X, \xi))\},$$

and the statistics "Penalty" is calculated according to

$$\kappa(\hat{f}_n^\lambda) = \mathbb{E}_{X,\xi}[\max\{-\frac{\partial}{\partial\tau}\hat{f}_n^\lambda(X, \xi), 0\}].$$

In practice, we generate $T = 10,000$ testing data $(X_t^{test}, Y_t^{test}, \xi_t^{tesr})_{t=1}^T$ to empirically calculate risks and penalty values.

In each figure, a vertical dashed line is also depicted at the value $\lambda = \log(n)$. It can be seen that crossing seldom happens when we choose a tiny value of the tuning parameter λ. And the loss caused by penalty can be negligible compared to the total risk, since the penalty values are generally of order $O(10^{-3})$ instead of $O(10^3)$ for the total risk. For large value of tuning parameter λ, the crossing nearly disappears which is intuitive and encouraged by the formulation of our penalty. However, the risk could be very large resulting a poor estimation of the target function. As shown by the dashed vertical line in each figure, numerically the choice of $\lambda = \log(n)$ can lead to a reasonable estimation of the target function with tiny risk (blue lines) and little crossing (red lines) across different model settings. Empirically, we choose $\lambda = \log(n)$ in general for the simulations. By Theorem 4, such choice of tuning parameter can lead to a consistent estimator with reasonable fast rate of convergence.

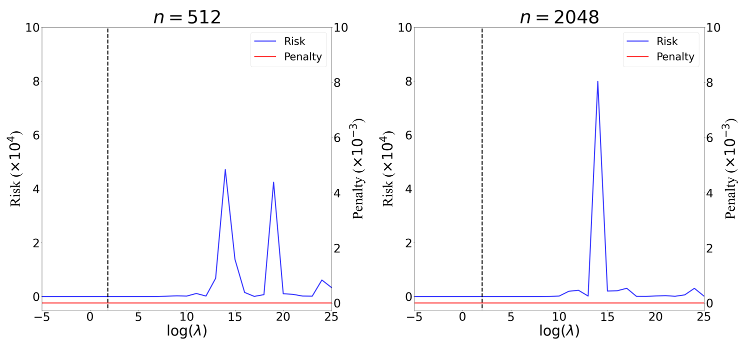


Figure 11: The value of risks and penalties under the univariate "Triangle" model when $n = 512, 2048$. A vertical dashed line is depicted at the value $\lambda = \log(n)$ on x-axis in each figure.

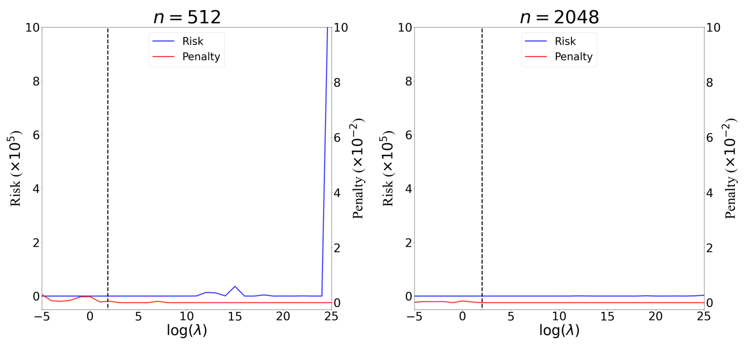


Figure 12: The value of risks and penalties under the multivariate additive model when $n = 512, 2048$ and $d = 8$. A vertical dashed line is depicted at the value $\lambda = \log(n)$ on x-axis in each figure.

8. Conclusions

We have proposed a penalized nonparametric approach to estimating the nonseparable quantile regression model using ReQU activated deep neural networks and introduced a novel penalty function to enforcing non-crossing quantile curves. We have established non-asymptotic excess risk bounds for the estimated QRP and derived the mean integrated squared error for the estimated QRP under mild smoothness and regularity conditions. We have also developed a new approximation error bound for C^s smooth functions with smoothness index $s > 0$ using ReQU activated neural networks. Our numerical experiments demonstrate that the proposed method is competitive with or outperforms two existing methods, including methods using reproducing kernels and random forests, for nonparametric quantile regression. Therefore, the proposed approach can be a useful addition to the methods for multivariate nonparametric regression analysis.

An often used existing approach is to estimate the quantile curve $f_0(\cdot, \tau)$ at each τ nonparametrically and then carry out a post-isotonization step to ensure non-crossing (Chernozhukov et al., 2010; Mammen, 1991; Brando et al., 2022). However, this only works for regression quantiles at finitely many quantile levels, not for the entire regression quantile process on $(0, 1)$. Our method optimizes a novel penalized objective function with a new penalty using differential networks to enforce non-crossing of quantile regression curves. Because we have a well formulated objective function, this enables us to establish non-asymptotic error bounds for the proposed estimator under reasonable conditions.

The results and methods of this work are expected to be useful in other nonparametric estimation problems. In particular, our approximation results on ReQU activated networks are of independent interest. It would be interesting to take advantage of the smoothness of ReQU activated networks and use them in other nonparametric estimation problems, such as the estimation of a regression function and its derivative.

A key assumption in our theoretical analysis is that the neural networks and their derivatives are bounded. The boundedness constraints can be implemented by truncating or clipping the network output and its derivative (Chen et al., 2020; Lee and Kifer, 2021). The commonly used assumptions in the existing works (Schmidt-Hieber, 2020; Zhong and Wang, 2021; Padilla et al., 2022) assume that the network parameters are sparse and the sup-norm or the operator norm of the weight matrices or bias vectors are bounded. These assumptions are stronger than our assumption and may destroy the approximation power of neural network functions (Huster et al., 2018). It would be interesting to relax such assumptions to strengthen the theoretical results. This is an important question that deserves further careful technical analysis in the future.

In applications, the target function f_0 may be neither smooth nor continuous in X, or the conditional distribution of $Y|X$ has point masses. In this case, we may transform the output of neural networks into discrete values to accommodate point mass quantile estimation. To achieve this, one may add additional layers (e.g. Softmax, Softplus or Indicator) at the end of the neural networks. Additional efforts are needed to study theoretical guarantees for quantile process estimation with discrete response under proper assumptions. We leave space here for future research.

Acknowledgements

The authors wish to thank the action editor and reviewers for their valuable and constructive comments, which have significantly improved the quality of this paper. G. Shen is partially supported by the Hong Kong Research Grants Council (Grant No. 15305523) and a research grant from The Hong Kong Polytechnic University. The work of Y. Jiao is supported by the National Nature Science Foundation of China (Grant No.12371441), "the Fundamental Research Funds for the Central Universities", and the research fund of KLATASDSMOE of China. Y. Lin's research was partially supported by the Hong Kong Research Grants Council (Grant No. 14306219, 14306620, 14304523), and Direct Grants for Research, The Chinese University of Hong Kong. The work of J. Huang is supported by the National Natural Science Foundation of China (Grant No. 72331005) and research grants from The Hong Kong Polytechnic University.

Appendix

In the appendix, we include proofs of the results stated in the main text and additional simulation results.

Appendix A. Proof of Theorems, Corollaries and Lemmas

In this section of the Appendix, we include the proofs for the results stated in Section 3 and the technical details needed in the proofs.

Proof of Proposition 1

For any random variable ξ supported on $(0,1)$, the risk

$$\begin{aligned}\mathcal{R}^{\xi}(f) =&\mathbb{E}_{X,Y,\xi}\{\rho_{\xi}(Y-f(X,\xi))\}\\ =&\int_0^1 \mathbb{E}_{X,Y}\{\rho_{\xi}(Y-f(X,\tau))\}\pi_{\xi}(\tau)d\tau\end{aligned}$$

where $\pi_{\xi}(\cdot)\geq 0$ is the density function of ξ. By the definition of f_0 and the property of quantile loss function, it is known f_0 minimizes $\mathbb{E}_{X,Y}\{\rho_{\tau}(Y-f(X,\tau))\}$ as well as $\mathbb{E}_{X,Y}\{\rho_{\tau}(Y-f(X,\tau))\}\pi_{\xi}(\tau)$ for each $\tau\in(0,1)$. Thus f_0 minimizes the integral or the risk $\mathcal{R}(\cdot)$ over measurable functions.

Note that if $\pi_{\xi}(\tau)=0$ for some $\tau\in T$ where T is a subset of $(0,1)$, then any function $\tilde{f}_0$ defined on $\mathcal{X}\times(0,1)$ that is different from f_0 only on $\mathcal{X}\times T$ will also be a minimizer of $\mathcal{R}^{\xi}(\cdot)$. To be exact,

$$\tilde{f}_0\in\arg\min_f \mathcal{R}(f) \qquad \text{if and only if} \qquad \tilde{f}_0=f_0 \text{ on } \mathcal{X}\times(0,1)\backslash T.$$

Further, if (X,ξ) has non zero density almost everywhere on $\mathcal{X}\times(0,1)$ and the probability measure of (X,ξ) is absolutely continuous with respect to Lebesgue measure, then above defined set $\mathcal{X}\times T$ is measure-zero and f_0 is the unique minimizer of $\mathcal{R}(\cdot)$ over all measurable functions in the sense of almost everywhere(almost surely), i.e.,

$$f_0=\arg\min_f \mathcal{R}^{\xi}(f)=\arg\min_f \mathbb{E}_{X,Y,\xi}\{\rho_{\xi}(Y-f(X,\xi))\},$$

up to a negligible set with respect to the probability measure of (X,ξ) on $\mathcal{X}\times(0,1)$. □

Proof of Lemma 1

Recall that $\hat{f}_n^{\lambda}$ is the penalized empirical risk minimizer. Then, for any $f\in\mathcal{F}_n$ we have

$$\mathcal{R}^{\xi}_{n,\lambda}(\hat{f}_n^{\lambda})\leq\mathcal{R}^{\xi}_{n,\lambda}(f).$$

Besides, for any $f\in\mathcal{F}$ we have $\kappa(f)\geq 0$ and $\kappa_n(f)\geq 0$ since κ and κ_n are nonnegative functions. Note that $\kappa(f_0)=\kappa_n(f_0)=0$ by the assumption that f_0 is increasing in its second argument. Then,

$$\mathcal{R}^{\xi}(\hat{f}_n^{\lambda})-\mathcal{R}^{\xi}(f_0)\leq\mathcal{R}^{\xi}(\hat{f}_n^{\lambda})-\mathcal{R}^{\xi}(f_0)+\lambda\{\kappa(\hat{f}_n^{\lambda})-\kappa(f_0)\}=\mathcal{R}^{\xi}_{\lambda}(\hat{f}_n^{\lambda})-\mathcal{R}^{\xi}_{\lambda}(f_0).$$

We can then give upper bounds for the excess risk $\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0)$. For any $f \in \mathcal{F}_n$,

$$\begin{aligned}
&\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0) \le \mathcal{R}_\lambda^\xi(\hat{f}_n^\lambda) - \mathcal{R}_\lambda^\xi(f_0)\\
&= \{\mathcal{R}_\lambda^\xi(\hat{f}_n^\lambda) - \mathcal{R}_{n,\lambda}^\xi(\hat{f}_n^\lambda)\} + \{\mathcal{R}_{n,\lambda}^\xi(\hat{f}_n^\lambda) - \mathcal{R}_{n,\lambda}^\xi(f)\} + \{\mathcal{R}_{n,\lambda}^\xi(f) - \mathcal{R}_\lambda^\xi(f)\} + \{\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)\}\\
&\le 2 \sup_{f\in\mathcal{F}_n} \Big|[\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)] - [\mathcal{R}_{n,\lambda}^\xi(f) - \mathcal{R}_{n,\lambda}^\xi(f_0)]\Big| + \{\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)\},
\end{aligned}$$

where the second inequality holds since $\mathcal{R}_{n,\lambda}^\xi(f) \ge \mathcal{R}_{n,\lambda}^\xi(\hat{f}_n^\lambda)$ for any $f \in \mathcal{F}_n$. Since the inequality holds for any $f \in \mathcal{F}_n$, we have

$$\mathcal{R}^\xi(\hat{f}_n^\lambda) - \mathcal{R}^\xi(f_0) \le 2 \sup_{f\in\mathcal{F}_n} \Big|[\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)] - [\mathcal{R}_{n,\lambda}^\xi(f) - \mathcal{R}_{n,\lambda}^\xi(f_0)]\Big| + \inf_{f\in\mathcal{F}_n}\{\mathcal{R}_\lambda^\xi(f) - \mathcal{R}_\lambda^\xi(f_0)\}.$$

This completes the proof. □

Proof of Theorem 4

The proof is straightforward by consequences of Theorem 14 and Corollary 17.

For any $N \in \mathbb{N}^+$, let $\mathcal{F}_n := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$ be the ReQU activated neural networks $f : \mathcal{X} \times (0,1) \to \mathbb{R}$ with depth $\mathcal{D} \le 2N-1$, width $\mathcal{W} \le 12N^d$, number of neurons $\mathcal{U} \le 15N^{d+1}$, number of parameters $\mathcal{S} \le 24N^{d+1}$ and satisfying $\mathcal{B} \ge \|f_0\|_{C^0}$ and $\mathcal{B}' \ge \|f_0\|_{C^1}$. Then we would compare the stochastic error bounds $8602\mathcal{U}\mathcal{S}$ and $5796\mathcal{D}\mathcal{S}(\mathcal{D}+\log_2 U)$. By simple math it can be shown that $\mathcal{D}\mathcal{S}(\mathcal{D}+\log_2 U) = \mathcal{O}(dN^{d+3})$ and $\mathcal{U}\mathcal{S} = \mathcal{O}(N^{2d+2})$. Since $d \ge 1$, then we choose apply the upper bound $\mathcal{D}\mathcal{S}(\mathcal{D}+\log_2 U)$ in Theorem 14 to get a excess risk bound with lower order in terms of N. This completes the proof. □

Proof of Corollary 5

The proof rests on the smoothness and bound conditions on the target function and the neural networks, as well as the Lipschitz property of the quantile check loss. Firstly, for any $\tau_1, \tau_2 \in (0,1)$ and any $a \in \mathbb{R}$, we have $|\rho_{\tau_1}(a) - \rho_{\tau_2}(a)| \le |a| \cdot |\tau_1 - \tau_2|$. Secondly, for any $t \in (0,1)$ and $a, b \in \mathbb{B}$, it holds that $|\rho_t(a) - \rho_t(b)| \le |a-b|$. For any $\tau \in (0,1)$ and $\delta \in [0,1)$, we consider the risk $\mathcal{R}^t(f) = \mathbb{E}_{X,Y}[\rho_t(Y - f(X,t))]$ for $t \in B_\xi^\tau(\delta)$ and $f \in \mathcal{F}_n$. Note that for any $t \in B_\xi^\tau(\delta)$ and $f \in \mathcal{F}_n$,

$$\begin{aligned}
\mathcal{R}^t(f) =&\mathbb{E}_{X,Y}\left[\rho_t(Y - f(X,t))\right] \ge \mathbb{E}_{X,Y}\left[\rho_\tau(Y - f(X,t))\right] - |Y - f(X,t)| \cdot |\tau - t|\\
\ge&\mathbb{E}_{X,Y}\left[\rho_\tau(Y - f(X,t))\right] - 2\delta\mathcal{B}\\
\ge&\mathbb{E}_{X,Y}\left[\rho_\tau(Y - f(X,\tau))\right] - |f(X,\tau) - f(X,t)| - 2\delta\mathcal{B}\\
\ge&\mathbb{E}_{X,Y}\left[\rho_\tau(Y - f(X,\tau))\right] - \delta\mathcal{B}' - 2\delta\mathcal{B},
\end{aligned}$$

where the second inequality holds since $\|f\|_\infty \le \mathcal{B}$, and the last inequality holds since $\|\frac{\partial}{\partial\tau}f\|_\infty \le \mathcal{B}'$ for all $f \in \mathcal{F}_n$. Similarly, given $\|f_0\|_{C^0} \le \mathcal{B}$ and $\|f_0\|_{C^1} \le \mathcal{B}'$, we have $\mathcal{R}^t(f_0) = \mathbb{E}_{X,Y}\left[\rho_t(Y - f_0(X,t))\right] \le \mathbb{E}_{X,Y}\left[\rho_\tau(Y - f_0(X,\tau))\right] + \delta\mathcal{B}' + 2\delta\mathcal{B}$ for any $t \in B_\xi^\tau(\delta)$.

Then for $\tau \in (0,1)$, $\delta \in [0,1)$ and any $f \in \mathcal{F}_n$,

$$\begin{aligned}
\mathcal{R}^{\xi}(f) - \mathcal{R}^{\xi}(f_0) &= \int_0^1 \left[\mathcal{R}^t(f) - \mathcal{R}^t(f_0)\right] \pi_{\xi}(t)dt \\
&\geq \int_{B_{\xi}^{\tau}(\delta)} \left[\mathcal{R}^t(f) - \mathcal{R}^t(f_0)\right] \pi_{\xi}(t)dt \\
&\geq \int_{B_{\xi}^{\tau}(\delta)} \left[\mathcal{R}^{\tau}(f) - \mathcal{R}^{\tau}(f_0) - 2\delta(\mathcal{B}' + 2\mathcal{B})\right] \pi_{\xi}(t)dt \\
&= \mathbb{P}(\xi \in B_{\xi}^{\tau}(\delta)) \left[\mathcal{R}^{\tau}(f) - \mathcal{R}^{\tau}(f_0) - 2\delta(\mathcal{B}' + 2\mathcal{B})\right],
\end{aligned}$$

where the first inequality holds since $\mathcal{R}^t(f) - \mathcal{R}^t(f_0) \geq 0$ for any f and $t \in (0,1)$. By simple math, we have

$$\mathcal{R}^{\tau}(f) - \mathcal{R}^{\tau}(f_0) \leq \frac{1}{\mathbb{P}(\xi \in B_{\xi}^{\tau}(\delta))} \left[\mathcal{R}^{\xi}(f) - \mathcal{R}^{\xi}(f_0)\right] + 2\delta(\mathcal{B}' + 2\mathcal{B}).$$

Taking expectation on both sides of the above inequality, and combining the upper bound for $\mathbb{E}[\mathcal{R}^{\xi}(f) - \mathcal{R}^{\xi}(f_0)]$, we complete the proof of the first part of Corollary 5. If ξ is uniformly distributed on $(0,1)$, then $\mathbb{P}(\xi \in B_{\xi}^{\tau}(\delta)) = \delta$ for any $\tau \in (0,1)$ and $\delta \in [0,1)$. This gives the uniform guarantee for the pointwise excess risk, and thus completes the proof of Corollary 5.

Proof of Lemma 7

By equation (B.3) in Belloni and Chernozhukov (2011), for any scalar $w, v \in \mathbb{R}$ and $\tau \in (0,1)$ we have

$$\rho_{\tau}(w - v) - \rho_{\tau}(w) = -v\{\tau - I(w \leq 0)\} + \int_0^v \{I(w \leq z) - I(w \leq 0)\}dz.$$

Given any quantile level $\tau \in (0,1)$, function f and $X = x$, let $w = Y - f_0(X,\tau)$, $v = f(X,\tau) - f_0(X,\tau)$. Suppose $|f(x,\tau) - f_0(x,\tau)| \leq K$, taking conditional expectation on above equation with respect to $Y \mid X = x$, we have

$$\begin{aligned}
&\mathbb{E}\{\rho_{\tau}(Y - f(X,\tau)) - \rho_{\tau}(Y - f_0(X,\tau)) \mid X = x\} \\
=&\mathbb{E}\big[-\{f(X,\tau) - f_0(X,\tau)\}\{\tau - I(Y - f_0(X,\tau) \leq 0)\} \mid X = x\big] \\
&+ \mathbb{E}\big[\int_0^{f(X,\tau)-f_0(X,\tau)} \{I(Y - f_0(X,\tau) \leq z) - I(Y - f_0(X,\tau) \leq 0)\}dz \mid X = x\big] \\
=&0 + \mathbb{E}\big[\int_0^{f(X,\tau)-f_0(X,\tau)} \{I(Y - f_0(X,\tau) \leq z) - I(Y - f_0(X,\tau) \leq 0)\}dz \mid X = x\big] \\
=&\int_0^{f(x,\tau)-f_0(x,\tau)} \{P_{Y|X}(f_0(x,\tau) + z) - P_{Y|X}(f_0(x,\tau))\}dz \\
\geq&\int_0^{f(x,\tau)-f_0(x,\tau)} k|z|dz \\
=&\frac{k}{2}|f(x,\tau) - f_0(x,\tau)|^2.
\end{aligned}$$

Suppose $f(x) - f_0(x) > K$, then similarly we have

$$\begin{aligned}
&\mathbb{E}\{\rho_\tau(Y - f(X,\tau)) - \rho_\tau(Y - f_0(X,\tau)) \mid X = x\} \\
=&\int_0^{f(x,\tau)-f_0(x,\tau)} \{P_{Y|X}(f_0(x,\tau)+z) - P_{Y|X}(f_0(x,\tau))\}dz \\
\geq&\int_{K/2}^{f(x,\tau)-f_0(x,\tau)} \{P_{Y|X}(f_0(x,\tau)+K/2) - P_{Y|X}(f_0(x,\tau))\}dz \\
\geq&(f(x,\tau) - f_0(x,\tau) - K/2)(kK/2) \\
\geq&\frac{kK}{4}|f(x,\tau) - f_0(x,\tau)|.
\end{aligned}$$

The case $f(x,\tau) - f_0(x,\tau) \leq -K$ can be handled similarly as in Padilla and Chatterjee (2021). The conclusion follows combining the three different cases and taking expectation with respect to X of above obtained inequality. Finally for any function $f : \mathcal{X} \times (0,1) \to \mathbb{R}$, we have

$$\begin{aligned}
\Delta^2(f, f_0) =&\mathbb{E}\min\{|f(X,\xi) - f_0(X,\xi)|, |f(X,\xi) - f_0(X,\xi)|^2\} \\
\leq&\max\{2/k, 4/(kK)\}\mathbb{E}\Big[\int_0^1 \rho_\tau(Y - f(X,\tau)) - \rho_\tau(Y - f_0(X,\tau))d\tau\Big] \\
=&\max\{2/k, 4/(kK)\}[\mathcal{R}(f) - \mathcal{R}(f_0)].
\end{aligned}$$

Especially, if $\|f\|_\infty \leq \mathcal{B}$, $\|f_0\|_\infty \leq \mathcal{B}$ and $2\mathcal{B} \leq K$, then $\|f - f_0\|_\infty \leq K$, and

$$\mathbb{E}|f(X,\xi) - f_0(X,\xi)|^2 \leq \frac{2}{k}\{\mathcal{R}(f) - \mathcal{R}(f_0)\}.$$

This completes the proof. □

Proof of Theorem 8

Theorem 8 is a direct consequence of Lemma 7 and Theorem 4.

Proof of Theorem 10

The proof is based on the Fano's inequality (Scarlett and Cevher, 2019) and the Varshamov-Gilber bound (Lemma 2.9 in Tsybakov (2008) and Lemma D.2 in Lu et al. (2021b)). The main idea of our proof is to construct a finite subset of the target function space, then turn the problem into a multiple hypothesis testing problem, and lastly apply the Fano's inequality to obtain the lower bound.

Firstly, we specify the space of the target function. Recall that the nonparametric regression model in our paper is a non-separable quantile regression model $Y = f_0(X,U)$, where $Y \in \mathbb{R}$ is the response, $X \in \mathcal{X} \subset \mathbb{R}^d$ is a d-dimensional vector of predictors, U is an unobservable random variable following the uniform distribution on $(0,1)$ and independent of X. The function $f_0 : \mathcal{X}\times(0,1) \to \mathbb{R}$ is an $(d+1)$-dimensional unknown function increasing in its second argument, and it is smooth with order $s \in \mathbb{N}^+$. With a slight abuse of notation, we denote the space of the target f_0 by C^s_+.

Next, we construct a finite subset of the target function space. According to the Varshamov-Gilbert lemma, for any positive integer m satisfying $m^{d+1} \geq 8$, there exists a subset $\mathcal{V} = \{v^{(0)}, \cdots, v^{(2^{m^{d+1}/8})}\}$ of m^{d+1} dimensional hypercube $\{0,1\}^{m^{d+1}}$ such that $v^{(0)} = (0, \ldots, 0)$ and the ℓ_1 distance between every two elements is larger than $m^{d+1}/8$, i.e.,

$$\sum_{i=1}^{m^{d+1}} \|v^{(j)} - v^{(k)}\|_1 \geq \frac{m^{d+1}}{8}, \quad \text{for all } 0 \leq j \neq k \leq 2^{m^{d+1}/8}.$$

According to the proof of Theorem D.1 in Lu et al. (2021b), we consider a simple C^∞ bump function supported on $[0,1]^d \times (0,1)$:

$$g(x,\tau) = \left(\int_{-\infty}^{\tau} h_0(t)dt - C \right) \times \prod_{i=1}^{d} h_1(x_i),$$

where $h_0, h_1 : \mathbb{R} \to \mathbb{R}$ are non-zero integrable function in $C^\infty(\mathbb{R})$ satisfy $h_0(x) > 0$ and $h_1(x) > 0$, and C is a constant satisfying $\int_{-\infty}^{+\infty} h_0(t)dt > C > 0$. It can be verified that $\frac{d}{d\tau} g(x,\tau) > 0$ over its domain $[0,1]^d \times (0,1)$. Then, we construct the multiple hypothesis on the regular grid $(x^{(j)}, \tau^{(j)}), j \in [m]^{d+1}$ by

$$f_k(x,\tau) = \sum_{j \in [m]^{d+1}} v_j^{(k)} \frac{\omega}{m^{s+d+1}} g(m(x - x^{(j)}), m(\tau - \tau^{(j)})), k = 1, 2, \ldots, 2^{m^{d+1}/8},$$

where ω is a constant. It can be checked that $f_k \in C^s$ and

$$\|f_i - f_k\|_{C^1} = \frac{C \cdot \omega}{m^{s+d}} \sum_{j \in [m]^{d+1}} |v_j^{(i)} - v_j^{(k)}|_1 \geq \frac{C \cdot \omega}{m^{s-1}}, \quad \forall 1 \leq i \neq k \leq 2^{m^{d+1}/8}. \tag{24}$$

Now we turn the problem into multiple hypothesis testing. Let $\{1, \ldots, 2^{m^{d+1}/8}\}$ be an index set, and let P_v denote the distribution of the data (X, Y) corresponding to the generating model $Y = f_v(X, U)$ for $v \in \{1, \ldots, 2^{m^{d+1}/8}\}$. Firstly, suppose that there is a sample $S_n = \{(X_i, Y_i)\}_{i=1}^n$ generated from the model $Y = f(X, U)$ for a given $f \in C^s$. For any estimator $\hat{f}_n$ based on S_n, and any $\varepsilon > 0$, we know that

$$\begin{aligned}\sup_{f \in C_+^s} \mathbb{E}\|\hat{f}_n - f\|_{C^1} &\geq \sup_{f \in C^s} \varepsilon \times \mathbb{P}(\|\hat{f}_n - f\|_{C^1} \geq \varepsilon) \\ &\geq \varepsilon \times \max_{v=1,\ldots,2^{m^{d+1}/8}} \mathbb{P}(\|\hat{f}_n - f_v\|_{C^1} \geq \varepsilon),\end{aligned}$$

where the first inequality follows from Markov's inequality and the second inequality follows by finding the maximum over a smaller set.

Next, we consider the multiple hypothesis testing over the set $\mathcal{I}_V = \{1, \ldots, 2^{m^{d+1}/8}\}$. Suppose that a random index V is drawn uniformly from $\mathcal{I}_V$, then the samples $S_n = \{(X_i, Y_i)\}_{i=1}^n$ are drawn from the distribution P_v corresponding to $Y = f_v(X, U)$ for $v = V$. Given any estimator $\hat{f}_n$ based on S_n, let $f_{\hat{V}}$ denote the one closest to $\hat{f}_n$ with respect to the metric $\|\cdot\|_{C^1}$ over $\{f_v : v \in \mathcal{I}_V\}$, i.e., $\hat{V} = \text{argmin}_{v \in \mathcal{I}_V} \|f_v - \hat{f}_n\|_{C^1}$. Using the triangle inequality and (24), for $\varepsilon = C \cdot \omega / m^{[t]-d-2}$ if $\|f_v - \hat{f}_n\|_{C^1} < \varepsilon/2$ then $\hat{V} = v$. Thus

$$\mathbb{P}(\|f_v - \hat{f}_n\|_{C^1} \geq \varepsilon/2) \geq \mathbb{P}(\hat{V} \neq v),$$

and for $\varepsilon = C \cdot \omega / m^{s-1}$ and any estimator $\hat{f}_n$,

$$
\begin{aligned}
\sup_{f \in C^s_+} \mathbb{E}\|\hat{f}_n - f\|_{C^1} \geq& \sup_{f \in C^s} \varepsilon \times \mathbb{P}(\|\hat{f}_n - f\|_{C^1} \geq \varepsilon) \\
&\geq \varepsilon \times \max_{v=1,\ldots,2^{m^{d+1}/8}} \mathbb{P}(\|\hat{f}_n - f_v\|_{C^1} \geq \varepsilon) \\
&\geq \varepsilon \times \max_{v=1,\ldots,2^{m^{d+1}/8}} \mathbb{P}(\hat{V} \neq v) \\
&\geq \varepsilon \times \frac{1}{2^{m^{d+1}/8}} \sum_{v=1}^{2^{m^{d+1}/8}} \mathbb{P}(\hat{V} \neq v) \\
&\geq \varepsilon \times \left(1 - \frac{I(V; S_n) + \log 2}{log(2^{m^{d+1}/8})}\right) \\
&= \frac{C \cdot \omega}{m^{s-1}} \times \left(1 - \frac{I(V; S_n) + \log 2}{m^{d+1} \log(2)/8}\right),
\end{aligned}
$$

where the second last inequality lower bounds the maximum by the average, and the last inequality follows from Fano's inequality (Theorem 1 in Scarlett and Cevher (2019)) and $I(V; S_n)$ denotes the mutual information of V and S_n. To calculate the lower bound, in what follows, we give bounds of the mutual information $I(V; S_n)$. Note that $S_n = \{(X_i, Y_i)\}_{i=1}^n$ contains n independent samples, then we can tensorize the mutual information $I(V; S_n)$ by a sum of n mutual information terms. By Lemma 2 in Scarlett and Cevher (2019), we know

$$
I(V; S_n) \leq \sum_{i=1}^n I(V; (X_i, Y_i)) = n \times I(V; (X, Y)).
$$

By the KL divergence based bounds (Lemma 4 in Scarlett and Cevher (2019)), we further have

$$
I(V; (X, Y)) \leq \max_{v, v' \in \{1, \ldots, 2^{m^{d+1}/8}\}} D_{\mathrm{KL}}(P_v \mid\mid P_{v'}),
$$

where $D_{\mathrm{KL}}(P_v \mid\mid P_{v'})$ denotes the KL distance between distributions P_v and $P_{v'}$, and P_v denotes the joint distribution of (X, Y) under the model $Y = f_v(X, U)$. For simplicity, we let p_v and $p_{v'}$ denote the density function of the joint distribution P_v and $P_{v'}$. It follows from the definition of KL divergence that

$$\begin{aligned}
D_{\mathrm{KL}}(P_v \mid\mid P_{v'}) &\leq D_{\chi^2}(P_v \mid\mid P_{v'}) \\
&= \mathbb{E}_{(X,Y)\sim P_{v'}} \left[\frac{\{p_v(X,Y) - p_{v'}(X,Y)\}^2}{p_{v'}^2(X,Y)} \right] \\
&= \mathbb{E}_{(X,Y)\sim P_{v'}} \left[\frac{\{p_v(Y \mid X) - p_{v'}(Y \mid X)\}^2}{p_{v'}^2(Y \mid X)} \right] \\
&\leq \frac{1}{k^2} \mathbb{E}_{(X,Y)\sim P_{v'}} \left| p_v(Y \mid X) - p_{v'}(Y \mid X) \right|^2 \\
&\leq \frac{C_1}{k^2} \mathbb{E}_{(X,Y)\sim P_v} \left| (f_v(X,\cdot)^{-1})'(Y) - (f_{v'}(X,\cdot)^{-1})'(Y) \right|^2 \\
&\leq \frac{2C_1}{k^2} \mathbb{E}_{(X,Y)\sim P_v} \Bigg[\left| (f_{v'}(X,\cdot))'(f_{v'}(X,\cdot)^{-1}(Y)) - (f_v(X,\cdot))'(f_{v'}(X,\cdot)^{-1}(Y)) \right|^2 \\
&\quad + \left| (f_v(X,\cdot))'(f_{v'}(X,\cdot)^{-1}(Y)) - [f_v(X,\cdot)]'(f_v(X,\cdot)^{-1}(Y)) \right|^2 \Bigg] \\
&\leq \frac{2C_1}{k^2} \times \frac{C_2 \cdot \omega}{m^{2s}} \leq \frac{C3}{m^{2s}},
\end{aligned}$$

where the first inequality holds due to the relationship between KL and χ^2 divergence (see, e.g., Lemma 6 of Scarlett and Cevher 2019), the second inequality follows from Assumption 6 where the conditional density function of Y given X is lower bounded by k, and the third inequality follows by the inverse function rule.

Combining the obtained bounds, for any estimator $\hat{f}_n$ based on the sample $S_n = \{(X_i, Y_i)\}_{i=1}^n$, we have

$$\sup_{f \in C_+^s} \mathbb{E}\|\hat{f}_n - f\|_{C^1} \geq \frac{C \cdot \omega}{m^{s-1}} \times \left(1 - \frac{C_4 \cdot \omega \cdot n}{m^{d+2s+1}}\right).$$

Then by choosing $m = \lfloor n^{1/(d+2s+1)} \rfloor$ and proper ω, we finally obtain

$$\inf_{\hat{f}_n} \sup_{f \in C_+^s} \mathbb{E}\|\hat{f}_n - f\|_{C^1} \geq C_5 \times n^{-(s-1)/(d+2s+1)}.$$

Similarly, we can obtain

$$\inf_{\hat{f}_n} \sup_{f \in C_+^s} \mathbb{E}\|\hat{f}_n - f\|_{C^0}^2 \geq C_6 \times n^{-2s/(d+2s+1)}.$$

□

Proof of Theorem 11

The proof is similar to that of Theorem 10 and we omit the details here.

Proof of Lemma 12

Let $\sigma_1(x) = \max\{0, x\}$ and $\sigma_2(x) = \max\{0, x\}^2$ denote the ReLU and ReQU activation functions respectively. Let $(d_0, d_1, \dots, d_{\mathcal{D}+1})$ be vector of the width (number of neurons) of each layer in the original ReQU network where $d_0 = d + 1$ and $d_{\mathcal{D}+1} = 1$ in our problem. We let $f_j^{(i)}$ be the function (subnetwork of the ReQU network) from $\mathcal{X} \times (0, 1) \subset \mathbb{R}^{d+1}$ to $\mathbb{R}$ which takes $(X, \xi) = (x_1, \dots, x_d, x_{d+1})$ as input and outputs the j-th neuron of the i-th layer for $j = 1, \dots, d_i$ and $i = 1, \dots, \mathcal{D} + 1$.

We next construct iteratively ReLU-ReQU activated subnetworks to compute $(\frac{\partial}{\partial\tau} f_1^{(i)}, \dots, f_{d_i}^{(i)})$ for $i = 1, \dots, \mathcal{D} + 1$, i.e., the partial derivatives of the original ReQU subnetworks step by step. We illustrate the details of the construction of the ReLU-ReQU subnetworks for the first two layers ($i = 1, 2$) and the last layer ($\imath = \mathcal{D} + 1$) and apply induction for layers $i = 3, \dots, \mathcal{D}$. Note that the derivative of ReQU activation function is $\sigma_2'(x) = 2\sigma_1(x)$, then when $i = 1$ for any $j = 1, \dots, d_1$,

$$\frac{\partial}{\partial\tau} f_j^{(1)} = \frac{\partial}{\partial\tau}\sigma_2\Big(\sum_{i=1}^{d+1} w_{ji}^{(1)} x_i + b_j^{(1)}\Big) = 2\sigma_1\Big(\sum_{i=1}^{d+1} w_{ji}^{(1)} x_i + b_j^{(1)}\Big) \cdot w_{j,d+1}^{(1)}, \tag{25}$$

where we denote $w_{ji}^{(1)}$ and $b_j^{(1)}$ by the corresponding weights and bias in 1-th layer of the original ReQU network and with a little bit abuse of notation we view x_{d+1} as the argument τ and calculate its partial derivative. Now we intend to construct a 4 layer (2 hidden layers) ReLU-ReQU network with width $(d_0, 3d_1, 10d_1, 2d_1)$ which takes $(X, \xi) = (x_1, \dots, x_d, x_{d+1})$ as input and outputs

$$(f_1^{(1)}, \dots, f_{d_1}^{(1)}, \frac{\partial}{\partial\tau} f_1^{(1)}, \dots, \frac{\partial}{\partial\tau} f_{d_1}^{(1)}) \in \mathbb{R}^{2d_1}.$$

Note that the output of such network contains all the quantities needed to calculated $(\frac{\partial}{\partial\tau} f_1^{(2)}, \dots, \frac{\partial}{\partial\tau} f_{d_2}^{(2)})$, and the process of construction can be continued iteratively and the induction proceeds. In the firstly hidden layer, we can obtain $3d_1$ neurons

$$(f_1^{(1)}, \dots, f_{d_1}^{(1)}, |w_{1,d_0}^{(1)}|, \dots, |w_{d_1,d_0}^{(1)}|, \sigma_1(\sum_{i=1}^{d_0} w_{1i}^{(1)} x_i + b_1^{(1)}), \dots, \sigma_1(\sum_{i=1}^{d_0} w_{d_1 i}^{(1)} x_i + b_{d_1}^{(1)})),$$

with weight matrix $A_1^{(1)}$ having $2d_0d_1$ parameters, bias vector $B_1^{(1)}$ and activation function vector Σ_1 being

$$A_1^{(1)} = \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & \cdots & \cdots & w_{1,d_0}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & \cdots & \cdots & w_{2,d_0}^{(1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{d_1,1}^{(1)} & w_{d_1,2}^{(1)} & \cdots & \cdots & w_{d_1,d_0}^{(1)} \\ 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 \\ w_{1,1}^{(1)} & w_{1,2}^{(1)} & \cdots & \cdots & w_{1,d_0}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & \cdots & \cdots & w_{2,d_0}^{(1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{d_1,1}^{(1)} & w_{d_1,2}^{(1)} & \cdots & \cdots & w_{d_1,d_0}^{(1)} \end{bmatrix} \in \mathbb{R}^{3d_1 \times d_0},$$

$$B_1^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \cdots \\ b_{d_1}^{(1)} \\ |w_{1,d_0}^{(1)}| \\ |w_{2,d_0}^{(1)}| \\ \cdots \\ |w_{d_1,d_0}^{(1)}| \\ b_1^{(1)} \\ b_2^{(1)} \\ \cdots \\ b_{d_1}^{(1)} \end{bmatrix} \in \mathbb{R}^{3d_1}, \quad \Sigma_1^{(1)} = \begin{bmatrix} \sigma_2 \\ \cdots \\ \sigma_2 \\ \sigma_1 \\ \cdots \\ \sigma_1 \\ \sigma_1 \\ \cdots \\ \sigma_1 \end{bmatrix},$$

where the first d_1 activation functions of Σ_1 are chosen to be σ_2 and others σ_1. In the second hidden layer, we can obtain $10d_1$ neurons. The first $2d_1$ neurons of the second hidden layer (or the third layer) are

$$(\sigma_1(f_1^{(1)}), \sigma_1(-f_1^{(1)})), \ldots, \sigma_1(f_{d_1}^{(1)}), \sigma_1(f_{d_1}^{(1)})),$$

which intends to implement identity map such that $(f_1^{(1)}, \ldots, f_{d_1}^{(1)})$ can be kept and outputted in the next layer since identity map can be realized by $x = \sigma_1(x) - \sigma_1(-x)$. The first $8d_1$ neurons of the second hidden layer (or the third layer) are

$$\begin{bmatrix} \sigma_2(w^{(1)}_{1,d_0} + \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{1i} x_i + b^{(1)}_1)) \\ \sigma_2(w^{(1)}_{1,d_0} - \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{1i} x_i + b^{(1)}_1)) \\ \sigma_2(-w^{(1)}_{1,d_0} + \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{1i} x_i + b^{(1)}_1) \\ \sigma_2(-w^{(1)}_{1,d_0} - \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{1i} x_i + b^{(1)}_1)) \\ \cdots \\ \sigma_2(w^{(1)}_{d_1,d_0} + \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{d_1 i} x_i + b^{(1)}_{d_1})) \\ \sigma_2(w^{(1)}_{d_1,d_0} - \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{d_1 i} x_i + b^{(1)}_{d_1})) \\ \sigma_2(-w^{(1)}_{d_1,d_0} + \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{d_1 i} x_i + b^{(1)}_{d_1}) \\ \sigma_2(-w^{(1)}_{d_1,d_0} - \sigma_1(\sum_{i=1}^{d_0} w^{(1)}_{d_1 i} x_i + b^{(1)}_{d_1})) \end{bmatrix} \in \mathbb{R}^{8d_1},$$

which is ready for implementing the multiplications in (25) to obtain $(\frac{\partial}{\partial \tau} f^{(1)}_1, \ldots, \frac{\partial}{\partial \tau} f^{(1)}_{d_1}) \in \mathbb{R}^{d_1}$ since

$$x \cdot y = \frac{1}{4}\{(x+y)^2 - (x-y)^2\} = \frac{1}{4}\{\sigma_2(x+y) + \sigma_2(-x-y) - \sigma_2(x-y) - \sigma_2(-x+y)\}.$$

In the second hidden layer (the third layer), the bias vector is zero $B^{(1)}_2 = (0, \ldots, 0) \in \mathbb{R}^{10d_1}$, activation functions vector

$$\Sigma^{(1)}_2 = (\underbrace{\sigma_1, \ldots, \sigma_1}_{2d_1 \text{ times}}, \underbrace{\sigma_2, \ldots, \sigma_2}_{8d_1 \text{ times}}),$$

and the corresponding weight matrix $A^{(1)}_2$ can be formulated correspondingly without difficulty which contains $2d_1 + 8d_1 = 10d_1$ non-zero parameters. Then in the last layer, by the identity maps and multiplication operations with weight matrix $A^{(1)}_3$ having $2d_1 + 4d_1 = 6d_1$ parameters, bias vector $B^{(1)}_3$ being zeros, we obtain

$$(f^{(1)}_1, \ldots, f^{(1)}_{d_1}, \frac{\partial}{\partial \tau} f^{(1)}_1, \ldots, \frac{\partial}{\partial \tau} f^{(1)}_{d_1}) \in \mathbb{R}^{2d_1}.$$

Such ReLU-ReQU neural network has 2 hidden layers (4 layers), $15d_1$ hidden neurons, $2d_0d_1 + 3d_1 + 10d_1 + 6d_1 = 2d_0d_1 + 19d_1$ parameters and its width is $(d_0, 3d_1, 10d_1, 2d_1)$. It worth noting that the ReLU-ReQU activation functions do not apply to the last layer since the construction here is for a single network. When we are combining two consecutive subnetworks into one long neural network, the ReLU-ReQU activation functions should apply to the last layer of the first subnetwork. Hence, in the construction of the whole big network, the last layer of the subnetwork here should output $4d_1$ neurons

$$\begin{aligned} &(\sigma_1(f^{(1)}_1), \sigma_1(-f^{(1)}_1) \ldots, \sigma_1(f^{(1)}_{d_1}), \sigma_1(-f^{(1)}_{d_1}), \\ &\quad \sigma_1(\frac{\partial}{\partial \tau} f^{(1)}_1), \sigma_1(-\frac{\partial}{\partial \tau} f^{(1)}_1) \ldots, \sigma_1(\frac{\partial}{\partial \tau} f^{(1)}_{d_1}), \sigma_1(-\frac{\partial}{\partial \tau} f^{(1)}_{d_1})) \in \mathbb{R}^{4d_1}, \end{aligned}$$

to keep $(f^{(1)}_1, \ldots, f^{(1)}_{d_1}, \frac{\partial}{\partial \tau} f^{(1)}_1, \ldots, \frac{\partial}{\partial \tau} f^{(1)}_{d_1})$ in use in the next subnetwork. Then for this ReLU-ReQU neural network, the weight matrix $A^{(1)}_3$ has $2d_1 + 8d_1 = 10d_1$ parameters, the bias vector $B_3^{(1)}$ is zeros and the activation functions vector $\Sigma_3^{(1)}$ has all σ_1 as elements. And such ReLU-ReQU neural network has 2 hidden layers (4 layers), $17d_1$ hidden neurons, $2d_0d_1 + 3d_1 + 10d_1 + 10d_1 = 2d_0d_1 + 23d_1$ parameters and its width is $(d_0, 3d_1, 10d_1, 4d_1)$.

Now we consider the second step, for any $j = 1, \ldots, d_2$,

$$\frac{\partial}{\partial \tau} f_j^{(2)} = \frac{\partial}{\partial \tau}\sigma_2\Big(\sum_{i=1}^{d_1} w_{ji}^{(2)} f_i^{(1)} + b_j^{(2)}\Big) = 2\sigma_1\Big(\sum_{i=1}^{d_1} w_{ji}^{(2)} f_i^{(1)} + b_j^{(2)}\Big) \cdot \sum_{i=1}^{d_1} w_{j,i}^{(2)} \frac{\partial}{\partial \tau} f_i^{(1)}, \qquad (26)$$

where $w_{ji}^{(2)}$ and $b_j^{2)}$ are defined correspondingly as the weights and bias in 2-th layer of the original ReQU network. By the previous constructed subnetwork, we can start with its outputs

$$(\sigma_1(f_1^{(1)}), \sigma_1(-f_1^{(1)}) \ldots, \sigma_1(f_{d_1}^{(1)}), \sigma_1(-f_{d_1}^{(1)}),$$
$$\sigma_1(\frac{\partial}{\partial \tau} f_1^{(1)}), \sigma_1(-\frac{\partial}{\partial \tau} f_1^{(1)}) \ldots, \sigma_1(\frac{\partial}{\partial \tau} f_{d_1}^{(1)}), \sigma_1(-\frac{\partial}{\partial \tau} f_{d_1}^{(1)})) \in \mathbb{R}^{4d_1},$$

as the inputs of the second subnetwork we are going to build. In the firstly hidden layer of the second subnetwork, we can obtain $3d_2$ neurons

$$\Big(f_1^{(2)}, \ldots, f_{d_2}^{(2)}, |\sum_{i=1}^{d_1} w_{1,i}^{(2)} \frac{\partial}{\partial \tau} f_i^{(1)}|, \ldots, |\sum_{i=1}^{d_1} w_{d_2,i}^{(2)} \frac{\partial}{\partial \tau} f_i^{(1)}|,$$
$$\sigma_1(\sum_{i=1}^{d_1} w_{1i}^{(2)} f_i^{(1)} + b_1^{(1)}), \ldots, \sigma_1(\sum_{i=1}^{d_1} w_{d_2 i}^{(2)} f_i^{(1)} + b_{d_2}^{(2)})\Big),$$

with weight matrix $A_1^{(2)} \in \mathbb{R}^{4d_1 \times 3d_2}$ having $6d_1d_2$ non-zero parameters, bias vector $B_1^{(2)} \in \mathbb{R}^{3d_2}$ and activation functions vector $\Sigma_1^{(2)} = \Sigma_1^{(1)}$. Similarly, the second hidden layer can be constructed to have $10d_2$ neurons with weight matrix $A_2^{(2)} \in \mathbb{R}^{3d_2 \times 10d_2}$ having $2d_2 + 8d_2 = 10d_2$ non-zero parameters, zero bias vector $B_1^{(2)} \in \mathbb{R}^{10d_2}$ and activation functions vector $\Sigma_2^{(2)} = \Sigma_2^{(1)}$. The second hidden layer here serves exactly the same as that in the first subnetwork, which intends to implement the identity map for

$$(f_1^{(2)}, \ldots, f_{d_2}^{(2)}),$$

and implement the multiplication in (26). Similarly, the last layer can also be constructed as that in the first subnetwork, which outputs

$$(\sigma_1(f_1^{(2)}), \sigma_1(-f_1^{(2)}) \ldots, \sigma_1(f_{d_2}^{(2)}), \sigma_1(-f_{d_2}^{(2)}),$$
$$\sigma_1(\frac{\partial}{\partial \tau} f_1^{(2)}), \sigma_1(-\frac{\partial}{\partial \tau} f_1^{(2)}) \ldots, \sigma_1(\frac{\partial}{\partial \tau} f_{d_2}^{(2)}), \sigma_1(-\frac{\partial}{\partial \tau} f_{d_2}^{(2)})) \in \mathbb{R}^{4d_2},$$

with the weight matrix $A_3^{(2)}$ having $2d_2 + 8d_2 = 10d_2$ parameters, the bias vector $B_3^{(2)}$ being zeros and the activation functions vector $\Sigma_3^{(1)}$ with elements being σ_1. Then the second ReLU-ReQU subnetwork has 2 hidden layers (4 layers), $17d_2$ hidden neurons, $6d_1d_2 + 3d_2 + 10d_2 + 10d_2 = 6d_1d_2 + 23d_2$ parameters and its width is $(4d_1, 3d_2, 10d_2, 4d_2)$.

Then we can continuing this process of construction. For integers $k = 3, \dots, \mathcal{D}$ and for any $j = 1, \dots, d_k$,

$$\begin{aligned}\frac{\partial}{\partial \tau} f_j^{(k)} &= \frac{\partial}{\partial \tau}\sigma_2\Big(\sum_{i=1}^{d_{k-1}} w_{ji}^{(k)} f_i^{(k-1)} + b_j^{(k)}\Big)\\ &= 2\sigma_1\Big(\sum_{i=1}^{d_{k-1}} w_{ji}^{(k)} f_i^{(k-1)} + b_j^{(k)}\Big)\cdot\sum_{i=1}^{d_{k-1}} w_{j,i}^{(k)}\frac{\partial}{\partial \tau} f_i^{(k-1)},\end{aligned}$$

where $w_{ji}^{(k)}$ and $b_j^{(k)}$ are defined correspondingly as the weights and bias in k-th layer of the original ReQU network. We can construct a ReLU-ReQU network taking

$$\begin{aligned}&(\sigma_1(f_1^{(k-1)}), \sigma_1(-f_1^{(k-1)}) \dots, \sigma_1(f_{d_{k-1}}^{(k-1)}), \sigma_1(-f_{d_{k-1}}^{(k-1)}),\\ &\quad\sigma_1(\frac{\partial}{\partial \tau} f_1^{(k-1)}), \sigma_1(-\frac{\partial}{\partial \tau} f_1^{(k-1)}) \dots, \sigma_1(\frac{\partial}{\partial \tau} f_{d_{k-1}}^{(k-1)}), \sigma_1(-\frac{\partial}{\partial \tau} f_{d_{k-1}}^{(k-1)})) \in \mathbb{R}^{4d_{k-1}},\end{aligned}$$

as input, and it outputs

$$\begin{aligned}&(\sigma_1(f_1^{(k)}), \sigma_1(-f_1^{(k)}) \dots, \sigma_1(f_{d_k}^{(k)}), \sigma_1(-f_{d_k}^{(k)}),\\ &\quad\sigma_1(\frac{\partial}{\partial \tau} f_1^{(k)}), \sigma_1(-\frac{\partial}{\partial \tau} f_1^{(k)}) \dots, \sigma_1(\frac{\partial}{\partial \tau} f_{d_k}^{(k)}), \sigma_1(-\frac{\partial}{\partial \tau} f_{d_k}^{(k)})) \in \mathbb{R}^{4d_k},\end{aligned}$$

with 2 hidden layers, $17d_k$ hidden neurons, $6d_{k-1}d_k + 23d_k$ parameters and its width is $(4d_{k-1}, 3d_k, 10d_k, 4d_K)$.

Iterate this process until the $k = \mathcal{D} + 1$ step, where the last layer of the original ReQU network has only 1 neurons. That is for the ReQU activated neural network $f \in \mathcal{F}_n = \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B}}$, the output of the network $f : \mathcal{X} \times (0, 1) \to \mathbb{R}$ is a scalar and the partial derivative with respect to τ is

$$\frac{\partial}{\partial \tau} f = \frac{\partial}{\partial \tau}\sum_{i=1}^{d_{\mathcal{D}+1}} w_i^{(\mathcal{D})} f_i^{(\mathcal{D})} + b^{(\mathcal{D})} = \sum_{i=1}^{d_{\mathcal{D}+1}} w_i^{(\mathcal{D})}\frac{\partial}{\partial \tau} f_i^{(\mathcal{D})},$$

where $w_i^{(\mathcal{D})}$ and $b^{(\mathcal{D})}$ are the weights and bias parameter in the last layer of the ReQU network. The the constructed $\mathcal{D} + 1$-th subnetwork taking

$$\begin{aligned}&(\sigma_1(f_1^{(\mathcal{D})}), \sigma_1(-f_1^{(\mathcal{D})}) \dots, \sigma_1(f_{d_{\mathcal{D}}}^{(\mathcal{D})}), \sigma_1(-f_{d_{\mathcal{D}}}^{(\mathcal{D})}),\\ &\quad\sigma_1(\frac{\partial}{\partial \tau} f_1^{(\mathcal{D})}), \sigma_1(-\frac{\partial}{\partial \tau} f_1^{(\mathcal{D})}) \dots, \sigma_1(\frac{\partial}{\partial \tau} f_{d_{\mathcal{D}}}^{(\mathcal{D})}), \sigma_1(-\frac{\partial}{\partial \tau} f_{d_{\mathcal{D}}}^{(\mathcal{D})})) \in \mathbb{R}^{4d_{\mathcal{D}}},\end{aligned}$$

as input and it outputs $\frac{\partial}{\partial \tau} f^{(\mathcal{D}+1)} = \frac{\partial}{\partial \tau} f$ which is the partial derivative of the whole ReQU network with respect to its last argument τ or $x_{d_0} = x_{d+1}$ here. The subnetwork should have 2 hidden layers width $(4d_{\mathcal{D}}, 2, 8, 1)$ with 11 hidden neurons, $4d_{\mathcal{D}} + 2 + 16 = 4d_{\mathcal{D}} + 18$ non-zero parameters.

Lastly, we combing all the $\mathcal{D} + 1$ subnetworks in order to form a big ReLU-ReQU network which takes $(X, \xi) = (x_1, \dots, x_{d+1}) \in \mathbb{R}^{d+1}$ as input and outputs $\frac{\partial}{\partial \tau} f$ for $f \in \mathcal{F}_n = \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B},\mathcal{B}'}$. Recall that here $\mathcal{D}, \mathcal{W}, \mathcal{U}, \mathcal{S}$ are the depth, width, number of neurons and number of parameters of the ReQU network respectively, and we have $\mathcal{U} = \sum_{i=0}^{\mathcal{D}+1} d_i$ and $\mathcal{S} = \sum_{i=0}^{\mathcal{D}} d_i d_{i+1} + d_{i+1}$. Then the big network has $3\mathcal{D}+3$ hidden layers (totally $3\mathcal{D}+5$ layers), $d_0 + \sum_{i=1}^{\mathcal{D}} 17d_i + 11 \leq 17\mathcal{U}$ neurons, $2d_0d_1 + 23d_1 + \sum_{i=1}^{\mathcal{D}}(6d_id_{i+1} + 23d_{i+1}) + 4d_{\mathcal{D}} + 18 \leq 23\mathcal{S}$ parameters and its width is $10\max\{d_1, \ldots, d_{\mathcal{D}}\} = 10\mathcal{W}$. This completes the proof. □

Proof of Lemma 13

Our proof has two parts. In the first part, we follow the idea of the proof of Theorem 6 in Bartlett et al. (2019) to prove a somewhat stronger result, where we give the upper bound of the Pseudo dimension of $\mathcal{F}$ in terms of the depth, size and number of neurons of the network. Instead of the VC dimension of sign($\mathcal{F}$) given in Bartlett et al. (2019), our Pseudo dimension bound is stronger since $\mathrm{VCdim}(\mathrm{sign}(\mathcal{F})) \leq \mathrm{Pdim}(\mathcal{F})$. In the second part, based on Theorem 2.2 in Goldberg and Jerrum (1995), we also follow and improve the result in Theorem 8 of Bartlett et al. (2019) to give an upper bound of the Pseudo dimension of $\mathcal{F}$ in terms of the size and number of neurons of the network.

Part I

Let $\mathcal{Z}$ denote the domain of the functions $f \in \mathcal{F}$ and let $t \in \mathbb{R}$, we consider a new class of functions

$$\tilde{\mathcal{F}} := \{\tilde{f}(z,t) = \mathrm{sign}(f(z) - t) : f \in \mathcal{F}\}.$$

Then it is clear that $\mathrm{Pdim}(\mathcal{F}) \leq \mathrm{VCdim}(\tilde{\mathcal{F}})$ and we next bound the VC dimension of $\tilde{\mathcal{F}}$. Recall that the the total number of parameters (weights and biases) in the neural network implementing functions in $\mathcal{F}$ is $\mathcal{S}$, we let $\theta \in \mathbb{R}^{\mathcal{S}}$ denote the parameters vector of the network $f(\cdot, \theta) : \mathcal{Z} \to \mathbb{R}$ implemented in $\mathcal{F}$. And here we intend to derive a bound for

$$K(m) := \Big|\{(\mathrm{sign}(f(z_1,\theta) - t_1), \ldots, \mathrm{sign}(f(z_m,\theta) - t_m)) : \theta \in \mathbb{R}^{\mathcal{S}}\}\Big|$$

which uniformly hold for all choice of $\{z_i\}_{i=1}^m$ and $\{t_i\}_{i=1}^m$. Note that the maximum of $K(m)$ over all all choice of $\{z_i\}_{i=1}^m$ and $\{t_i\}_{i=1}^m$ is just the growth function of $\tilde{\mathcal{F}}$. To give a uniform bound of $K(m)$, we use the Theorem 8.3 in Anthony and Bartlett (1999) as a main tool to deal with the analysis.

Lemma 18 (Theorem 8.3 in Anthony and Bartlett (1999)) *Let $p_1, \ldots, p_m$ be polynomials in n variables of degree at most d. If $n \leq m$, define*

$$K := |\{(\mathrm{sign}(p_1(x), \ldots, \mathrm{sign}(p_m(x))) : x \in \mathbb{R}^n)\}|,$$

i.e. K is the number of possible sign vectors given by the polynomials. Then $K \leq 2(2emd/n)^n$.

Now if we can find a partition $\mathcal{P} = \{P_1, \ldots, P_N\}$ of the parameter domain $\mathbb{R}^{\mathcal{S}}$ such that within each region P_i, the functions $f(z_j, \cdot)$ are all fixed polynomials of bounded degree, then $K(m)$ can be bounded via the following sum

$$K(m) \leq \sum_{i=1}^{N} \Big|\{(\mathrm{sign}(f(z_1,\theta) - t_1), \ldots, \mathrm{sign}(f(z_m,\theta) - t_m)) : \theta \in P_i\}\Big|, \tag{27}$$

and each term in this sum can be bounded via Lemma 18. Next, we construct the partition follows the same way as in Bartlett et al. (2019) iteratively layer by layer. We define the a sequence of successive refinements $\mathcal{P}_1, \ldots, \mathcal{P}_{\mathcal{D}}$ satisfying the following properties:

1. The cardinality $|\mathcal{P}_1| = 1$ and for each $n \in \{1, \ldots, \mathcal{D}\}$,
$$\frac{|\mathcal{P}_{n+1}|}{|\mathcal{P}_n|} \leq 2\Big(\frac{2emk_n(1+(n-1)2^{n-1})}{\mathcal{S}_n}\Big)^{\mathcal{S}_n},$$
where k_n denotes the number of neurons in the n-th layer and $\mathcal{S}_n$ denotes the total number of parameters (weights and biases) at the inputs to units in all the layers up to layer n.

2. For each $n \in \{1, \ldots, \mathcal{D}\}$, each element of P of $\mathcal{P}_n$, each $j \in \{1, \ldots, m\}$, and each unit u in the n-th layer, when θ varies in P, the net input to u is a fixed polynomial function in $\mathcal{S}_n$ variables of θ, of total degree no more than $1 + (n-1)2^{n-1}$ (this polynomial may depend on P, j and u.)

One can define $\mathcal{P}_1 = \mathbb{R}^{\mathcal{S}}$, and it can be verified that $\mathcal{P}_1$ satisfies property 2 above. Note that in our case, for fixed z_j and t_j and any subset $P \subset \mathbb{R}^{\mathcal{S}}$, $f(z_j, \theta) - t_j$ is a polynomial with respect to θ with degree the same as that of $f(z_j, \theta)$, which is no more than $1+(\mathcal{D}-1)2^{\mathcal{D}-1}$. Then the construction of $\mathcal{P}_1, \ldots, \mathcal{P}_{\mathcal{D}}$ and its verification for properties 1 and 2 can follow the same way in Bartlett et al. (2019). Finally we obtain a partition $\mathcal{P}_{\mathcal{D}}$ of $\mathbb{R}^{\mathcal{S}}$ such that for $P \in \mathcal{P}_{\mathcal{D}}$, the network output in response to any z_j is a fixed polynomial of $\theta \in P$ of degree no more than $1 + (\mathcal{D}-1)2^{\mathcal{D}-1}$ (since the last node just outputs its input). Then by Lemma 18

$$\Big|\{(\operatorname{sign}(f(z_1,\theta)-t_1), \ldots, \operatorname{sign}(f(z_m,\theta)-t_m)) : \theta \in P\}\Big| \leq 2\Big(\frac{2em(1+(\mathcal{D}-1)2^{\mathcal{D}-1})}{\mathcal{S}_{\mathcal{D}}}\Big)^{\mathcal{S}_{\mathcal{D}}}.$$

Besides, by property 1 we have

$$|\mathcal{P}_{\mathcal{D}}| \leq \Pi_{i=1}^{\mathcal{D}-\infty} 2\Big(\frac{2emk_i(1+(i-1)2^{i-1})}{\mathcal{S}_i}\Big)^{\mathcal{S}_i}.$$

Then using (27), and since the sample $z_1, \ldots, Z_m$ are arbitrarily chosen, we have

$$\begin{aligned}
K(m) &\leq \Pi_{i=1}^{\mathcal{D}} 2\Big(\frac{2emk_i(1+(i-1)2^{i-1})}{\mathcal{S}_i}\Big)^{\mathcal{S}_i} \\
&\leq 2^{\mathcal{D}}\Big(\frac{2em\sum k_i(1+(i-1)2^{i-1})}{\sum \mathcal{S}_i}\Big)^{\sum \mathcal{S}_i} \\
&\leq \Big(\frac{4em(1+(\mathcal{D}-1)2^{\mathcal{D}-1})\sum k_i}{\sum \mathcal{S}_i}\Big)^{\sum \mathcal{S}_i} \\
&\leq \Big(4em(1+(\mathcal{D}-1)2^{\mathcal{D}-1})\Big)^{\sum \mathcal{S}_i},
\end{aligned}$$

where the second inequality follows from weighted arithmetic and geometric means inequality, the third holds since $\mathcal{D} \leq \sum \mathcal{S}_i$ and the last holds since $\sum k_i \leq \sum \mathcal{S}_i$. Since $K(m)$ is the growth function of $\tilde{\mathcal{F}}$, we have

$$2^{\mathrm{Pdim}(\mathcal{F})} \leq 2^{\mathrm{VCdim}(\tilde{\mathcal{F}})} \leq K(\mathrm{VCdim}(\tilde{\mathcal{F}})) \leq 2^{\mathcal{D}}\Big(\frac{2emR \cdot \mathrm{VCdim}(\tilde{\mathcal{F}})}{\sum \mathcal{S}_i}\Big)^{\sum \mathcal{S}_i}$$

where $R := \sum_{i=1}^{\mathcal{D}} k_i(1+(i-1)2^{i-1}) \le \mathcal{U} + \mathcal{U}(\mathcal{D}-1)2^{\mathcal{D}-1}$. Since $\mathcal{U} > 0$ and $2eR \ge 16$, then by Lemma 16 in Bartlett et al. (2019) we have

$$\mathrm{Pdim}(\mathcal{F}) \le \mathcal{D} + (\sum_{i=1}^{n} \mathcal{S}_i) \log_2(4eR\log_2(2eR)).$$

Note that $\sum_{i=1}^{\mathcal{D}} \mathcal{S}_i \le \mathcal{D}\mathcal{S}$ and $\log_2(R) \le \log_2(\mathcal{U}\{1+(\mathcal{D}-1)2^{\mathcal{D}-1}\}) \le \log_2(\mathcal{U}) + 2\mathcal{D}$, then we have

$$\mathrm{Pdim}(\mathcal{F}) \le \mathcal{D} + \mathcal{D}\mathcal{S}(4\mathcal{D} + 2\log_2 \mathcal{U} + 6) \le 7\mathcal{D}\mathcal{S}(\mathcal{D} + \log_2 \mathcal{U}))$$

for some universal constant $c > 0$.

Part II

We first list Theorem 2.2 in Goldberg and Jerrum (1995).

Lemma 19 (Theorem 2.2 in Goldberg and Jerrum (1995)) *Let k, n be positive integers and $f : \mathbb{R}^n \times \mathbb{R}^k \to \{0, 1\}$ be a function that can be expressed as a Boolean formula containing s distinct atomic predicates where each atomic predicate is a polynomial inequality or equality in $k + n$ variables of degree at most d. Let $\mathcal{F} = \{f(\cdot, w) : w \in \mathbb{R}^k\}$. Then* $\mathrm{VCdim}(\mathcal{F}) \le 2k \log_2(8eds)$.

Suppose the functions in $f \in \mathcal{F}$ are implemented by ReLU-REQU neural networks with $\mathcal{S}$ parameters (weights and bias) and $\mathcal{U}$ neurons. The activation function of $f \in \mathcal{F}$ is piecewise polynomial of degree at most 2 with 2 pieces. As in Part I of the proof, let $\mathcal{Z}$ denote the domain of the functions $f \in \mathcal{F}$ and let $t \in \mathbb{R}$, we consider the class of functions

$$\tilde{\mathcal{F}} := \{\tilde{f}(z, t) = \mathrm{sign}(f(z) - t) : f \in \mathcal{F}\}.$$

Since the outputs of functions in $\tilde{\mathcal{F}}$ are 0 or 1, to apply above lemma, we intend to show that the function in $\tilde{\mathcal{F}}$ as Boolean functions consisting of no more than $2 \cdot 3^{\mathcal{U}}$ atomic predicates with each being a polynomial inequality of degree at most $3 \cdot 2^{\mathcal{U}}$. We topologically sort the neurons of the network since the neural network graph is acyclic. Let u_i be the i-th neuron in the topological ordering for $i = 1, \ldots, \mathcal{U}+1$. Note that the input to each neuron u comes from one of the 2 pieces of the activation function ReLU or ReQU, then we call "u_i is in the state j" if the input of u_i lies in the j-th piece for $i \in \{1, \ldots, \mathcal{U}+1\}$ and $j \in \{1, 2\}$. For u_1 and $j \in \{1, 2\}$, the predicate "u_1 is in state$j$" is a single atomic predicate thus the state of u_1 can be expressed as a function of 2 atomic predicates. Given that u_1 is in a certain state, the state of u_2 can be decided by 2 atomic predicates, which are polynomial inequalities of degree at most $2 \times 2 + 1$. Hence the state of u_2 can be determined using $2 + 2^2$ atomic predicates, each of which is a polynomial of degree no more than $2 \times 2 + 1$. By induction, the state of u_i is decided using $2(1+2)^{i-1}$ atomic predicates, each of which is a polynomial of degree at most $2^{i-1} + \sum_{j=0}^{i-1} 2^j$. Then the state of all neurons can be decided using no more than $3^{\mathcal{U}+1}$ atomic predicates, each of which is a polynomial of degree at most $2^{\mathcal{U}} + \sum_{j=0}^{\mathcal{U}} 2^j \le 3 \cdot 2^{\mathcal{U}}$. Then by Lemma 19, an upper bound for $\mathrm{VCdim}(\tilde{\mathcal{F}})$ is $2\mathcal{S}\log_2(8e \cdot 3^{\mathcal{U}+1} \cdot 3 \cdot 2^{\mathcal{U}}) = 2\mathcal{S}[\log_2(6^{\mathcal{U}}) + \log_2(72e)] \le 22\mathcal{S}\mathcal{U}$. Since $\mathrm{Pdim}(\mathcal{F}) \le \mathrm{VCdim}(\tilde{\mathcal{F}})$, then the upper bounds hold also for $\mathrm{Pdim}(\mathcal{F})$,

$$\mathrm{Pdim}(\mathcal{F}) \le 22\mathcal{S}\mathcal{U},$$

which completes the proof. □

Proof of Theorem 14

Recall that the stochastic error is

$$\sup_{f\in\mathcal{F}_n}\Big|[\mathcal{R}^{\xi}_{\lambda}(f)-\mathcal{R}^{\xi}_{\lambda}(f_0)]-[\mathcal{R}^{\xi}_{n,\lambda}(f)-\mathcal{R}^{\xi}_{n,\lambda}(f_0)]\Big|. \tag{28}$$

Since $\mathcal{R}^{\xi}_{\lambda}(f)=\mathcal{R}^{\xi}(f)+\lambda\kappa(f)$, then the stochastic error can be further bounded by

$$\begin{aligned}\sup_{f\in\mathcal{F}_n}&\Big|[\mathcal{R}^{\xi}_{\lambda}(f)-\mathcal{R}^{\xi}_{\lambda}(f_0)]-[\mathcal{R}^{\xi}_{n,\lambda}(f)-\mathcal{R}^{\xi}_{n,\lambda}(f_0)]\Big|\\ &\le\sup_{f\in\mathcal{F}_n}\Big|[\mathcal{R}^{\xi}(f)-\mathcal{R}^{\xi}(f_0)]-[\mathcal{R}^{\xi}_{n}(f)-\mathcal{R}^{\xi}_{n}(f_0)]\Big| \qquad (29)\\ &+\lambda\sup_{f\in\mathcal{F}_n}\Big|[\kappa(f)-\kappa(f_0)]-[\kappa_n(f)-\kappa_n(f_0)]\Big|. \qquad (30)\end{aligned}$$

We firstly bound the first item (29) of empirical process $\sup_{f\in\mathcal{F}_n}|[\mathcal{R}^{\xi}(f)-\mathcal{R}^{\xi}(f_0)]-[\mathcal{R}^{\xi}_n(f)-\mathcal{R}^{\xi}_n(f_0)]|$, and the second one can be dealt with similarly.

We consider the difference of the loss items in the empirical process. Define

$$L(f,Z_i):=\rho_{\xi_i}(Y_i-f(X_i,\xi_i))-\rho_{\xi_i}(Y_i-f_0(X_i,\xi_i)),$$

for sample $Z_i=(X_i,Y_i,\xi_i), i=1,\ldots,n$, and $f\in\mathcal{F}_n$. By the Lipschitz property of the check loss ρ_ξ and the assumption that f_0 and functions in $\mathcal{F}_n$ are uniformly bounded by $\mathcal{B}$, we have $L(f,Z_i)\in[-\mathcal{B},\mathcal{B}]$.

For any given $\epsilon>0$, let $f_1,f_2,\ldots,f_{\mathcal{N}}$ be the anchor points of an ϵ-covering for the function class $\mathcal{F}_n$, and we denote $\mathcal{N}:=\mathcal{N}_n(\epsilon,\mathcal{F}_n,\|\cdot\|_\infty)$ as the covering number of $\mathcal{F}_n$ with radius ϵ under the norm $\|\cdot\|_\infty$. By definition, for any $f\in\mathcal{F}_n$, there exists an anchor f_j for $j\in\{1,\ldots,\mathcal{N}\}$ such that $\|f_j-f\|_\infty\le\epsilon$. The Lipschitz property of ρ_ξ them implies $|L(f,Z_i)-L(f_j,Z_i)|\le\epsilon$, $|\mathcal{R}^{\xi}(f)-\mathcal{R}^{\xi}(f_j)|\le\epsilon$ and $|\mathcal{R}^{\xi}_n(f)-\mathcal{R}^{\xi}(f_j)_n|\le\epsilon$. Then triangular inequality gives

$$\big|[\mathcal{R}^{\xi}(f)-\mathcal{R}^{\xi}(f)]-[\mathcal{R}^{\xi}_n(f)-\mathcal{R}^{\xi}_n(f_0)]\big|\le\big|[\mathcal{R}^{\xi}(f_j)-\mathcal{R}^{\xi}(f_0)]-[\mathcal{R}^{\xi}_n(f_j)-\mathcal{R}^{\xi}_n(f_0)]\big|+2\epsilon.$$

Then, for any $t>0$ we have,

$$\begin{aligned}&\mathbb{P}\Big(\sup_{f\in\mathcal{F}_n}\big|[\mathcal{R}^{\xi}(f)-\mathcal{R}^{\xi}(f)]-[\mathcal{R}^{\xi}_n(f)-\mathcal{R}^{\xi}_n(f_0)]\big|\ge t+2\epsilon\Big)\\ \le&\mathbb{P}\Big(\exists j\in\{1,\ldots,\mathcal{N}\}:|[\mathcal{R}^{\xi}(f_j)-\mathcal{R}^{\xi}(f_0)]-[\mathcal{R}^{\xi}_n(f_j)-\mathcal{R}^{\xi}_n(f_0)]\big|\ge t\Big)\\ \le&\mathcal{N}_n(\epsilon,\mathcal{F}_n,\|\cdot\|_\infty)\max_{j\in\{1,\ldots,\mathcal{N}\}}\mathbb{P}\Big(|[\mathcal{R}^{\xi}(f_j)-\mathcal{R}^{\xi}(f_0)]-[\mathcal{R}^{\xi}_n(f_j)-\mathcal{R}^{\xi}_n(f_0)]\big|\ge t\Big)\\ =&\mathcal{N}_n(\epsilon,\mathcal{F}_n,\|\cdot\|_\infty)\max_{j\in\{1,\ldots,\mathcal{N}\}}\mathbb{P}\Big(|\sum_{i=1}^n L(f_j,Z_i)-n\mathbb{E}[L(f_j,Z)])|\ge nt\Big)\\ \le&2\mathcal{N}_n(\epsilon,\mathcal{F}_n,\|\cdot\|_\infty)\exp\left(-\frac{nt^2}{2\mathcal{B}^2}\right), \qquad (31)\end{aligned}$$

where the last inequality comes from the Hoeffding's inequality.

Given any $\delta > 0$, we let $\epsilon = 1/n$ and $t = \sqrt{2}\mathcal{B}\sqrt{\log(2\mathcal{N}_n(1/n, \mathcal{F}_n, \|\cdot\|_\infty)/\delta)/n}$. Then the right-hand side of (31) equals to δ, we have

$$\mathbb{P}\Big(\sup_{f\in\mathcal{F}_n}\big|[\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f)] - [\mathcal{R}^\xi_n(f) - \mathcal{R}^\xi_n(f_0)]\big| \geq t + 2\epsilon\Big) \leq \delta.$$

Equivalently, with probability at least $1-\delta$ we have

$$\begin{aligned}&\sup_{f\in\mathcal{F}_n}\big|[\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f)] - [\mathcal{R}^\xi_n(f) - \mathcal{R}^\xi_n(f_0)]\big| \leq t + 2\epsilon\\ \leq&\frac{\sqrt{2}\mathcal{B}\sqrt{\log 2\mathcal{N}_n(1/n, \mathcal{F}_n, \|\cdot\|_\infty)}}{\sqrt{n}} + \frac{\sqrt{2}\mathcal{B}\sqrt{\log(1/\delta)}}{\sqrt{n}} + \frac{2}{n},\end{aligned}$$

where the inequality follows from the fact that $\sqrt{c+d} \leq \sqrt{c} + \sqrt{d}$ for any $c, d \geq 0$. Furthermore, we know from Lemma 25 in Appenthat with probability at least $1-\delta$

$$\begin{aligned}&\sup_{f\in\mathcal{F}_n}\big|[\mathcal{R}^\xi(f) - \mathcal{R}^\xi(f)] - [\mathcal{R}^\xi_n(f) - \mathcal{R}^\xi_n(f_0)]\big|\\ &\leq \frac{\sqrt{2}\mathcal{B}}{\sqrt{n}}\left(C\sqrt{\log\left(\frac{en^2\mathcal{B}}{\mathrm{Pdim}(\mathcal{F}_n)}\right)^{\mathrm{Pdim}(\mathcal{F}_n)}} + \sqrt{\log(1/\delta)}\right) + \frac{2}{n}\\ &\leq \frac{\sqrt{2}\mathcal{B}}{\sqrt{n}}\left(C\sqrt{\mathrm{Pdim}(\mathcal{F}_n)\log n} + \sqrt{\log(1/\delta)}\right) + \frac{2}{n},\end{aligned} \tag{32}$$

for $n \geq \mathrm{Pdim}(\mathcal{F}_n)$, where $\mathrm{Pdim}(\mathcal{F}_n)$ is the pseudo-dimension of $\mathcal{F}_n$ stated in Definition 23 in Appendix B and $C > 0$ is a universal constant.

Following similar arguments as above, we can get the upper bound for the other empirical process. Given any $\delta > 0$, we can obtain with probability at least $1-\delta$

$$\lambda\sup_{f\in\mathcal{F}_n}\Big|[\kappa(f) - \kappa(f_0)] - [\kappa_n(f) - \kappa_n(f_0)]\Big| \leq \frac{\sqrt{2}\lambda\mathcal{B}'}{\sqrt{n}}\left(C\sqrt{\mathrm{Pdim}(\mathcal{F}'_n)\log n} + \sqrt{\log(1/\delta)}\right) + \frac{2}{n} \tag{33}$$

for $n \geq \mathrm{Pdim}(\mathcal{F}'_n)$ where $\mathcal{F}'_n$ is the function class of derivatives of ReQU networks in $\mathcal{F}_n$. Next, combining (29), (30), (32) and (33) we know that for any $\delta > 0$, with probability at least $1-\delta$ it holds

$$\begin{aligned}&\sup_{f\in\mathcal{F}_n}\Big|[\mathcal{R}^\xi_\lambda(f) - \mathcal{R}^\xi_\lambda(f_0)] - [\mathcal{R}^\xi_{n,\lambda}(f) - \mathcal{R}^\xi_{n,\lambda}(f_0)]\Big|\\ &\leq \frac{\sqrt{2}}{\sqrt{n}}\left(C\sqrt{\log(n)}[\lambda\mathcal{B}'\sqrt{\mathrm{Pdim}(\mathcal{F}'_n)} + \mathcal{B}\sqrt{\mathrm{Pdim}(\mathcal{F}_n)}] + 2(\mathcal{B} + \lambda\mathcal{B}')\sqrt{\log(2/\delta)}\right) + \frac{4}{n}\end{aligned} \tag{34}$$

for $n \geq \max\{\mathrm{Pdim}(\mathcal{F}_n), \mathrm{Pdim}(\mathcal{F}'_n)\}$ and some universal constant $c_0 > 0$. By Lemma 13, for the function class $\mathcal{F}_n$ implemented by ReLU-ReQU activated multilayer perceptrons with depth no more than $\mathcal{D}$, width no more than $\mathcal{W}$, number of neurons (nodes) no more than $\mathcal{U}$ and size or number of parameters (weights and bias) no more than $\mathcal{S}$, we have

$$\mathrm{Pdim}(\mathcal{F}_n) \leq \min\{7\mathcal{D}\mathcal{S}(\mathcal{D} + \log_2\mathcal{U}), 22\mathcal{U}\mathcal{S}\},$$

and by Lemma 12, for any function $f \in \mathcal{F}_n$, its partial derivative $\frac{\partial}{\partial \tau} f$ can be implemented by a ReLU-ReQU activated multilayer perceptron with depth $3\mathcal{D}+3$, width $10\mathcal{W}$, number of neurons $17\mathcal{U}$, number of parameters $23\mathcal{S}$ and bound $\mathcal{B}'$. Then

$$\mathrm{Pdim}(\mathcal{F}_n') \leq \min\{5796\mathcal{D}\mathcal{S}(\mathcal{D}+\log_2\mathcal{U}), 8602\mathcal{U}\mathcal{S}\}.$$

Finally, for any $\delta > 0$, with probability at least $1-\delta$

$$\begin{aligned}&\sup_{f\in\mathcal{F}_n}\left|[\mathcal{R}^{\xi}_{\lambda}(f)-\mathcal{R}^{\xi}_{\lambda}(f_0)]-[\mathcal{R}^{\xi}_{n,\lambda}(f)-\mathcal{R}^{\xi}_{n,\lambda}(f_0)]\right|\\ &\leq c_1\big(\mathcal{B}+\lambda\mathcal{B}'\big)\sqrt{\frac{\min\{5796\mathcal{D}\mathcal{S}(\mathcal{D}+\log_2\mathcal{U}), 8602\mathcal{U}\mathcal{S}\}\log(n)}{n}}+\frac{9(\mathcal{B}+\lambda\mathcal{B}')}{\sqrt{n}}\sqrt{\log(2/\delta)}\Big),\end{aligned}$$

and

$$\begin{aligned}&\mathbb{E}\left[\sup_{f\in\mathcal{F}_n}\left|[\mathcal{R}^{\xi}_{\lambda}(f)-\mathcal{R}^{\xi}_{\lambda}(f_0)]-[\mathcal{R}^{\xi}_{n,\lambda}(f)-\mathcal{R}^{\xi}_{n,\lambda}(f_0)]\right|\right]\\ &\qquad\leq c_2\big(\mathcal{B}+\lambda\mathcal{B}'\big)\sqrt{\frac{\min\{5796\mathcal{D}\mathcal{S}(\mathcal{D}+\log_2\mathcal{U}), 8602\mathcal{U}\mathcal{S}\}\log(n)}{n}},\end{aligned}$$

for $n \geq \max\{\mathrm{Pdim}(\mathcal{F}_n), \mathrm{Pdim}(\mathcal{F}_n')\}$ and some universal constants $c_1, c_2 > 0$. This completes the proof. □

Proof of Theorem 15

The idea of the proof is to construct a ReQU network that computes a multivariate polynomial with degree N with no error. We begin our proof with consider the simple case, which is to construct a proper ReQU network to represent a univariate polynomial with no error. Recall that to approximate the multiplication operator is simple and straightforward, we can leverage Horner's method or Qin Jiushao's algorithm in China to construct such networks. Suppose $f(x) = a_0 + a_1x + \cdots + a_Nx^N$ is a univariate polynomial of degree N, then it can be written as

$$f(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{N-1} + xa_N)))).$$

We can illiterately calculate a sequence of intermediate variables $b_1, \ldots, b_N$ by

$$b_k = \begin{cases} a_{N-1} + xa_N, & k = 1, \\ a_{N-k} + xb_{N-1}, & k = 2, \ldots, N. \end{cases}$$

Then we can obtain $b_N = f(x)$. By the basic approximation property we know that to calculate b_1 needs a ReQU network with 1 hidden layer and 4 hidden neurons, and to calculate b_2 needs a ReQU network with 3 hidden layer, $2 \times 4 + 2 - 1$ hidden neurons. By induction, to calculate $b_N = f(x)$ needs a ReQU network with $2N-1$ hidden layer, $N \times 4 + N - 1 = 5N - 1$ hidden neurons, $8N$ parameters(weights and bias), and its width equals to 4.

Apart from the construction based on the Horner's method, another construction is shown in Theorem 2.2 of Li et al. (2019a), where the constructed ReQU network has

$\lfloor \log_2 N \rfloor + 1$ hidden layers, $8N-2$ neurons and no more than $61N$ parameters (weights and bias).

Now we consider constructing ReQU networks to compute multivariate polynomial f with total degree N on $\mathbb{R}^d$. For any $d \in \mathbb{N}^+$ and $N \in \mathbb{N}_0$, let

$$f_N^d(x_1, \ldots, x_d) = \sum_{i_1+\cdots+i_d=0}^{N} a_{i_1,i_2,\ldots,i_d} x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d},$$

denote the polynomial with total degree N of d variables, where $i_1, i_2, \ldots, i_d$ are non-negative integers, $\{a_{i_1,i_2,\ldots,i_d} : i_1 + \cdots + i_d \le N\}$ are coefficients in $\mathbb{R}$. Note that the multivariate polynomial f_N^d can be written as

$$f_N^d(x_1, \ldots, x_d) = \sum_{i_1=0}^{N} \Big(\sum_{i_2+\cdots+i_d=0}^{N-i_1} a_{i_1,i_2\ldots,i_d} x_2^{i_2} \cdots x_d^{i_d} \Big) x_1^{i_1},$$

and we can view f_N^d as a univariate polynomial of x_1 with degree N if $x_2, \ldots, x_d$ are given and for each $i_1 \in \{0, \ldots, N\}$ the $(d-1)$-variate polynomial $\sum_{i_2+\cdots+i_d=0}^{N-i_1} a_{i_1,i_2\ldots,i_d} x_2^{i_2} \cdots x_d^{i_d}$ with degree no more than N can be computed by a proper ReQU network. This reminds us the construction of ReQU network for f_N^d can be implemented iteratively via composition of $f_N^1, f_N^2, \ldots, f_N^d$ by induction.

By Horner's method we have constructed a ReQU network with $2N-1$ hidden layers, $5N-1$ hidden neurons and $8N$ parameters to exactly compute f_N^1. Now we start to show f_N^2 can be computed by proper ReQU networks. We can write f_N^2 as

$$f_N^2(x_1, x_2) = \sum_{i+j=0}^{N} a_{ij} x_1^i x_2^j = \sum_{i=0}^{N} \Big(\sum_{j=0}^{N-i} a_{ij} x_2^j \Big) x_1^i.$$

Note that for $i \in \{0, \ldots, N\}$, the the degree of polynomial $\sum_{j=0}^{N-i} a_{ij} x_2^j$ is $N-i$ which is less than N. But we can still view it as a polynomial with degree N by padding (adding zero terms) such that $\sum_{j=0}^{N-i} a_{ij} x_2^j = \sum_{j=0}^{N} a_{ij}^* x_2^j$ where $a_{ij}^* = a_{ij}$ if $i+j \le N$ and $a_{ij}^* = 0$ if $i+j > N$. In such a way, for each $i \in \{0, \ldots, N\}$ the polynomial $\sum_{j=0}^{N-i} a_{ij} x_2^j$ can be computed by a ReQU network with $2N-1$ hidden layers, $5N-1$ hidden neurons, $8N$ parameters and its width equal to 4. Besides, for each $i \in \{0, \ldots, N\}$, the monomial x^i can also be computed by a ReQU network with $2N-1$ hidden layers, $5N-1$ hidden neurons, $8N$ parameters and its width equal to 4, in whose implementation the identity maps are used after the $(2i-1)$-th hidden layer. Now we parallel these two sub networks to get a ReQU network which takes x_1 and x_2 as input and outputs $(\sum_{j=0}^{N-i} a_{ij} x_2^j) x^i$ with width 8, hidden layers $2N-1$, number of neurons $2 \times (5N-1)$ and size $2 \times 8N$. Since for each $i \in \{0, \ldots, N\}$, such paralleled ReQU network can be constructed, then with straightforward paralleling of N such ReQU networks, we obtain a ReQU network exactly computes f_N^2 with width $8N$, hidden layers $2N-1$, number of neurons $2 \times (5N-1) \times N$ and number of parameters $2 \times 8N \times N = 16N^2$.

Similarly for polynomial f_N^3 of 3 variables, we can write f_N^3 as

$$f_N^3(x_1, x_2, x_3) = \sum_{i+j+k=0}^{N} a_{ijk} x_1^i x_2^j x_3^k = \sum_{i=0}^{N} \Big(\sum_{j+k=0}^{N-i} a_{ijk} x_2^j x_3^k \Big) x_1^i.$$

By our previous argument, for each $i \in \{0, \ldots, N\}$, there exists a ReQU network which takes (x_1, x_2, x_3) as input and outputs $\left(\sum_{j+k=0}^{N-i} a_{ijk} x_2^j x_3^k\right) x_1^i$ with width $8N+4$, hidden layers $2N-1$, number of neurons $2N(5N-1)+(5N-1)$ and parameters $16N^2+8N$. And by paralleling N such subnetworks, we obtain a ReQU network that exactly computes f_N^3 with width $(8N+4) \times N = 8N^2+4N$, hidden layers $2N-1$, number. of neurons $N(2N(5N-1)+(5N-1)) = 2N^2(5N-1)+N(5N-1)$ and number of parameters $16N^3+8N^2$.

Continuing this process, we can construct ReQU networks exactly compute polynomials of any d variables with total degree N. With a little bit abuse of notations, we let $\mathcal{W}_k$, $\mathcal{D}_k$, $\mathcal{U}_k$ and $\mathcal{S}_k$ denote the width, number of hidden layers, number of neurons and number of parameters (weights and bias) respectively of the ReQU network computing f_N^k for $k = 1, 2, 3, \ldots$. We have known that

$$\mathcal{D}_1 = 2N-1 \qquad \mathcal{W}_1 = 4 \qquad \mathcal{U}_1 = 5N-1 \qquad \mathcal{S}_1 = 8N$$

Besides, based on the iterate procedure of the network construction, by induction we can see that for $k = 2, 3, 4, \ldots$ the following equations hold,

$$\begin{aligned}
\mathcal{D}_k =& 2N-1, \\
\mathcal{W}_k =& N \times (\mathcal{W}_{k-1} + \mathcal{W}_1), \\
\mathcal{U}_k =& N \times (\mathcal{U}_{k-1} + \mathcal{U}_1), \\
\mathcal{S}_k =& N \times (\mathcal{S}_{k-1} + \mathcal{S}_1).
\end{aligned}$$

Then based on the values of $\mathcal{D}_1, \mathcal{W}_1, \mathcal{U}_1, \mathcal{S}_1$ and the recursion formula, we have for $k = 2, 3, 4, \ldots$

$$\begin{gathered}
\mathcal{D}_k = 2N-1, \\
\mathcal{W}_k = 8N^{k-1} + 4\frac{N^{k-1}-N}{N-1} \leq 12N^{k-1}, \\
\mathcal{U}_k = N \times (\mathcal{U}_{k-1} + \mathcal{U}_1) = 2(5N-1)N^{k-1} + (5N-1)\frac{N^{k-1}-N}{N-1} \leq 15N^k, \\
\mathcal{S}_k = N \times (\mathcal{S}_{k-1} + \mathcal{S}_1) = 16N^k + 8N\frac{N^{k-1}-N}{N-1} \leq 24N^k.
\end{gathered}$$

This completes our proof. □

Proof of Theorem 16

The proof is straightforward by and leveraging the approximation power of multivariate polynomials since Theorem 15 told us any multivariate polynomial can be represented by proper ReQU networks. The theories for polynomial approximation have been extensively studies on various spaces of smooth functions. We refer to Bagby et al. (2002) for the polynomial approximation on smooth functions in our proof.

Lemma 20 (Theorem 2 in Bagby et al. (2002)) *Let f be a function of compact support on $\mathbb{R}^d$ of class C^s where $s \in \mathbb{N}^+$ and let K be a compact subset of $\mathbb{R}^d$ which contains* *the support of f. Then for each nonnegative integer N there is a polynomial p_N of degree at most N on $\mathbb{R}^d$ with the following property: for each multi-index α with $|\alpha|_1 \leq \min\{s, N\}$ we have*

$$\sup_K |D^\alpha (f - p_N)| \leq \frac{C}{N^{s-|\alpha|_1}} \sum_{|\alpha|_1 \leq s} \sup_K |D^\alpha f|,$$

where C is a positive constant depending only on d, s and K.

The proof of Lemma 20 can be found in Bagby et al. (2002) based on the Whitney extension theorem (Theorem 2.3.6 in Hörmander (2015)). To use Lemma 20, we need to find a ReQU network to compute the p_N for each $N \in \mathbb{N}^+$. By Theorem 15, we know that any p_N of $d+1$ variables can be computed by a ReQU network with $2N-1$ hidden layer, no more than $15N^{d+1}$ neurons, no more than $24N^{d+1}$ parameters and width no more than $12N^d$. This completes the proof. By examining the proof of Theorem 1 in Bagby et al. (2002), the dependence of the constant C in Lemma 20 on the d, s and K can be detailed. □

Proof of Corollary 17

Recall that

$$\begin{aligned}
&\inf_{f \in \mathcal{F}_n} \Big[\mathcal{R}(f) - \mathcal{R}(f_0) + \lambda\{\kappa(f) - \kappa(f_0)\}\Big] \\
&= \inf_{f \in \mathcal{F}_n} \Bigg[\mathbb{E}_{X,Y,\xi}\{\rho_\xi(Y - f(X,\xi)) - \rho_\xi(Y - f_0(X,\xi)) \\
&\qquad\qquad + \lambda(\max\{-\frac{\partial}{\partial \tau} f(X,\xi), 0\} - \max\{-\frac{\partial}{\partial \tau} f_0(X,\xi), 0\})\}\Bigg] \\
&\leq \inf_{f \in \mathcal{F}_n} \Big[\mathbb{E}_{X,Y,\xi}\Big\{|f(X,\xi) - f_0(X,\xi)| + \lambda|\frac{\partial}{\partial \tau} f(X,\xi) - \frac{\partial}{\partial \tau} f_0(X,\xi)|\Big\}\Big].
\end{aligned}$$

By Theorem 16, for each $N \in \mathbb{N}^+$, there exists a ReQU network $\phi_N \in \mathcal{F}_n$ with $2N-1$ hidden layer, no more than $15N^{d+1}$ neurons, no more than $24N^{d+1}$ parameters and width no more than $12N^d$ such that for each multi-index $\alpha \in \mathbb{N}_0^d$ with $|\alpha|_1 \leq \min\{s, N\}$ we have

$$\sup_{\mathcal{X} \times (0,1)} |D^\alpha (f - \phi_N)| \leq C(s, d, \mathcal{X}) \times N^{-(s - |\alpha|_1)} \|f\|_{C^s},$$

where $C(s, d, \mathcal{X})$ is a positive constant depending only on d, s and the diameter of $\mathcal{X} \times (0,1)$. This implies

$$\sup_{\mathcal{X} \times (0,1)} |f - \phi_N| \leq C(s, d, \mathcal{X}) \times N^{-s} \|f\|_{C^s},$$

and

$$\sup_{\mathcal{X} \times (0,1)} \Big|\frac{\partial}{\partial \tau}(f - \phi_N)\Big| \leq C(s, d, \mathcal{X}) \times N^{-(s-1)} \|f\|_{C^s}.$$

Combine above two uniform bounds, we have

$$
\begin{aligned}
&\inf_{f\in\mathcal{F}_n}\Big[\mathcal{R}(f)-\mathcal{R}(f_0)+\lambda\{\kappa(f)-\kappa(f_0)\}\Big]\\
\leq&\Big[|\mathbb{E}_{X,\xi}\Big\{\phi_N(X,\xi)-f_0(X,\xi)|+\lambda|\frac{\partial}{\partial\tau}\phi_N(X,\xi)-\frac{\partial}{\partial\tau}f_0(X,\xi)|\Big\}\Big]\\
\leq&C(s,d,\mathcal{X})\times N^{-s}\|f\|_{C^s}+\lambda C(s,d,\mathcal{X})\times N^{-(s-1)}\|f\|_{C^s}\\
\leq&C(s,d,\mathcal{X})(1+\lambda)N^{-(s-1)}\|f\|_{C^s},
\end{aligned}
$$

which completes the proof. □

Appendix B. Definitions and Supporting Lemmas

B.1 Definitions

The following definitions are used in the proofs.

Definition 21 (Covering number) *Let $\mathcal{F}$ be a class of function from $\mathcal{X}$ to $\mathbb{R}$. For a given sequence $x=(x_1,\ldots,x_n)\in\mathcal{X}^n$, let $\mathcal{F}_n|_x=\{(f(x_1),\ldots,f(x_n):f\in\mathcal{F}_n\}$ be the subset of $\mathbb{R}^n$. For a positive number δ, let $\mathcal{N}(\delta,\mathcal{F}_n|_x,\|\cdot\|_\infty)$ be the covering number of $\mathcal{F}_n|_x$ under the norm $\|\cdot\|_\infty$ with radius δ. Define the uniform covering number $\mathcal{N}_n(\delta,\|\cdot\|_\infty,\mathcal{F}_n)$ to be the maximum over all $x\in\mathcal{X}$ of the covering number $\mathcal{N}(\delta,\mathcal{F}_n|_x,\|\cdot\|_\infty)$, i.e.,*

$$\mathcal{N}_n(\delta,\mathcal{F}_n,\|\cdot\|_\infty)=\max\{\mathcal{N}(\delta,\mathcal{F}_n|_x,\|\cdot\|_\infty):x\in\mathcal{X}^n\}. \tag{35}$$

Definition 22 (Shattering) *Let $\mathcal{F}$ be a family of functions from a set $\mathcal{Z}$ to $\mathbb{R}$. A set $\{z_1,\ldots,Z_n\}\subset\mathcal{Z}$ is said to be shattered by $\mathcal{F}$, if there exists $t_1,\ldots,t_n\in\mathbb{R}$ such that*

$$\Big|\Big\{\Big[\begin{array}{c}\mathrm{sgn}(f(z_1)-t_1)\\ \cdots\\ \mathrm{sgn}(f(z_n)-t_n)\end{array}\Big]:f\in\mathcal{F}\Big\}\Big|=2^n,$$

where rmsgn is the sign function returns $+1$ or -1 and $|\cdot|$ denotes the cardinality of a set. When they exist, the threshold values $t_1,\ldots,t_n$ are said to witness the shattering.

Definition 23 (Pseudo dimension) *Let $\mathcal{F}$ be a family of functions mapping from $\mathcal{Z}$ to $\mathbb{R}$. Then, the pseudo dimension of $\mathcal{F}$, denoted by $\mathrm{Pdim}(\mathcal{F})$, is the size of the largest set shattered by $\mathcal{F}$.*

Definition 24 (VC dimension) *Let $\mathcal{F}$ be a family of functions mapping from $\mathcal{Z}$ to $\mathbb{R}$. Then, the Vapnik–Chervonenkis (VC) dimension of $\mathcal{F}$, denoted by $\mathrm{VCdim}(\mathcal{F})$, is the size of the largest set shattered by $\mathcal{F}$ with all threshold values being zero, i.e., $t_1=\ldots,=t_n=0$.*

B.2 Supporting Lemmas

The following lemma gives an upper bound for the covering number in terms of the pseudo-dimension.

Lemma 25 (Theorem 12.2 in Anthony and Bartlett (1999)) *Let $\mathcal{F}$ be a set of real functions from domain $\mathcal{Z}$ to the bounded interval $[0, B]$. Let $\delta > 0$ and suppose that $\mathcal{F}$ has finite pseudo-dimension* $\mathrm{Pdim}(\mathcal{F})$ *then*

$$\mathcal{N}_n(\delta, \mathcal{F}, \|\cdot\|_\infty) \leq \sum_{i=1}^{\mathrm{Pdim}(\mathcal{F})} \binom{n}{i} \left(\frac{B}{\delta}\right)^i,$$

which is less than $\{enB/(\delta \mathrm{Pdim}(\mathcal{F}))\}^{\mathrm{Pdim}(\mathcal{F})}$ *for* $n \geq \mathrm{Pdim}(\mathcal{F})$.

Appendix C. Additional simulation results

In this section, we include additional simulation results for the "Linear" model.

Table 11: Data is generated from the "Linear" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	Sample size	$n = 512$		$n = 2048$	
τ	Method	L_1	L_2^2	L_1	L_2^2
0.05	DQRP	0.395(0.219)	0.283(0.324)	0.282(0.165)	0.138(0.155)
	DQRP*	0.338(0.157)	0.174(0.153)	0.263(0.132)	0.106(0.099)
	DQR	**0.316(0.143)**	**0.141(0.134)**	**0.208(0.076)**	0.059(0.038)
	Kernel QR	0.277(0.213)	0.143(0.227)	0.149(0.167)	**0.056(0.172)**
	QR Forest	0.740(0.176)	1.479(1.828)	0.715(0.066)	1.250(1.048)
0.25	DQRP	0.126(0.051)	0.027(0.022)	0.084(0.037)	0.012(0.009)
	DQRP*	0.143(0.056)	0.033(0.023)	0.097(0.053)	0.015(0.015)
	DQR	**0.128(0.054)**	**0.023(0.021)**	0.093(0.034)	0.011(0.008)
	Kernel QR	0.149(0.058)	0.037(0.026)	**0.077(0.030)**	**0.010(0.008)**
	QR Forest	0.278(0.049)	0.125(0.043)	0.279(0.021)	0.127(0.019)
0.5	DQRP	0.118(0.054)	0.023(0.021)	0.084(0.043)	0.012(0.012)
	DQRP*	0.125(0.054)	0.025(0.018)	0.092(0.050)	0.013(0.014)
	DQR	**0.114(0.048)**	**0.018(0.015)**	0.082(0.028)	0.009(0.005)
	Kernel QR	0.131(0.042)	0.029(0.018)	**0.065(0.021)**	**0.007(0.004)**
	QR Forest	0.232(0.034)	0.087(0.025)	0.230(0.016)	0.085(0.011)
0.75	DQRP	0.179(0.099)	0.052(0.057)	0.113(0.075)	0.023(0.033)
	DQRP*	0.150(0.066)	0.035(0.027)	0.117(0.063)	0.022(0.021)
	DQR	**0.129(0.051)**	**0.023(0.017)**	0.094(0.033)	0.012(0.008)
	Kernel QR	0.146(0.044)	0.035(0.020)	**0.072(0.027)**	**0.009(0.007)**
	QR Forest	0.275(0.043)	0.124(0.041)	0.279(0.022)	0.127(0.020)
0.95	DQRP	0.386(0.211)	0.224(0.219)	0.252(0.151)	0.101(0.122)
	DQRP*	0.403(0.182)	0.217(0.164)	0.222(0.146)	0.082(0.104)
	DQR	0.305(0.142)	**0.132(0.111)**	0.216(0.075)	0.068(0.045)
	Kernel QR	**0.272(0.200)**	0.134(0.204)	**0.147(0.155)**	**0.051(0.149)**
	QR Forest	0.705(0.123)	0.997(0.500)	0.731(0.083)	1.438(1.531)

Table 12: Data is generated from the multivariate "Linear" model with training sample size $n = 512, 2048$ and the number of replications $R = 100$. The averaged L_1 and L_2^2 test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

	Sample size	$n = 512$		$n = 2048$	
τ	Method	L_1	L_2^2	L_1	L_2^2
0.05	DQRP	0.971(0.184)	1.377(0.433)	0.681(0.148)	0.660(0.229)
	DQRP*	0.897(0.173)	1.064(0.349)	0.681(0.155)	0.629(0.242)
	DQR	**0.703(0.129)**	**0.793(0.279)**	**0.609(0.094)**	**0.617(0.181)**
	Kernel QR	1.095(0.101)	1.610(0.251)	0.908(0.065)	1.019(0.124)
	QR Forest	0.939(0.100)	1.355(0.278)	0.648(0.052)	0.679(0.112)
0.25	DQRP	0.537(0.077)	0.551(0.152)	0.328(0.047)	0.204(0.053)
	DQRP*	**0.411(0.072)**	**0.296(0.114)**	**0.325(0.054)**	**0.182(0.055)**
	DQR	0.475(0.071)	0.385(0.111)	0.408(0.053)	0.290(0.069)
	Kernel QR	0.586(0.044)	0.580(0.090)	0.367(0.023)	0.230(0.028)
	QR Forest	0.739(0.037)	0.847(0.081)	0.514(0.018)	0.413(0.028)
0.5	DQRP	0.531(0.071)	0.523(0.137)	0.309(0.044)	0.178(0.047)
	DQRP*	**0.381(0.060)**	**0.252(0.086)**	**0.290(0.047)**	**0.144(0.043)**
	DQR	0.442(0.052)	0.337(0.078)	0.367(0.045)	0.239(0.055)
	Kernel QR	0.557(0.044)	0.534(0.089)	0.350(0.022)	0.210(0.026)
	QR Forest	0.658(0.031)	0.680(0.060)	0.458(0.016)	0.331(0.022)
0.75	DQRP	0.595(0.105)	0.653(0.244)	0.382(0.089)	0.262(0.112)
	DQRP*	**0.389(0.069)**	**0.256(0.095)**	**0.306(0.072)**	**0.159(0.070)**
	DQR	0.466(0.054)	0.371(0.082)	0.376(0.050)	0.258(0.070)
	Kernel QR	0.584(0.047)	0.580(0.095)	0.366(0.021)	0.227(0.026)
	QR Forest	0.741(0.040)	0.848(0.088)	0.515(0.019)	0.414(0.029)
0.95	DQRP	0.971(0.213)	1.419(0.480)	0.635(0.163)	0.596(0.256)
	DQRP*	0.838(0.173)	0.918(0.326)	0.774(0.175)	0.744(0.279)
	DQR	**0.635(0.120)**	**0.652(0.229)**	**0.485(0.081)**	**0.418(0.136)**
	Kernel QR	1.088(0.098)	1.596(0.246)	0.883(0.073)	0.975(0.132)
	QR Forest	0.962(0.117)	1.441(0.434)	0.649(0.049)	0.683(0.109)

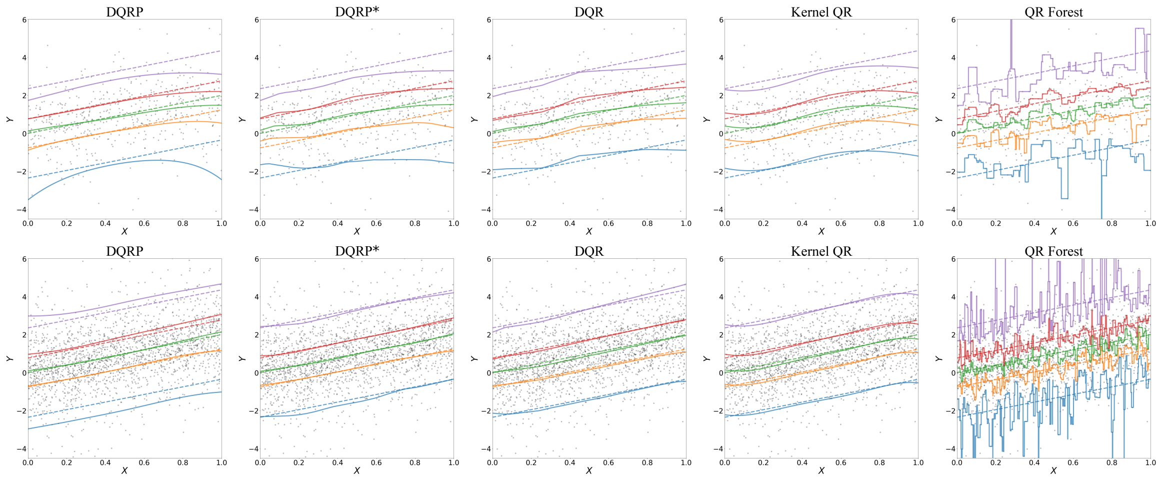


Figure 13: The fitted quantile curves by different methods under the univariate "Linear" model when $n = 512, 2048$. The training data is depicted as grey dots.The target quantile functions at the quantile levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple) are depicted as dashed curves, and the estimated quantile functions are represented by solid curves with the same color. From the top to the bottom, the rows correspond to the sample size $n = 512, 2048$. From the left to the right, the columns correspond to the methods DQRP, DQRP*, DQR, kernel QR and QR Forest.

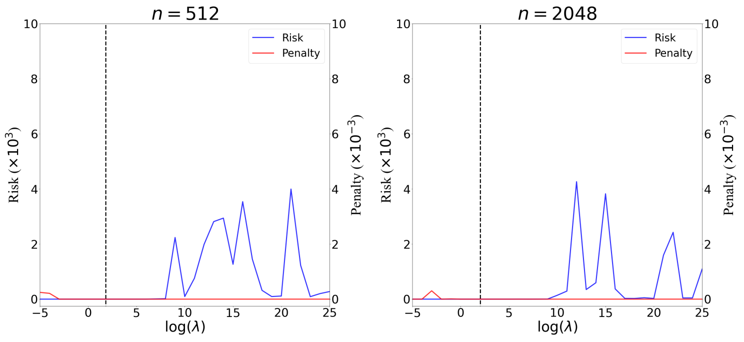


Figure 14: The value of risks and penalties under the univariate "Wave" model when $n = 512, 2048$. A vertical dashed line is depicted at the value $\lambda = \log(n)$ on x-axis in each figure.

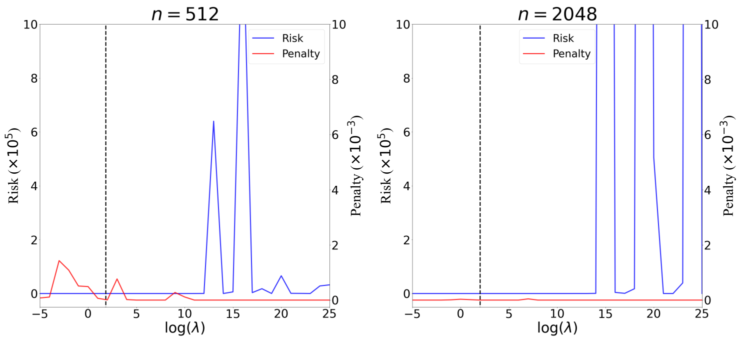


Figure 15: The value of risks and penalties under the multivariate single index model when $n = 512, 2048$ and $d = 8$. A vertical dashed line is depicted at the value $\lambda = \log(n)$ on x-axis in each figure.

References

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, 1999.

Laura K Bachrach, Trevor Hastie, May-Choo Wang, Balasubramanian Narasimhan, and Robert Marcus. Bone mineral acquisition in healthy asian, hispanic, black, and caucasian youth: a longitudinal study. *The Journal of Clinical Endocrinology & Metabolism*, 84(12): 4702–4712, 1999.

Thomas Bagby, Len Bos, and Norman Levenberg. Multivariate simultaneous approximation. *Constructive approximation*, 18(4):569–577, 2002.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20:Paper No. 63, 17, 2019. ISSN 1532-4435.

Benedikt Bauer and Michael Kohler. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Ann. of Statist.*, 47(4):2261–2285, 2019.

Alexandre Belloni and Victor Chernozhukov. ℓ1-penalized quantile regression in high-dimensional sparse models. *Ann. Statist.*, 39(1):82–130, 2011.

Alexandre Belloni, Victor Chernozhukov, Denis Chetverikov, and Ivan Fernandez-Val. Conditional quantile processes based on series or many regressors. *Journal of Econometrics*, 213(1):4–29, 2019.

Richard Blundell, Joel Horowitz, and Matthias Parey. Nonparametric estimation of a non-separable demand function under the Slutsky inequality restriction. *The Review of Economics and Statistics*, 99(2):291–304, 2017.

Howard D. Bondell, Brian J. Reich, and Huixia Wang. Noncrossing quantile regression curve estimation. *Biometrika*, 97(4):825–838, 08 2010.

Axel Brando, Joan Gimeno, Jose A Rodríguez-Serrano, and Jordi Vitrià. Deep non-crossing quantiles through the partial derivative. *arXiv preprint arXiv:2201.12848*, 2022.

Shih-Kang Chao, Stanislav Volgushev, and Guang Cheng. Quantile processes for semi and nonparametric regression. *Electronic Journal of Statistics*, 11, 04 2016.

Probal Chaudhuri. Nonparametric estimates of regression quantiles and their local bahadur representation. *The Annals of statistics*, 19(2):760–777, 1991.

Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Nonparametric regression on low-dimensional manifolds using deep relu networks. *arXiv preprint arXiv:1908.01842*, 2019.

Xiangyi Chen, Steven Z Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems*, 33: 13773–13782, 2020.

Victor Chernozhukov and Christian Hansen. An IV model of quantile treatment effects. *Econometrica*, 73(1):245–261, 2005.

Victor Chernozhukov, Guido W. Imbens, and Whitney K. Newey. Instrumental variable estimation of nonseparable models. *Journal of Econometrics*, 139(1):4–14, 2007.

Victor Chernozhukov, Iván Fernández-Val, and Alfred Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/40664520`.

Chenguang Duan, Yuling Jiao, Yanming Lai, Xiliang Lu, and Zhijian Yang. Convergence rate analysis for deep ritz method. *arXiv preprint arXiv:2103.13330*, 2021.

Max H. Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.

Olivier Fercoq and Pascal Bianchi. A coordinate-descent primal-dual algorithm with large step size and possibly nonseparable functions. *SIAM Journal on Optimization*, 29(1): 100–134, 2019.

Paul W Goldberg and Mark R Jerrum. Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18(2-3):131–148, 1995.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009.

Xuming He. Quantile curves without crossing. *The American Statistician*, 51(2):186–192, 1997.

Xuming He and Pin Ng. Quantile splines with several covariates. *Journal of Statistical Planning and Inference*, 75(2):343–352, 1999.

Xuming He and Peide Shi. Convergence rate of b-spline estimators of nonparametric conditional quantile functions. *Journaltitle of Nonparametric Statistics*, 3(3-4):299–308, 1994.

Lars Hörmander. *The Analysis of Linear Partial Differential Operators I: Distribution Theory and Fourier Analysis*. Springer, 2015.

Joel L. Horowitz and Sokbae Lee. Nonparametric instrumental variables estimation of a quantile regression model. *Econometrica*, 75(4):1191–1208, 2007.

Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 16–29. Springer, 2018.

Yuling Jiao, Guohao Shen, Yuanyuan Lin, and Jian Huang. Deep nonparametric regression on approximate manifolds: Nonasymptotic error bounds with polynomial prefactors. *The Annals of Statistics*, 51(2):691–716, 2023.

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

R. Koenker. *Quantile Regression.* Cambridge University Press, 2005.

R. Koenker and G. Bassett. Regression quantiles. *Econometrica*, 46:33–50, 1978.

R. Koenker, P. Ng, and S. Portnoy. Quantile smoothing splines. *Biometrica*, 81:673–680, 1994.

Michael Kohler, Adam Krzyzak, and Sophie Langer. Estimation of a function of low local dimensionality by deep neural networks. *arXiv preprint arXiv:1908.11140*, 2019.

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

Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *Proceedings on Privacy Enhancing Technologies*, 2021(1), 2021.

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*, 2019a.

Bo Li, Shanshan Tang, and Haijun Yu. Powernet: Efficient representations of polynomials and smooth functions by deep neural networks with rectified power units. *arXiv preprint arXiv:1909.05136*, 2019b.

Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021a.

Yiping Lu, Haoxuan Chen, Jianfeng Lu, Lexing Ying, and Jose Blanchet. Machine learning for elliptic pdes: Fast rate generalization bound, neural scaling law and minimax optimality. *arXiv preprint arXiv:2110.06897*, 2021b.

Enno Mammen. Nonparametric regression under qualitative smoothness assumptions. *The Annals of Statistics*, 19(2):741 – 759, 1991.

Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of Machine Learning Research*, 7(6), 2006.

Alexandre Mösching and Lutz Dümbgen. Monotone least squares and isotonic quantiles. *Electronic journal of statistics*, 14(1):24–49, 2020.

Ryumei Nakada and Masaaki Imaizumi. Adaptive approximation and estimation of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, 21: 1–38, 2020.

Oscar Hernan Madrid Padilla and Sabyasachi Chatterjee. Risk bounds for quantile trend filtering. *arXiv preprint arXiv:2007.07472v5*, 2021.

Oscar Hernan Madrid Padilla, Wesley Tansey, and Yanzhen Chen. Quantile regression with ReLU networks: Estimators and minimax rates. *Journal of Machine Learning Research*, 23(247):1–42, 2022.

Maxime Sangnier, Olivier Fercoq, and Florence d'Alché Buc. Joint quantile regression in vector-valued RKHSs. *Advances in Neural Information Processing Systems*, 29:3693–3701, 2016.

Jonathan Scarlett and Volkan Cevher. An introductory guide to Fano's inequality with applications in statistical estimation. *arXiv preprint arXiv:1901.00555*, 2019.

Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. *Annals of Statistics*, 48(4):1875–1897, 2020.

Guohao Shen, Yuling Jiao, Yuanyuan Lin, Joel L Horowitz, and Jian Huang. Deep quantile regression: mitigating the curse of dimensionality through composition. *arXiv preprint arXiv:2107.04907*, 2021.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Nonlinear approximation via compositions. *Neural Networks*, 119:74–84, 2019.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation characterized by number of neurons. *Commun. Comput. Phys.*, 28(5):1768–1811, 2020.

Ingo Steinwart. How to compare different loss functions and their risks. *Constructive Approximation*, 26(2):225–287, 2007.

Charles J Stone. Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, pages 1040–1053, 1982.

Ichiro Takeuchi, Quoc V. Le, Timothy D. Sears, and Alexander J. Smola. Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7(45):1231–1264, 2006.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation.* Springer Series in Statistics, Springer, 2008.

Lan Wang, Yichao Wu, and Runze Li. Quantile regression for analyzing heterogeneity in ultra-high dimension. *Journal of the American Statistical Association*, 107:214–222, 2012.

Halbert White. Nonparametric estimation of conditional quantiles using neural networks. In *Computing Science and Statistics*, pages 190–199. Springer, 1992.

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

Dmitry Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In *Conference on Learning Theory*, pages 639–649. PMLR, 2018.

Qi Zheng, Limin Peng, and Xuming He. Globally adaptive quantile regression with ultra-high dimensional data. *Ann. Statist.*, 43(5):2225–2258, 2015.

Qixian Zhong and Jane-Ling Wang. Neural networks for partially linear quantile regression. *arXiv preprint arXiv:2106.06225*, 2021.