Generative Quantile Regression with Variability Penalty

Shijie Wang[*]

Department of Statistics, University of South Carolina, Columbia, SC 29208

and

Minsuk Shin

Gauss Labs, Palo Alto, CA 94301

and

Ray Bai

Department of Statistics, University of South Carolina, Columbia, SC 29208

February 29, 2024

Abstract

Quantile regression and conditional density estimation can reveal structure that is missed by mean regression, such as multimodality and skewness. In this paper, we introduce a deep learning generative model for joint quantile estimation called Penalized Generative Quantile Regression (PGQR). Our approach simultaneously generates samples from many random quantile levels, allowing us to infer the conditional distribution of a response variable given a set of covariates. Our method employs a novel variability penalty to avoid the problem of vanishing variability, or memorization, in deep generative models. Further, we introduce a new family of partial monotonic neural networks (PMNN) to circumvent the problem of crossing quantile curves. A major benefit of PGQR is that it can be fit using a single optimization, thus bypassing the need to repeatedly train the model at multiple quantile levels or use computationally expensive cross-validation to tune the penalty parameter. We illustrate the efficacy of PGQR through extensive simulation studies and analysis of real datasets. Code to implement our method is available at `https://github.com/shijiew97/PGQR`.

Keywords: conditional quantile, deep generative model, generative learning, joint quantile model, neural networks, nonparametric quantile regression

[*]The authors gratefully acknowledge financial support from the National Science Foundation (Grant no. NSF DMS-2015528).

1 Introduction

Quantile regression is a popular alternative to classical mean regression (Koenker and Bassett Jr, 1982). For a response variable $Y \in \mathbb{R}$ and covariates $\boldsymbol{X} = (X_1, X_2, \dots, X_p)^\top \in \mathbb{R}^p$, we define the τ-th conditional quantile, $\tau \in (0,1)$, of Y given $\boldsymbol{X}$ as

$$Q_{Y|\boldsymbol{X}}(\tau) = \inf\{y : F_{Y|\boldsymbol{X}}(y) \geq \tau\}, \tag{1}$$

where $F_{Y|\boldsymbol{X}}$ is the conditional cumulative distribution function of Y given $\boldsymbol{X}$. For a fixed quantile level $\tau \in (0,1)$, linear quantile regression aims to model (1) as $Q_{Y|\boldsymbol{X}}(\tau) = \boldsymbol{X}^\top \boldsymbol{\beta}_\tau$. Linear quantile regression is less sensitive to outliers than least squares regression and is more robust when the assumption of independent and identically distributed (iid) Gaussian residual errors is violated (Koenker and Hallock, 2001; Koenker et al., 2017). Thus, it is widely used for modeling heterogeneous covariate-response associations. To reveal complex nonlinear structures, *nonparametric* quantile regression has also been proposed (Chaudhuri and Loh, 2002; Li et al., 2021; Koenker et al., 1994). For some function class $\mathcal{F}$, nonparametric quantile regression aims to estimate $f(\boldsymbol{X}, \tau) \in \mathcal{F}$ for $Q_{Y|\boldsymbol{X}}(\tau) = f(\boldsymbol{X}, \tau)$ at a given quantile level τ. While quantile regression has traditionally focused on estimating $f(\boldsymbol{X}, \tau)$ at a single τ-th quantile, *joint* quantile regression aims to estimate multiple quantile levels *simultaneously* for a set of $K \geq 2$ quantiles $\{\tau_1, \tau_2, \dots, \tau_K\}$ (Zou and Yuan, 2008; Jiang et al., 2012; Xu et al., 2017).

Despite its flexibility, nonparametric quantile regression carries the risk that the estimated curves at different quantile levels might *cross* each other. For instance, the estimate of the 95th conditional quantile of Y_i given covariates $\boldsymbol{X}_i$ might be smaller than the estimate of the 90th conditional quantile. This is known as the *crossing quantile* phenomenon. The left two panels of Figure 3 illustrate the crossing problem. When quantile curves cross each other, the quantile estimates violate the laws of probability and are not reasonable. To tackle this issue, many approaches have been proposed, such as constraining the model space (Takeuchi et al., 2006; Sangnier et al., 2016; Moon et al., 2021) or constraining the model parameters (Meinshausen, 2006; Cannon, 2018).

Recently in the area of deep learning, neural networks have also been applied to nonparametric quantile regression. Zhong and Wang (2023) introduced a semiparametric quantile regression method where the covariates of primary interest are modeled linearly, while all other covariates are modeled nonparametrically by a neural network. Shen et al. (2021) proposed a deep quantile regression (DQR) estimator to approximate the target conditional quantile function with compositional structure. Under the assumption that the conditional quantile function is a composition of low-dimentional functions, DQR is shown to achieve the minimax optimal convergence rate.

Researchers have also used neural networks to learn *multiple* quantiles using composite quantile loss functions. Cannon (2018) proposed the monotone composite quantile regression neural network (MCQRNN) and achieved non-crossing quantiles by imposing monotone increasing bias parameters for the neural network. Moon et al. (2021) recently introduced noncrossing multiple quantile regression with neural networks (NMQN) and a scalable, efficient ℓ_1-penalization algorithm to fit NMQN. However, existing neural network-based approaches for joint quantile estimation are restricted at a prespecified quantile set. If only several quantiles (e.g. the interquartiles $\tau \in \{0.25, 0.5, 0.75\}$) are of interest, then these methods are adequate to produce desirable inference. Otherwise, one typically has to refit the model at new quantiles *or* specify a large enough quantile candidate set in order to infer the full conditional density $p(Y \mid \boldsymbol{X})$. Dabney et al. (2018) introduced the implicit quantile network (IQN), which takes a grid of quantile levels as inputs and approximates the conditional density at *any* new quantile level. Unfortunately, IQN *cannot* guarantee that quantile functions do not cross each other. More recently, Shen et al. (2022) constructed a quantile regression process similar to IQN but which avoids crossing quantiles by imposing a positivity constraint on the first derivatives.

Joint quantile regression is closely related to the problem of *conditional density estimation* (CDE) of $p(Y \mid \boldsymbol{X})$. Apart from traditional CDE methods that estimate the unknown probability density curve (Izbicki and Lee, 2017; Pospisil and Lee, 2019), several deep *generative* approaches besides IQN have also been proposed. Zhou et al. (2023) introduced the generative conditional distribution sampler (GCDS), while Liu et al. (2021) introduced the Wasserstein generative conditional sampler (WGCS) for CDE. GCDS and WGCS employ the idea of generative adversarial networks (GANs) (Goodfellow et al., 2014) to *generate samples* from the conditional distribution $p(Y \mid \boldsymbol{X})$. In these deep generative models, random noise inputs are used to learn and generate samples from the target distribution. However, a well-known problem with deep generative networks is that the generator may simply memorize the training samples instead of generalizing to new test data (Arpit et al., 2017; van den Burg and Williams, 2021). When *memorization* occurs, the random noise variable no longer reflects any variability, and the predicted density $p(Y \mid \boldsymbol{X})$ approaches a point mass. We refer to this phenomenon as *vanishing variability.*

To conduct joint quantile estimation as well as conditional density estimation, we propose a new deep generative approach called penalized generative quantile regression (PGQR). PGQR employs a novel variability penalty to avoid vanishing variability. We further guarantee the monotonicity (i.e. non-crossing) of the estimated quantile functions from PGQR by designing a *new* family of partial monotonic neural networks (PMNNs). The PMNN architecture ensures partial monotonicity with respect to quantile inputs, while retaining the expressiveness of neural networks.

The performance of PGQR depends crucially on carefully choosing the variability penalty parameter λ, where λ lies in the range of $[0, \lambda_{\max}]$ and $\lambda_{\max}$ should be determined before optimization. In all of our analyses, we chose $\lambda_{\max} = \exp(1)$. Unfortunately, common tuning procedures such as cross-validation are impractical for deep learning, since this would involve repetitive evaluations of the network for different training sets and choices of λ. Inspired by Shin et al. (2022), we construct our deep generative network in such a way that λ is included as an additional random input. This way, only a *single* optimization is needed to learn the neural network parameters, and then it is effortless to generate samples from a set of candidate values for λ. We propose a criterion for selecting the optimal choice of λ to generate the final desired samples from $p(Y \mid \boldsymbol{X})$.

Our main contributions can be summarized as follows:

1. We propose a deep generative approach for joint quantile regression and conditional density estimation called *penalized generative quantile regression.* PGQR simultaneously generates samples from *multiple* random quantile levels, thus precluding the need to refit the model at different quantiles.

2. We introduce a novel variability penalty to avoid the vanishing variability phenomenon in deep generative models and apply this regularization technique to PGQR.

3. We construct a new family of partial monotonic neural networks (PMNNs) to circumvent the problem of crossing quantile curves.

4. To facilitate scalable computation for PGQR and bypass computationally expensive cross-validation for tuning the penalty parameter, we devise a strategy that allows PGQR to be implemented using only a *single* optimization.

The rest of the article is structured as follows. Section 2 introduces the generative quantile regression framework and our variability penalty. In Section 3, we introduce the PMNN family for preventing quantile curves from crossing each other. Section 4 discusses scalable computation for PGQR, namely how to tune the variability penalty parameter with only single-model training. In Section 5, we demonstrate the utility of PGQR through simulation studies and analyses of additional real datasets. Section 6 concludes the paper with some discussion and directions for future research. All proofs of propositions can be found in Section C of the Supplementary Material.

2 Penalized Generative Quantile Regression

2.1 Generative Quantile Regression

Before introducing PGQR, we first introduce our framework for generative quantile regression (without variability penalty). Given n training samples $(\boldsymbol{X}_1, Y_1), (\boldsymbol{X}_2, Y_2), \ldots, (\boldsymbol{X}_n, Y_n)$ and quantile level $\tau \in (0,1)$, nonparametric quantile regression minimizes the empirical quantile loss,

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \rho_\tau \left(Y_i - f(\boldsymbol{X}_i, \tau) \right), \tag{2}$$

where $\rho_\tau(u) = u(\tau - I(u < 0))$ is the check function, and $\mathcal{F}$ is a function class such as a reproducing kernel Hilbert space or a family of neural networks. Neural networks are a particularly attractive way to model the quantile function $f(\boldsymbol{X}, \tau)$ in (2), because they are universal approximators for any Lebesgue integrable function (Lu et al., 2017).

In this paper, we model the quantile function in (2) using deep neural networks (DNNs). We define a DNN as a neural network with at least two hidden layers and a large number of hidden neurons. We refer to Emmert-Streib et al. (2020) for a comprehensive review of DNNs. Despite the universal approximation properties of DNNs, we must *also* take care to ensure that our estimated quantile functions do *not* cross each other. Therefore, we have to consider a wide enough *monotonic* function class $\mathcal{G}^m$ to cover the true $Q_{Y|\boldsymbol{X}}(\cdot)$. We formally introduce this class $\mathcal{G}^m$ in Section 3.

Let G denote the constructed DNN, which is defined as a feature map function $\{G \in \mathcal{G}^m : \mathbb{R}^{p+1} \mapsto \mathbb{R}^1\}$ that takes $(\boldsymbol{X}_i, \tau)$ as input and generates the conditional quantile for Y_i given $\boldsymbol{X}_i$ at level τ as output. We refer to this generative framework as *Generative Quantile Regression* (GQR).

The optimization problem for GQR is

$$\widehat{G} = \underset{G \in \mathcal{G}^m}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\tau \big\{ \rho_\tau \big(Y_i - G(\boldsymbol{X}_i, \tau) \big) \big\}, \tag{3}$$

where $\widehat{G}$ denotes the estimated quantile function with optimized parameters (i.e. the weights and biases) of the DNN. Note that optimizing the integrative loss $\mathbb{E}_\tau\{\cdot\}$ over τ in (3) is justified by Proposition 2.1 introduced in the next section (i.e. we set $\lambda = 0$ in Proposition 2.1).

In order to solve (3), we can use stochastic gradient descent (SGD) with mini-batching (Emmert-Streib et al., 2020). For each mini-batch evaluation, $\boldsymbol{X}_i$ is paired with a quantile level τ sampled from Uniform$(0, 1)$. Consequently, if there are M mini-batches, the expectation in (3) can be approximated by a Monte Carlo average of the random τ's, i.e. we approximate $\mathbb{E}_\tau \big\{ \rho_\tau \big(Y_i - G(\boldsymbol{X}_i, \tau) \big) \big\}$ with $M^{-1} \sum_{k=1}^{M} \big\{ \rho_{\tau_k} \big(Y_i - G(\boldsymbol{X}_i, \tau_k) \big) \big\}$. Once we have solved (3), it is straightforward to use $\widehat{G}$ to generate *new* samples $\widehat{G}(\boldsymbol{X}, \xi_k), k = 1, 2, \ldots, b$, at various quantile levels $\boldsymbol{\xi} = \{\xi_1, \xi_2, \ldots, \xi_b\}$, where the ξ's are random Uniform$(0, 1)$ noise inputs. Provided that b is large enough, the generated samples at $\boldsymbol{\xi} \in (0, 1)^b$ can be used to reconstruct the full conditional density $p(Y \mid \boldsymbol{X})$.

As discussed in Section 1, there are several other deep generative models (Zhou et al., 2023; Liu et al., 2021) for generating samples from $p(Y \mid \boldsymbol{X})$. These generative approaches also take random noise z as an input (typically $z \sim \mathcal{N}(0, 1)$) to reflect variability, but there is no statistical meaning for z. In contrast, the random noise τ in GQR (3) has a clear interpretation as a quantile level $\tau \in (0, 1)$. Although GQR and these other deep generative approaches are promising approaches for conditional sampling, these methods are all unfortunately prone to memorization of the training data. When this occurs, the random noise does not generate *any* variability. To remedy this, we now introduce our *variability penalty* in conjunction with our GQR loss function (3).

2.2 Variability Penalty for GQR

Overparameterization in neural networks occurs when the number of learnable parameters (i.e. the weights and biases) is much greater than the number of training samples. In practice, it is common for a DNN to be overparameterized. Empirical and theoretical studies have shown that overparameterization improves the generalization and robustness of DNNs, since it greatly enhances the representation power of the DNN and simplifies the optimization landscape (Allen-Zhu et al., 2019; Zhang et al., 2021; Soltanolkotabi et al., 2019; Montanari and Zhong, 2022). Unfortunately, when a DNN is overparameterized to capture the underlying structure in the training data, the random noise in GQR is likely to reflect no variability, no matter what value it inputs. This is a common problem in deep generative models (Arpit et al., 2017; Arora et al., 2017; van den Burg and Williams, 2021). We refer this phenomenon as *vanishing variability.* To be more specific, let $G(\cdot, z)$ be the generator function constructed by a DNN, where z is a random noise variable following some reference distribution such as a standard Gaussian or a standard uniform. The vanishing variability phenomenon occurs when

$$\widehat{G}(\boldsymbol{X}_i, z) = Y_i, \;\; i = 1, ..., n. \tag{4}$$

In other words, there is no variability when inputting different random noise z, generating almost surely a discrete point mass at the training data. Given a *new* feature vector $\boldsymbol{X}_{\text{new}}$, $G(\boldsymbol{X}_{\text{new}}, z)$ can also only generate one novel sample from the true data distribution because of vanishing variability.

Since GQR takes the training data $\boldsymbol{X} \in \mathbb{R}^p$ as input and the target quantile estimate lies in $\mathbb{R}^1$, the weights in the DNN associated with $\boldsymbol{X} \in \mathbb{R}^p$ are very likely to overwhelm those associated with the noise input τ. As a result, GQR is very prone to encountering vanishing variability, as are other generative approaches with multidimensional features. From another point of view, we can see that due to the nonnegativity of the check function, the GQR loss (3) achieves a minimum value of zero when we have vanishing variability (4).

To remedy this problem, we propose a new regularization term that encourages the network to have *more* variability when vanishing variability occurs. Given a set of features $\boldsymbol{X} \in \mathbb{R}^p$, the proposed *variability penalty* is formulated as follows:

$$\text{pen}_{\lambda,\alpha}(G(\boldsymbol{X}, \tau), G(\boldsymbol{X}, \tau')) = -\lambda \log \left\{ \|G(\boldsymbol{X}, \tau) - G(\boldsymbol{X}, \tau')\|_1 + 1/\alpha \right\}, \tag{5}$$

where $\lambda \geq 0$ is the hyperparameter controlling the degree of penalization, $\alpha > 0$ is a fixed hyper-parameter, and $\tau, \tau' \sim \text{Uniform}(0, 1)$. The addition of $1/\alpha$ inside the logarithmic term of (5) is mainly to ensure that the penalty function is always well-defined (i.e. the quantity inside of $\log\{\cdot\}$ of (5) cannot equal zero). Our sensitivity analysis in Section D of the Supplementary Material shows that our method is not usually sensitive to the choice of α. We find that fixing $\alpha = 1$ or

$\alpha = 5$ works well in practice.

With the addition of the variability penalty (5) to our GQR loss function (3), our *penalized* GQR (PGQR) method solves the optimization,

$$\widehat{G} = \underset{G \in \mathcal{G}^m}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left[\mathbb{E}_{\tau} \left\{ \rho_{\tau}\big(Y_i - G(\boldsymbol{X}_i, \tau)\big) \right\} + \mathbb{E}_{\tau,\tau'} \left\{ \text{pen}_{\lambda,\alpha} \left(G(\boldsymbol{X}_i, \tau), G(\boldsymbol{X}_i, \tau') \right) \right\} \right], \tag{6}$$

where $\mathcal{G}^m$ denotes the PMNN family introduced in Section 3, and τ and τ' independently follow a uniform distribution on $(0, 1)$. The expectation $\mathbb{E}_{\tau,\tau'}$ is again approximated by a Monte Carlo average. Note that when $\lambda = 0$, the PGQR loss (6) reduces to the (non-penalized) GQR loss (3).

The next proposition states that an estimated generator function $\widehat{G}(\boldsymbol{X}, \tau)$ under the PGQR objective (6) is equivalent to a neural network estimator $\widehat{g}_{\tau}(\boldsymbol{X})$ based on the individual (non-integrative) quantile loss function, provided that the family $\mathcal{G}^m$ in (6) is large enough.

Proposition 2.1 (equivalence of $\widehat{g}_{\tau}(\boldsymbol{X})$ and $\widehat{G}(\boldsymbol{X}, \tau)$)**.** *For fixed $\lambda \geq 0$ and $\alpha > 0$, let* $\widehat{g}_{\tau}(\boldsymbol{X}) = \operatorname{argmin}_{g \in \mathcal{H}} \sum_{i=1}^{n} \left[\rho_{\tau}(Y_i - g(\boldsymbol{X}_i)) + \mathbb{E}_{\tau,\tau'}\{\text{pen}_{\lambda,\alpha}(g(\boldsymbol{X}_i, \tau), g(\boldsymbol{X}_i, \tau'))\} \right]$*, for a class of neural networks $\mathcal{H}$, and $\tau, \tau' \overset{iid}{\sim}$ Uniform$(0, 1)$. Consider a class of generator functions $\mathcal{G}$, where $\{G \in \mathcal{G} : \mathbb{R}^p \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}\}$. Assume that for all $\boldsymbol{X} \in \mathcal{X} \subset \mathbb{R}^p$ and quantile levels $\tau \in (0, 1)$, there exists $G \in \mathcal{G}$ such that the target neural network $\widehat{g}_{\tau}(\boldsymbol{X})$ can be represented by a hyper-network $G(\boldsymbol{X}, \tau)$. Then, for $i = 1, \ldots, n$,*

$$\widehat{g}_{\tau}(\boldsymbol{X}_i) = \widehat{G}(\boldsymbol{X}_i, \tau) \ \ a.s.,$$

with respect to the probability law related to $\mathbb{E}_{\tau}$, where $\widehat{G}$ is a solution to (6)*.*

Proposition 2.1 justifies optimizing the *integrative* quantile loss over τ in the PGQR objective (6). The next proposition justifies adding the variability penalty (5) to the GQR loss (3) by showing that there exists $\lambda > 0$ so that memorization of the training data does *not* occur under PGQR.

Proposition 2.2 (PGQR does not memorize the training data)**.** *Suppose that $\alpha > 0$ is fixed in the variability penalty* (5)*. Denote any minimizer of the PGQR objective function* (6) *by $\widehat{G}$. Then, there exists $\lambda > 0$ such that*

$$\widehat{G}(\boldsymbol{X}_i, \tau) \neq Y_i \ \textit{for at least one}\ i = 1, \ldots, n.$$

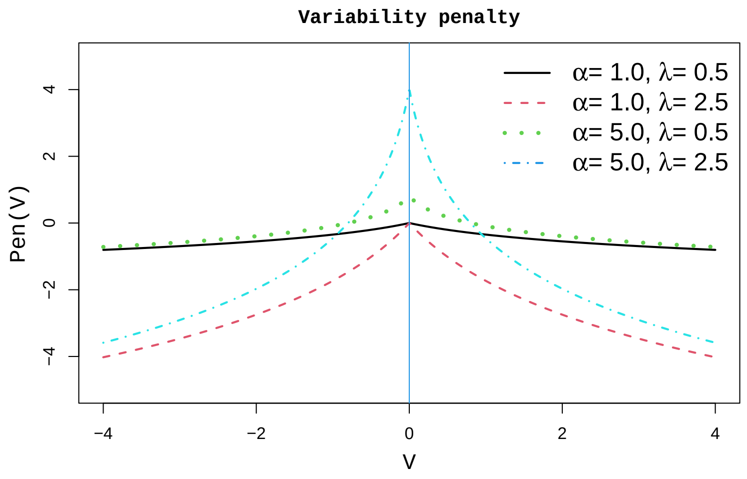

Figure 1: A plot of the penalty function $\text{pen}_{\lambda,\alpha}(V) = -\lambda \log(|V| + 1/\alpha)$.

We now provide some intuition into the variability penalty (5). We can quantify the amount of variability in the network by $V := G(\boldsymbol{X}, \tau) - G(\boldsymbol{X}, \tau')$. Clearly, $V = 0$ when vanishing variability (4) occurs. To avoid data memorization, we should thus penalize V *more heavily* when $V \approx 0$, with the maximum amount of penalization being applied when $V = 0$. Figure 1 plots the variability penalty, $\text{pen}_{\lambda,\alpha}(V) = -\lambda \log(|V| + 1/\alpha)$, for several pairs of (α, λ). Figure 1 shows that $\text{pen}_{\lambda,\alpha}(V)$ is sharply peaked at zero and strictly convex decreasing in $|V|$. Thus, maximum penalization occurs when $V = 0$, and there is less penalization for larger $|V|$. As λ increases, $\text{pen}_{\lambda,\alpha}(V)$ also increases for all values of $V \in (-\infty, \infty)$, indicating that larger values of λ lead to more penalization.

Our penalty function takes a specific form (5). However, any other penalty on $G(\boldsymbol{X}, \tau) - G(\boldsymbol{X}, \tau')$ for $\boldsymbol{X} \in \mathcal{X}$ that has a similar shape as the PGQR penalty (i.e. sharply peaked around zero, as in Figure 1) would also conceivably encourage the network to have greater variability. Another way to control the variability is to directly penalize the empirical variance $s^2 = (n-1)^{-1} \sum_{i=1}^{n} [G(\boldsymbol{X}_i, \tau) - \bar{G}]^2$, where $\bar{G} = n^{-1} \sum_{i=1}^{n} G(\boldsymbol{X}_i, \tau)$, or the empirical standard deviation $s = \sqrt{s^2}$. However, penalties on s^2 or s do *not* have an additive form and individual penalty terms depend on each other, and therefore, these are *not* conducive to SGD-based optimization.

We now pause to highlight the novelty of our constructed variability penalty. In many other nonparametric models, e.g. those based on nonparametric smoothing, a roughness penalty is often added to an empirical loss function in order to control the smoothness of the function or density estimate (Green and Silverman, 1994). These roughness penalties encourage greater smoothness

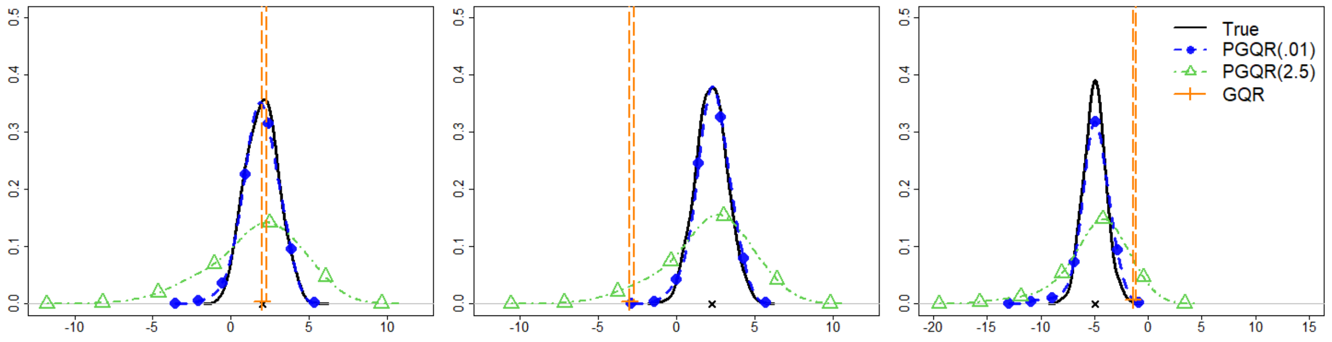

Figure 2: Plots of the estimated conditional densities $p(Y \mid \boldsymbol{X}_{\text{test}})$ for three different test observations under non-penalized GQR and PGQR with two different choices of $\lambda \in \{0.01, 2.5\}$. $\lambda^\star = 0.01$ is the optimal λ chosen by our tuning parameter selection method in Section 4.2.

of the target function in the spirit of reducing variance in the bias-variance trade-off. In contrast, our variability penalty (5) directly penalizes generators with *too small* variance. By adding the penalty to the GQR loss (3), we are encouraging the estimated network to have *more* variability.

To illustrate how PGQR prevents vanishing variability, we carry out a simple simulation study under the model, $Y_i = \boldsymbol{X}_i^\top \boldsymbol{\beta} + \epsilon_i, i = 1, \ldots, n$, where $\boldsymbol{X}_i \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{20})$, $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, 1)$, and the coefficients in $\boldsymbol{\beta}$ are equispaced over $[-2, 2]$. We used 2000 samples to train GQR (3) and PGQR (6), and an additional 200 validation samples were used for tuning parameter selection of λ (described in Section 4.2). To train the generative network, we fixed $\alpha = 1$ and tuned λ from a set of 100 equispaced values between 0 and 2.5. We then generated 1000 samples $\{\widehat{G}(\boldsymbol{X}_{\text{test}}, \xi_k, \lambda)\}_{k=1}^{1000}$, $\xi_k \overset{iid}{\sim} \text{Uniform}(0, 1)$, from the estimated conditional density of $p(Y \mid \boldsymbol{X}_{\text{test}})$ for three different choices of out-of-sample test data for $\boldsymbol{X}_{\text{test}}$. In our simulation, the tuning parameter selection procedure introduced in Section 4.2 chose an optimal λ of $\lambda^\star = 0.01$.

As shown in Figure 2, the *non*-penalized GQR model introduced in Section 2.1 (dashed orange line) suffers from vanishing variability. In particular, GQR is constructed by a neural network of three hidden layers, each layer having 1000 hidden neurons. Namely, GQR generates values near the true test sample Y_{test} almost surely, despite the fact that we used 1000 different inputs for ξ to generate the $\widehat{G}$ samples. In contrast, the PGQR model with optimal $\lambda^\star = 0.01$ (solid blue line with filled circles) approximates the true conditional density (solid black line) very closely, capturing the Gaussian shape and matching the true underlying variance.

On the other hand, Figure 2 also illustrates that if λ is chosen to be *too* large in PGQR (6), then the subsequent approximated conditional density will exhibit larger variance than it should. In particular, when $\lambda = 2.5$, Figure 2 shows that PGQR (dashed green line with hollow triangles) *overestimates* the true conditional variance for Y given $\boldsymbol{X}$. This demonstrates that the choice of λ in the variability penalty (1) plays a very crucial role in the practical performance of PGQR. In Section 4.2, we describe how to select an "optimal" λ so that PGQR neither underestimates *nor* overestimates the true conditional variance. Our method for tuning λ also requires only a *single* optimization, making it an attractive and scalable alternative to cross-validation.

As discussed earlier, the quantile loss function (3) is equal to zero when the estimated quantiles are exactly equal to the observed responses Y's. Therefore, if a complex model like a DNN is used for $G(\boldsymbol{X}, \tau)$ in (3), the estimated model is more prone to interpolate the observed responses. It is natural to wonder whether picking a simpler neural network (i.e. one with fewer hidden layers and/or neurons) will remedy the vanishing variability phenomenon. In Section D of the Supplement, we show that tuning the model complexity for *non*-penalized GQR (e.g. choosing a simpler neural network with one or two hidden layers and five or 50 neurons) does *not* completely avoid vanishing variability. A simpler neural network structure also invariably loses representation power. Our PGQR model (6) allows us to realize the benefits of overparameterization (Allen-Zhu et al., 2019; Zhang et al., 2021; Soltanolkotabi et al., 2019; Montanari and Zhong, 2022) while *simultaneously* avoiding vanishing variability.

3 Partial Monotonic Neural Network

Without any constraints on the network, PGQR is just as prone as other unconstrained nonparametric quantile regression models to suffer from crossing quantiles. For a fixed data point $\boldsymbol{X}_i$ and an estimated network $\widehat{G}$, the *crossing problem* occurs when

$$\widehat{G}(\boldsymbol{X}_i, \tau_1) > \widehat{G}(\boldsymbol{X}_i, \tau_2) \text{ when } 0 < \tau_1 < \tau_2 < 1. \tag{7}$$

In the neural network literature, one popular way to address the crossing problem is to add a penalty term to the loss such as $\max(0, -\partial G(\boldsymbol{X}_i; \tau)/\partial \tau)$ (Tagasovska and Lopez-Paz, 2018; Liu et al., 2020; Shen et al., 2022). Through regularization, the network is encouraged to have larger partial derivatives with respect to τ. Another natural way to construct a monotonic neural network (MNN) is by restricting the weights of network to be nonnegative through a transformation or through weight clipping (Zhang and Zhang, 1999; Daniels and Velikova, 2010; Mikulincer and Reichman, 2022). Because *all* the weights are constrained to be nonnegative, this class of MNNs might require longer optimization (i.e. more epochs in SGD) and/or a large amount of hidden neurons to ensure the network's final expressiveness.

Instead of constraining *all* the weights in the neural network, we make a simple modification to the MNN architecture which we call the *partial* monotonic neural network (PMNN). The PMNN family consists of *two* feedforward neural networks (FNN) as sub-networks which divide the input into two segments. The first sub-network is a weight-constrained quantile network $\mathbb{R}^1 \mapsto \mathbb{R}^h$ where the only input is the quantile level τ and h denotes the number of hidden neurons. The FNN structure for one hidden layer in this sub-network is $g(\tau) = \sigma(\boldsymbol{U}_{\text{pos}}\tau + \boldsymbol{b})$, where $\boldsymbol{U}_{\text{pos}}$ is the weights matrix of *nonnegative* weights, $\boldsymbol{b}$ is the bias matrix, and σ is the activation function, such as the rectified linear unit (ReLU) function $\sigma(x) = \max\{0, x\}$. The K_1-layer constrained sub-network g_c can then be constructed as $g_c(\tau) = g_K \circ \cdots \circ g_1$.

The second sub-network is a K_2-layer *unconstrained* network $g_{uc} : \mathbb{R}^p \mapsto \mathbb{R}^h$, taking the data $\boldsymbol{X}$ as inputs. The structure of one hidden layer in this sub-network is $g'(\boldsymbol{X}) = \sigma(\boldsymbol{U}\boldsymbol{X} + \boldsymbol{b})$, where $\boldsymbol{U}$ is an *unconstrained* weights matrix. We construct g_{uc} as $g_{uc}(\boldsymbol{X}) = g'_{K_2} \circ \cdots \circ g'_1$. Finally, we construct a single weight-constrained connection layer $f : \mathbb{R}^h \mapsto \mathbb{R}^1$ that connects the two sub-networks to quantile estimators,

$$G(\boldsymbol{X}, \tau) = f \ \circ (g_c + g_{uc}). \tag{8}$$

With our PMNN architecture, the monotonicity of $G(\cdot, \tau)$ with respect to τ is guaranteed by the positive weights in the quantile sub-network g_c and the final layer f. At the same time, since the data sub-network g_{uc} is unconstrained in its weights, g_{uc} can learn the features of the training data well. In particular, the PMNN family is more flexible in its ability to learn features of the data and is easier to optimize than MNN because of this unconstrained sub-network. We use our proposed PMNN architecture as the family $\mathcal{G}^m$ over which to optimize the PGQR objective for all of the simulation studies and real data analyses in this manuscript.

To investigate the performance of our proposed PMNN family of neural networks, we fit the

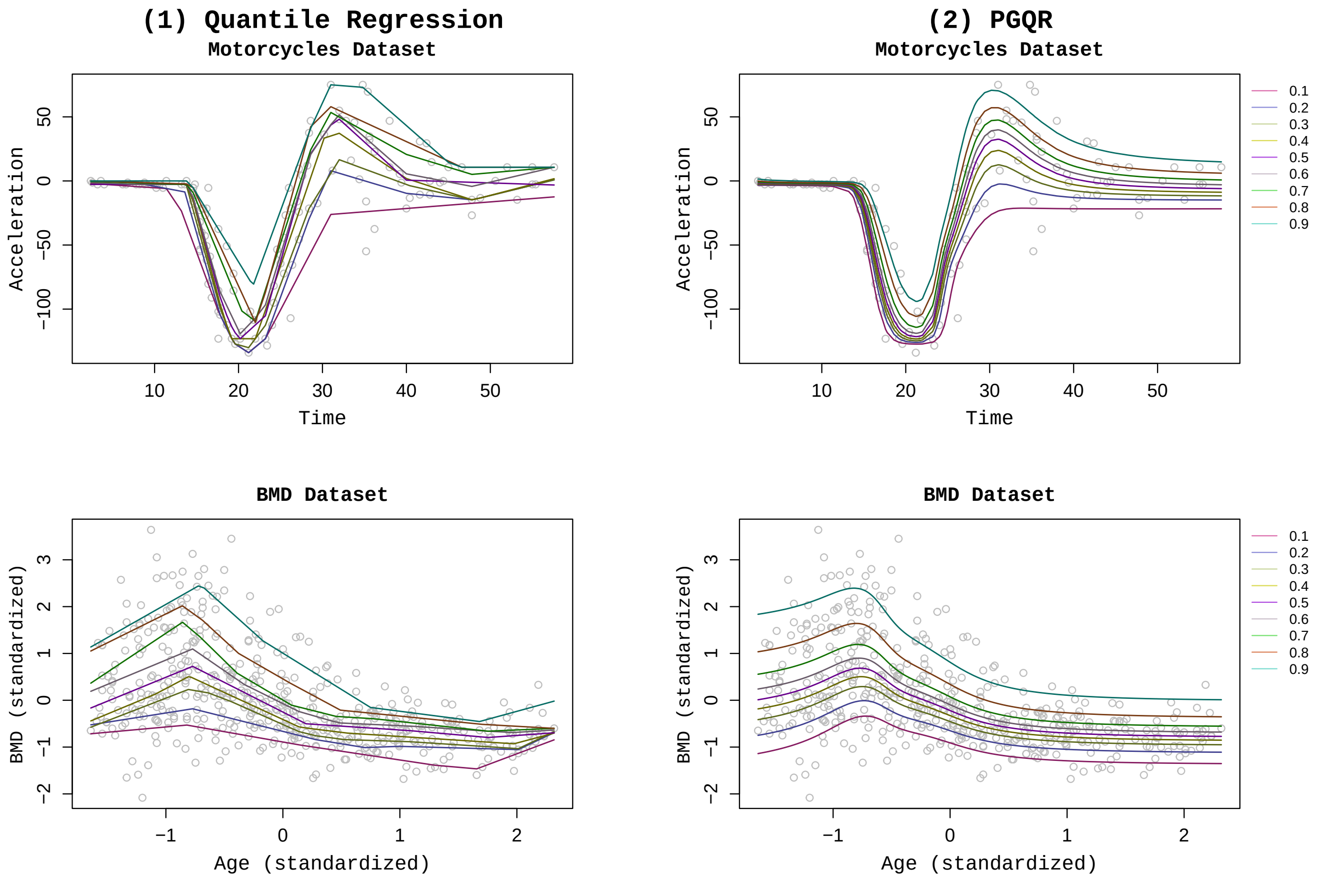

Figure 3: Estimated quantile curves at levels $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ for the motorcycle and BMD datasets. The left two panels plot the estimated curves for unconstrained nonparametric quantile regression, and the right two panels plot the estimated curves for PGQR with the PMNN family.

PGQR model with PMNN as the function class $\mathcal{G}^m$ on two benchmark datasets. The first dataset is the motorcycles data (Silverman, 1985) where the response variable is head acceleration (in g) and the predictor is time from crash impact (in ms). The second application is a bone mineral density (BMD) dataset (Takeuchi et al., 2006) where the response is the standardized relative change in spinal BMD in adolescents and the predictor is the standardized age of the adolescents. For these two datasets, we estimated the quantile functions at different quantile levels $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ over the domain of the predictor. We compared PGQR under PMNN to the unconstrained nonparametric quantile regression approach implemented in the R package `quantreg`. The left two panels of Figure 3 demonstrate that the estimated quantile curves under *unconstrained* nonparametric quantile regression are quite problematic for both the motorcycle and BMD datasets; the quantile curves cross each other at multiple points in the predictor domain. In comparison, the right two panels of Figure 3 show that PGQR with the PMNN family ensures *no* crossing quantiles for either dataset.

4 Scalable Computation for PGQR

4.1 Single-Model Training for PGQR

As illustrated in Section 2.2, PGQR's performance depends greatly on the regularization parameter $\lambda \geq 0$ in the variability penalty (5). If λ is too small or if $\lambda = 0$, then we may underestimate the true conditional variance of $p(Y \mid \boldsymbol{X})$ and/or encounter vanishing variability. On the other hand, if λ is too large, then we may overestimate the conditional variance. Due to the large number of parameters in a DNN, tuning λ would be quite burdensome if we had to repeatedly evaluate the deep generative network $G(\cdot, \tau)$ for multiple choices of λ and training sets. In particular, using cross-validation to tune λ is infeasible for DNNs, especially if the size of the training set is large.

Inspired by the idea of *Generative Multiple-purpose Sampler* (GMS) (Shin et al., 2022), we instead propose to *include* λ as an additional input in the generator, along with data $\boldsymbol{X}$ and quantile τ. In other words, we impose a discrete uniform distribution $p(\lambda)$ for λ whose support Λ is a grid of candidate values for λ. We then estimate the network $G(\cdot, \tau, \lambda)$ with the modified PGQR loss function,

$$\widehat{G} = \underset{G \in \mathcal{G}^m}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\tau,\lambda} \left[\rho_\tau\big(Y_i - G(\boldsymbol{X}_i, \tau, \lambda)\big) + \mathbb{E}_{\tau,\tau'} \left\{ \text{pen}_{\lambda,\alpha} \left(G(\boldsymbol{X}_i, \tau, \lambda), G(\boldsymbol{X}_i, \tau', \lambda) \right) \right\} \right]. \quad (9)$$

Note that (9) differs from the earlier formulation (6) in that $\lambda \in \Lambda$ is *not* fixed in advance. In the PMNN family $\mathcal{G}^m$, λ is included in the unconstrained sub-network $g_{uc} : \mathbb{R}^{p+1} \mapsto \mathbb{R}^h$.

The next corollary to Proposition 2.1 justifies including λ in the generator and optimizing the integrative loss over $\{\tau, \lambda\}$ in the modified PGQR objective (9). The proof of Corollary 4.1 is a straightforward extension of the proof of Proposition 2.1 and is therefore omitted.

Corollary 4.1. *Let* $\widehat{g}_{\tau,\lambda}(\boldsymbol{X}) = \operatorname{argmin}_{g \in \mathcal{H}} \sum_{i=1}^{n} \left[\rho_\tau(Y_i - g(\boldsymbol{X}_i)) + \mathbb{E}_{\tau,\tau'}\{pen_{\lambda,\alpha}\left(g(\boldsymbol{X}_i, \tau), g(\boldsymbol{X}_i, \tau')\right)\}\right]$, *for a class of neural networks* $\mathcal{H}$, *where* $\alpha > 0$ *is fixed and* $\tau, \tau' \overset{iid}{\sim}$ *Uniform*$(0,1)$. *Consider a class of generator functions* $\mathcal{G}$ *where* $\{G \in \mathcal{G} : \mathbb{R}^p \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}\}$. *Suppose that for all* $\boldsymbol{X} \in \mathcal{X} \subset \mathbb{R}^p$, *quantile levels $\tau \in (0,1)$, and tuning parameters $\lambda \in \Lambda$, there exists $G \in \mathcal{G}$ such that the target neural networks $\widehat{g}_{\tau,\lambda}(\boldsymbol{X})$ can be represented by some $G(\boldsymbol{X}, \tau, \lambda)$. Then, for $i = 1, \dots, n$,*

$$\widehat{g}_{\tau,\lambda}(\boldsymbol{X}_i) = \widehat{G}(\boldsymbol{X}_i, \tau, \lambda) \;\; a.s.,$$

with respect to the probability law related to $\mathbb{E}_{\tau,\lambda}$, where $\widehat{G}$ is a solution to (9).

We note that Shin et al. (2022) proposed to use GMS for inference in *linear* quantile regression. In contrast, PGQR is a method for *nonparametric* joint quantile estimation and conditional density estimation (CDE). Linear quantile regression cannot be used for CDE; as a linear model, GMS linear quantile regression also does not suffer from vanishing variability. This is not the case for GQR, which motivates us to introduce the variability penalty in Section 2.2. Finally, we employ the GMS idea for *tuning parameter* selection in PGQR, rather than for constructing pointwise confidence bands (as in Shin et al. (2022)).

By including $\lambda \sim p(\lambda)$ in the generator, PGQR only needs to perform one *single* optimization in order to estimate the network G. We can then *use* the estimated network $\widehat{G}$ to tune λ, as we detail in the next section. This single-model training stands in contrast to traditional smoothing methods for nonparametric quantile regression, which typically require *repeated* model evaluations via (generalized) cross-validation to tune hyperparameters such as bandwidth or roughness penalty.

4.2 Selecting An Optimal Regularization Parameter

We now introduce our method for selecting the regularization parameter λ in our variability penalty (5). Considering the candidate set Λ, the basic idea is to select the best $\lambda^\star \in \Lambda$ that minimizes the distance between the generated conditional distribution and the true conditional distribution. Denote the trained network under the modified PGQR objective (9) as $\widehat{G}(\cdot, \tau, \lambda)$, and let $(\boldsymbol{X}_{\text{val}}, Y_{\text{val}})$ denote a validation sample. After optimizing (9), it is effortless to generate the conditional quantile functions for each $\boldsymbol{X}_{\text{val}}$ in the validation set, each $\lambda \in \Lambda$, and any quantile level $\xi \in (0,1)$. In order to select the optimal $\lambda^\star \in \Lambda$, we first introduce the following proposition.

Proposition 4.2. *Suppose that two univariate random variables Q and W are independent. Then, Q and W have the same distribution if and only if $P(Q < W \mid W)$ follows a standard uniform distribution.*

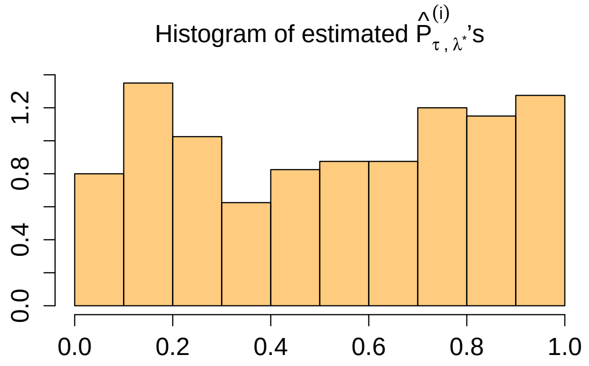

Figure 4: Histogram of the estimated $\widehat{P}^{(i)}_{\tau,\lambda^\star}$'s in the validation set for the optimal $\lambda^\star$.

Based on Proposition 4.2, we know that, given a validation sample $(\boldsymbol{X}_{\text{val}}, Y_{\text{val}})$, the best $\lambda^\star \in \Lambda$ should satisfy $P_{\tau,\lambda}(\widehat{G}(\boldsymbol{X}_{\text{val}}, \tau, \lambda^\star) < Y_{\text{val}} \mid Y_{\text{val}}) \sim \text{Uniform}(0,1)$. In practice, if we have M random quantile levels $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_M) \in (0,1)^M$, we can estimate the probability $P_{\tau,\lambda} := P_{\tau,\lambda}(\widehat{G}(\boldsymbol{X}_{\text{val}}, \tau, \lambda^\star) < Y_{\text{val}} \mid Y_{\text{val}})$ with a Monte Carlo approximation,

$$\widehat{P}^{(i)}_{\tau,\lambda} = M^{-1} \sum_{k=1}^{M} \mathbb{I}\{\widehat{G}(\boldsymbol{X}^{(i)}_{\text{val}},\ \tau_k, \lambda) < Y^{(i)}_{\text{val}}\} \approx P_{\tau,\lambda}(\widehat{G}(\boldsymbol{X}_{\text{val}}, \tau, \lambda^\star) < Y_{\text{val}} \big| \, Y_{\text{val}}), \tag{10}$$

for each ith validation sample $(\boldsymbol{X}^{(i)}_{\text{val}}, Y^{(i)}_{\text{val}})$. With n_{val} validation samples, we proceed to estimate $\widehat{P}^{(i)}_{\tau,\lambda}$ for all n_{val} validation samples $(\boldsymbol{X}^{(i)}_{\text{val}}, Y^{(i)}_{\text{val}})$'s and each $\lambda \in \Lambda$. We then compare the empirical distribution of the $\widehat{P}^{(i)}_{\tau,\lambda}$'s to a standard uniform distribution for each $\lambda \in \Lambda$ and select the $\lambda^\star$ that *minimizes* the distance between $\widehat{P}_{\tau,\lambda}$ and Uniform$(0,1)$. That is, given a valid distance measure d, we select the $\lambda^\star$ which satisfies

$$\lambda^\star = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\, d(\widehat{P}_{\tau,\lambda}, \text{Uniform}(0,1)). \tag{11}$$

There are many possible choices for d in (11). In this paper, we use the Cramer-von Mises (CvM) criterion due to the simplicity of its computation and its good empirical performance. The CvM criterion is straightforward to compute as

$$d(\widehat{P}_{\tau,\lambda}, \text{Uniform}(0,1)) = \sum_{i=1}^{n_{\text{val}}} (i/n_{\text{val}} - \widehat{P}_{\tau,\lambda}^{(i)} - 2/n_{\text{val}})^2 / n_{\text{val}}. \tag{12}$$

After selecting λ^* according to (11), we can easily generate samples from the conditional quantiles of $p(Y \mid \boldsymbol{X})$ for any feature data vector $\boldsymbol{X}$. We simply generate $\widehat{G}(\boldsymbol{X}, \xi_k, \lambda^\star)$, where $k = 1, \ldots, b$, for random quantiles $\xi_1, \ldots, \xi_b \overset{iid}{\sim} \text{Uniform}(0,1)$. For sufficiently large b, the conditional density $p(Y \mid \boldsymbol{X})$ can be inferred from $\{\widehat{G}(\boldsymbol{X}, \xi_k, \lambda^\star)\}_{k=1}^{b}$. The complete algorithm for scalable computation of PGQR is given in Algorithm 1.

We now illustrate our proposed selection method for $\lambda \in \Lambda$ in the simulated linear regression example from Section 2.2. Figure 4 plots the histogram of the estimated $\widehat{P}_{\tau,\lambda^\star}^{(i)}$'s in the validation set for the $\lambda^\star$ which minimizes the CvM criterion. We see that the empirical distribution of the $\widehat{P}_{\tau,\lambda^\star}^{(i)}$'s closely follows a standard uniform distribution.

Figure 5 illustrates how the PGQR solution changes as $\log(\lambda)$ increases for every $\lambda \in \Lambda$. Figure 5 plots the conditional standard deviation (left panel), the CvM criterion (middle panel), and the coverage rate of the 95% confidence intervals (right panel) for the PGQR samples in the validation set. The vertical dashed red line denotes the optimal $\lambda^\star = 0.01$. We see that the $\lambda^\star$ which minimizes the CvM (middle panel) captures the true conditional standard deviation of $\sigma = 1$ (left panel) and attains coverage probability close to the nominal rate (right panel). This example demonstrates that our proposed selection method provides a practical alternative to computationally burdensome cross-validation for tuning λ in the variability penalty (5). In Section B of the Supplementary Material, we further demonstrate that our tuning parameter selection method is suitable to use even when the true conditional variance is very small (true $\sigma^2 = 0.01$). In this scenario, our procedure selects a very small $\lambda^\star \approx 0$, so that PGQR does *not* overestimate the conditional variance.

5 Numerical Experiments and Real Data Analysis

We evaluated the performance of PGQR on several simulated and real datasets. We fixed $\alpha = 1$ or $\alpha = 5$ in the variability penalty (5). We optimized the modified PGQR loss (9) over the

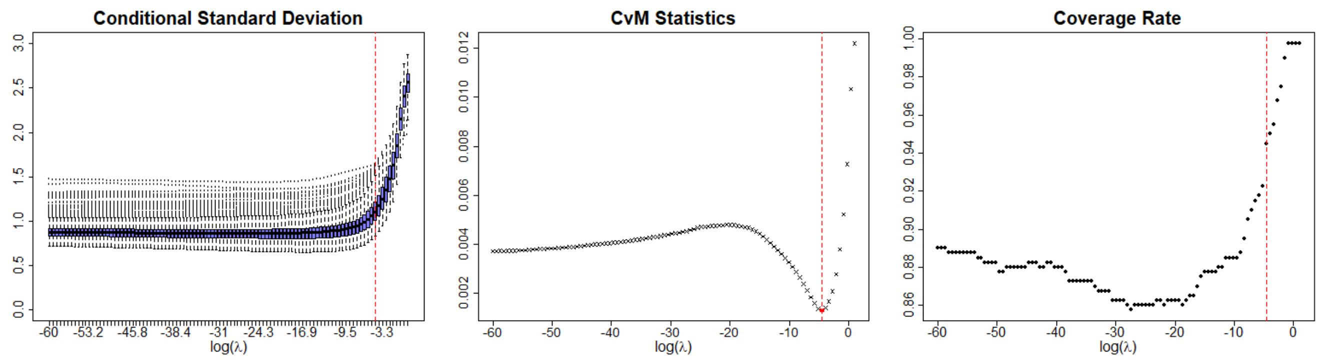

Figure 5: The conditional standard deviation (left panel), CvM statistic (middle panel), and coverage rate (right panel) in the validation set vs. $\log(\lambda)$ for each $\lambda \in \Lambda$. The red dashed line corresponds to the $\lambda^\star$ which minimizes the CvM criterion.

PMNN family (Section 3), where each sub-network had three hidden layers with 1000 neurons per layer and ReLU (Nair and Hinton, 2010) was used as the activation function. The support for Λ was chosen to be 100 equispaced values between 0 and $\exp(1)$. `PyTorch` was used to implement PGQR. To ensure numerical stability, Algorithm 1 was initialized with random weights close to but not exactly zero. We estimated $\widehat{G}$ in (9) using the Adam optimizer (Kingma and Ba, 2014), which has been empirically shown to be robust and helps to mitigate gradient explosion. Gradient clipping (Pascanu et al., 2013) could also be employed to prevent exploding gradients; however, we found that gradient clipping was not needed for PGQR. Throughout our simulations and real data applications, we did not experience any numerical overflow, underflow, or exploding gradients.

We compared PGQR to several other state-of-the-art methods:

- **GCDS** (Zhou et al., 2023). Following Zhou et al. (2023), we trained both a generator and a discriminator using FNNs with one hidden layer of 25 neurons. Despite this simple architecture, we found that GCDS might still encounter vanishing variability. For fair comparison to PGQR, we also increased the number of hidden layers to three and the number of nodes per layer to 1000. We refer to this modification as **deep-GCDS**.

- **WGCS** (Liu et al., 2021). We adopted the gradient penalty recommended by Liu et al. (2021) and set the hyperparameter associated with the gradient penalty as 0.1.

- **FlexCoDE** (Izbicki and Lee, 2017). We considered three ways of estimating the basis functions $\beta_j(\boldsymbol{X})$: nearest-neighbor regression (NNR), sparse additive model (SAM), and XGBoost

Algorithm 1 Scalable Implementation of PGQR

```
1: Split non-test data into non-overlapping training set $(\boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$ and validation set $(\boldsymbol{X}_{\text{val}}, \boldsymbol{Y}_{\text{val}})$
2: Initialize $G$ parameters $\phi$, learning rate $\gamma$, width $h$ of DNN hidden layers, and total epoches $T$
3: procedure: Optimizing $G$
4:     for epoch $t$ in $1, \dots, T$ do
5:         Sample $\tau, \tau' \sim \text{Uniform}(0,1)$ and $\lambda \in \Lambda$
6:         Evaluate loss (9) with $(\boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$
7:         Update $G$ parameters $\phi$ via SGD
8:     end for
9:     return $\widehat{G}$
10: end procedure
11: procedure: Tuning $\lambda$
12:     Set $M = 1000$ and sample $\tau_1, \dots, \tau_M \overset{iid}{\sim} \text{Uniform}(0,1)$
13:     Generate $\{\widehat{G}(\boldsymbol{X}_{\text{val}}^{(i)}, \tau_1, \lambda_l), \dots, \widehat{G}(\boldsymbol{X}_{\text{val}}^{(i)}, \tau_M, \lambda_l)\}$ for each $\lambda_l \in \Lambda$ and each $\boldsymbol{X}_{\text{val}}^{(i)}$, $i = 1, \dots, n_{\text{val}}$
14:     Compute $\widehat{P}_{\tau,\lambda_l}^{(i)}$ as in (10) on $(\boldsymbol{X}_{\text{val}}^{(i)}, Y_{\text{val}}^{(i)})$ for each $i = 1, \dots, n_{\text{val}}$ and each $\lambda_l \in \Lambda$
15:     Select $\lambda^*$ according to (11) with CvM criterion (12) as $d$
16:     return $\lambda^*$
17: end procedure
18: procedure: Estimating $p(Y \mid \boldsymbol{X}_{\text{test}})$ for test data $\boldsymbol{X}_{\text{test}}$
19:     Set $b = 1000$ and sample $\xi_1, \dots, \xi_b \overset{iid}{\sim} \text{Uniform}(0,1)$
20:     Estimate $\xi_k$-th quantile of $Y \mid \boldsymbol{X}_{\text{test}}$ as $\widehat{G}(\boldsymbol{X}_{\text{test}}, \xi_k, \lambda^*)$ for $k = 1, \dots, b$
21:     return $\{\widehat{G}(\boldsymbol{X}_{\text{test}}, \xi_1, \lambda^*), \dots, \widehat{G}(\boldsymbol{X}_{\text{test}}, \xi_b, \lambda^*)\}$
22: end procedure
```

(FlexZboost).

- **Random Forest CDE**, or RFCDE (Pospisil and Lee, 2019).

For FlexCoDE and RFCDE, we adopted the default hyperparameter settings of Izbicki and Lee (2017) and Pospisil and Lee (2019) respectively.

5.1 Simulation Studies

For our simulation studies, we generated data from $Y_i = g(\boldsymbol{X}_i) + \epsilon_i, i = 1, \dots, 2000$, for some function g, where $\boldsymbol{X}_i \overset{iid}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$ and the residual errors ϵ_i's were independent. The specific simulation settings are described below.

1. **Simulation 1: Multimodal and heteroscedastic.** $Y_i = \beta_i X_i + \epsilon_i$, where $\beta_i = \{-1, 0, 1\}$ with equal probability and $\epsilon_i = (0.25 \cdot |X_i|)^{1/2}$.

2. **Simulation 2: Mixture of left-skewed and right-skewed.** $Y_i = \boldsymbol{X}_i^\top \boldsymbol{\beta} + \epsilon_i$, where $\boldsymbol{\beta} \in \mathbb{R}^5$ is equispaced between $[-2, 2]$, and $\epsilon_i = \chi^2(1,1)\mathcal{I}(X_1 >= 0.5) + \log[\chi^2(1,1)]\mathcal{I}(X_1 < 0.5)$. Here, the skewness is controlled by the covariate X_1.

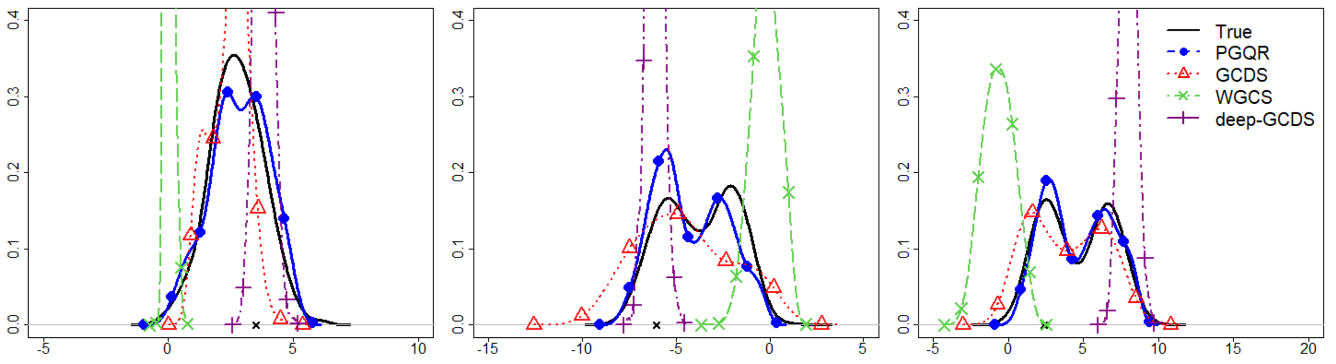

Figure 6: Plots of the estimated conditional densities $p(Y \mid \boldsymbol{X}_{\text{test}})$ for three different test observations from one replication of Simulation 3. Plotted are the estimated conditional densities for PGQR ($\alpha = 1$), GCDS, WGCS, and deep-GCDS.

3. **Simulation 3: Mixture of unimodal and bimodal.** $Y_i = \boldsymbol{X}_i^\top \boldsymbol{\beta}_i + \epsilon_i$, where $\boldsymbol{\beta}_i \in \mathbb{R}^5$ with $\beta_{i1} \in \{-2, 2\}$ with equal probability, $(\beta_{i2}, \beta_{i3}, \beta_{i4}, \beta_{i5})^\top$ are equispaced between $[-2, 2]$, and $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, 1)$. Here, $p(Y \mid \boldsymbol{X})$ is unimodal when $X_1 \approx 0$, and otherwise, it is bimodal.

In our simulations, 80% of the data was used to train the model, 10% was used as the validation set for tuning parameter selection, and the remaining 10% was used as test data to evaluate model performance. In Section B of the Supplementary Material, we present additional simulation results for the following scenarios: $g(\boldsymbol{X}_i)$ is a nonlinear function of $\boldsymbol{X}_i$ (Simulation 4), the error variance is very small (Simulation 5), and the error variance is dependent on $\|\boldsymbol{X}\|_1$ (Simulation 6).

Figure 6 compares the estimated conditional density of $p(Y \mid \boldsymbol{X}_{\text{test}})$ for three test observations from one replication of Simulation 3. We see that PGQR (solid blue line with filled circles) is able to capture both the unimodality *and* the bimodality of the ground truth conditional densities (solid black line). Meanwhile, GCDS (dashed red line with hollow triangles), WGCS (dashed green line with crosses), and deep-GCDS (dashed purple line with pluses) struggled to capture the true conditional densities for at least some test points. In particular, Figure 6 shows some evidence of variance *underestimation* for WGCS and deep-GCDS, whereas this is counteracted by the variability penalty in PGQR. Additional figures from our simulation studies are provided in Section B of the Supplement.

We repeated our simulations for 20 replications. For each experiment, we computed the predicted mean squared error (PMSE) for different summaries of the conditional densities for the test

	Simulation 1			Simulation 2			Simulation 3		
Method	$\mathbb{E}(Y \mid \boldsymbol{X})$	sd$(Y \mid \boldsymbol{X})$	Cov (Width)	$\mathbb{E}(Y \mid \boldsymbol{X})$	sd$(Y \mid \boldsymbol{X})$	Cov (Width)	$\mathbb{E}(Y \mid \boldsymbol{X})$	sd$(Y \mid \boldsymbol{X})$	Cov (Width)
PGQR (α=1)	0.41	0.34	0.95 (23.48)	0.38	0.11	0.93 (8.14)	0.30	0.08	0.92 (6.61)
PGQR (α=5)	**0.36**	**0.31**	0.95 (23.41)	**0.31**	**0.07**	**0.95 (8.83)**	**0.25**	**0.06**	**0.96 (6.60)**
GCDS	10.49	25.82	0.68 (15.48)	0.53	0.27	0.92 (8.55)	0.33	0.12	0.84 (5.78)
WGCS	229.91	73.68	0.15 (4.88)	6.57	1.45	0.80 (9.17)	6.25	1.98	0.71 (8.08)
deep-GCDS	7.99	56.87	0.38 (7.42)	5.41	2.89	0.42 (2.12)	6.11	2.78	0.28 (2.04)
FlexCoDE-NNR	0.97	0.54	0.96 (23.94)	0.83	0.36	0.91 (9.03)	1.12	0.75	0.92 (9.11)
FlexCoDE-SAM	0.37	0.62	0.97 (25.07)	0.73	1.03	0.93 (11.15)	1.01	1.99	0.93 (10.91)
FlexZBoost	0.77	63.81	**1.00 (46.11)**	1.29	0.36	0.91 (8.28)	1.77	0.74	0.85 (7.88)
RFCDE	1.98	0.72	0.44 (25.46)	0.61	0.34	0.56 (6.26)	0.83	0.65	0.96 (23.06)

Table 1: Table reporting the PMSE for the conditional expectation and standard deviation, as well as the coverage rate (Cov) and average width of the 95% prediction intervals, for Simulations 1 through 3. Results were averaged across 20 replicates.

data. We define the PMSE as

$$\text{PMSE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \left(\widehat{m}(\boldsymbol{X}_{\text{test},i}) - m(\boldsymbol{X}_{\text{test},i}) \right)^2 , \tag{13}$$

where $m(\boldsymbol{X})$ generically refers to the conditional mean of Y given $\boldsymbol{X}$, i.e. $\mathbb{E}(Y \mid \boldsymbol{X})$, or the conditional standard deviation, i.e. sd$(Y \mid \boldsymbol{X})$. For the generative models (PGQR, GCDS, WGCS, and deep-GCDS), we approximated $\widehat{m}$ in (13) by Monte Carlo simulation using 1000 generated samples, while for FlexCoDE and RFCDE, we approximated $\widehat{m}$ using numerical integration. In addition to PMSE, we also used the 2.5th and 97.5th percentiles of the predicted conditional densities to construct 95% prediction intervals for the test data. We then calculated the coverage probability (Cov) and the average width for these prediction intervals.

Table 1 summarizes the results for PGQR with $\alpha \in \{1, 5\}$ in (5) and all competing methods, averaged across 20 experiments. There was not much substantial difference between $\alpha = 1$ and $\alpha = 5$ for PGQR. Table 1 shows that PGQR had the lowest PMSE in all three simulations and attained coverage close to the nominal rate. In Simulations 2 and 3, the prediction intervals produced by PGQR had the highest coverage rate. In Simulation 1, FlexZBoost had 100% coverage, but the average width of the prediction intervals for FlexZBoost was considerably larger than that of the other methods, suggesting that the intervals produced by FlexZBoost may be too conservative to be informative.

Since PGQR approximates conditional quantile functions, we also compared PGQR with two other neural network approaches for nonparametric joint quantile regression. Specifically, we com-

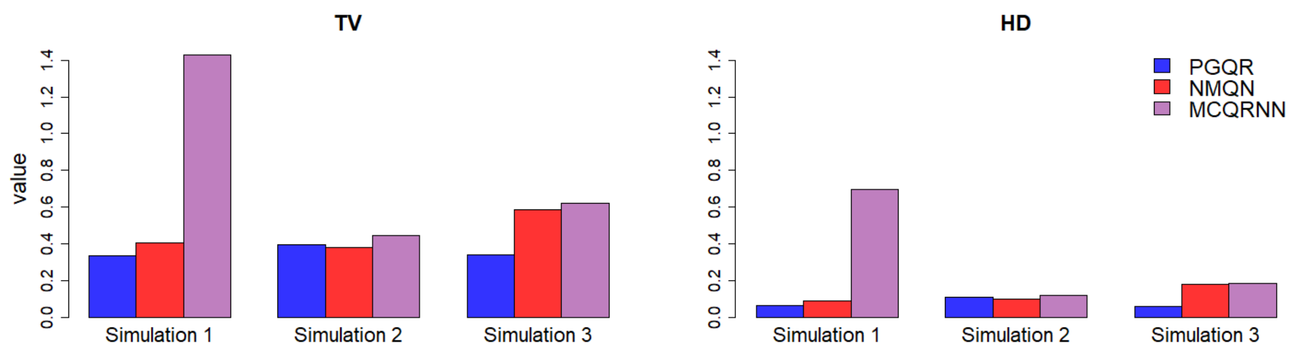

Figure 7: Barplots of the average total variation distance (TV) and Hellinger distance (HD) across 20 replicates evaluated at $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ for Simulations 1 through 3.

pared PGQR to the MCQRNN method of Cannon (2018) and the NMQN method of Moon et al. (2021) with ℓ_1-penalization. These methods are available in the R packages `qrnn` and `l1pm` respectively.

To compare PGQR to NMQN and MCQRNN, we considered the quantile accuracy criterion used by Moon et al. (2021). Given K prespecified quantile levels, let $Q_{\tau_k}(\cdot)$ be the conditional quantile function for $k = 1, \ldots, K$, and let $\widehat{Q}_{\tau_k}$ denote the associated estimated quantile function. Given $\boldsymbol{X}$, we obtained the quantile accuracy, $\widehat{r}_k(\boldsymbol{X}; \widehat{Q}_{\tau_k}) = B^{-1} \sum_{b=1}^{B} \mathbb{I}(\boldsymbol{Y}_b^* \leq \widehat{Q}_{\tau_k}(\boldsymbol{X}))$, where $\boldsymbol{Y}_{1:B}^*$ were sampled from $P(Y \mid \boldsymbol{X})$ with $B = 2000$ and $K = 9$ ($\tau_k \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$). The relative frequency $\widehat{p}_k(\boldsymbol{X}, \widehat{Q}_{\tau_k})$ between τ_{k-1} and τ_k was then calculated as $\widehat{r}_k(\boldsymbol{X}; \widehat{Q}_{\tau_k}) - \widehat{r}_k(\boldsymbol{X}; \widehat{Q}_{\tau_{k-1}})$. If $\widehat{Q}_{\tau_k}$ is identical to true conditional quantile function, then $\widehat{p}_k(\boldsymbol{X}, \widehat{Q}_{\tau_k}) \xrightarrow{p} 1/(K+1)$ as $B \to \infty$. Based on this fact, we used total variation distance (TV) and Hellinger distance (HD) to measure the distance between $\{\widehat{p}_k(\boldsymbol{X}, \widehat{Q}_{\tau_k})\}_{k=1,\ldots,K}$ and $\{1/(K+1), \ldots, 1/(K+1)\}$. A more detailed description can be found in Moon et al. (2021).

In Figure 7, we plot the performance of PGQR, NMQN, and MCQRNN for Simulations 1 through 3. Additional simulation results are provided in Section B of the Supplement. Overall, PGQR had comparable performance to NMQN, and both PGQR and NMQN performed better than MCQRNN with lower average TV and HD. We reiterate that a major difference between these methods is that PGQR uses a deep *generative* model to *generate* the conditional quantiles from many random quantile levels τ's, whereas NMQN and MCQRNN require the practitioner to specify K target quantile levels $\tau_k, k = 1, \ldots, K$, at which to estimate the conditional quantiles.

			PGQR		GCDS		deep-GCDS		WGCS	
Dataset	n	p	Cov	Width	Cov	Width	Cov	Width	Cov	Width
machine	557	4	**0.96**	2.82	0.91	1.92	0.87	1.52	0.69	1.66
fish	908	6	**0.96**	3.29	0.79	2.38	0.54	1.08	0.80	2.36
noise	1503	5	**0.92**	7.58	0.00	1.68	0.00	0.24	0.00	22.41
YVRprecip	10958	5	**0.94**	12.6	0.73	3.62	0.89	16.1	0.06	2.49

Table 2: Results from our real data analysis. Cov and width denote the coverage rate and average width respectively of the 95% prediction intervals for the test observations.

5.2 Real Data Analysis

We examined the performance of PGQR on three real datasets from the UCI Machine Learning Repository, which we denote as: `machine`, `fish`, and `noise`.[1] We also applied PGQR to `YVRprecip` dataset analyzed by Cannon (2018) and Moon et al. (2021). The `machine` dataset comes from a real experiment that collected the excitation current (Y) and four machine attributes ($\boldsymbol{X}$) for a set of synchronous motors (Kahraman, 2014). The `fish` dataset contains the concentration of aquatic toxicity (Y) that can cause death in fathead minnows and six molecular descriptors ($\boldsymbol{X}$) described in Cassotti et al. (2015). The `noise` dataset measures the scaled sound pressure in decibels (Y) at different frequencies, angles of attacks, wind speed, chord length, and suction side displacement thickness ($\boldsymbol{X}$) for a set of airfoils (Lopez et al., 2008). Finally, the `YVRprecip` dataset collects daily precipitation totals (mm) at Vancouver International Airport from 1971 to 2000. The five covariates in `YVRprecip` are seasonal cycle, daily sea-level pressures, 700-hPa specific humidities, and 500-hPa geopotential heights (Cannon, 2018; Moon et al., 2021). The `noise` and `YVRprecip` datasets display extreme skewness in the responses, rendering them especially challenging for conditional quantile and conditional density estimation.

We examined the out-of-sample performance for PGQR (with fixed $\alpha = 1$), GCDS, deep-GCDS, and WGCS. In particular, 80% of each dataset was randomly selected as training data, 10% was used as validation data for tuning parameter selection, and the remaining 10% was used as test data for model evaluation. To compare these deep generative methods, we considered the out-of-sample coverage rate (Cov) and the average width of the 95% prediction intervals in the test data.

The results from our real data analysis are summarized in Table 2. On all four datasets, PGQR achieved higher coverage that was closer to the nominal rate than the competing methods. It seems

[1]Accessed from `https://archive.ics.uci.edu/ml/index.php`.

as though the other generative approaches may have been impacted by the vanishing variability phenomenon, resulting in too narrow prediction intervals that did not cover as many test samples. In particular, GCDS, deep-GCDS, and WGCS all performed very poorly on the `noise` dataset, with an out-of-sample coverage rate of zero. On the other hand, with the help of the variability penalty (5), PGQR did *not* underestimate the variance and demonstrated an overwhelming advantage over these other methods in terms of predictive power. Moreover, the average widths of the PGQR prediction intervals were not overwhelmingly large so as to be uninformative.

Additional illustrations and data analyses are provided in Section A of the Supplement. In particular, the Supplement presents a clinically important application of PGQR for predicting muscular strength in older adults.

6 Discussion

In this paper, we have made contributions to both the quantile regression and deep learning literature. Specifically, we proposed PGQR as a new deep generative approach to joint quantile estimation and conditional density estimation. Different from existing conditional sampling methods (Zhou et al., 2023; Liu et al., 2021), PGQR employs a novel variability penalty to counteract the *vanishing variability* phenomenon in deep generative networks. We introduced the PMNN family of neural networks to enforce the monotonicity of quantile functions. Finally, we provided a scalable implementation of PGQR which requires solving only a single optimization to select the regularization term in the variability penalty. Through analyses of real and simulated datasets, we demonstrated PGQR's ability to capture critical aspects of the conditional distribution such as multimodality, heteroscedasticity, and skewness.

We anticipate that our penalty on vanishing variability in generative networks is broadly applicable for a wide number of loss functions besides the check function. In the future, we will extend the variability penalty to other deep generative models and other statistical problems besides quantile regression. In addition, we plan to pursue variable selection, so that our method can also identify the most relevant covariates when the number of features p is large. Owing to its single-model training, PGQR is scalable for large n. However, further improvements are needed in order for PGQR to avoid the curse of dimensionality for large p.

ACKNOWLEDGMENTS

We are grateful to the two anonymous reviewers whose thoughtful feedback helped us greatly improve our paper. We also thank Dr. Jun Liu from Harvard University for his helpful comments.

DISCLOSURE STATEMENT

The authors report there are no competing interests to declare.

SUPPLEMENTARY MATERIAL

Supplementary Material: The Supplementary Material contains additional illustrations and real data analyses, more simulation results, proofs of the propositions, and analyses of model complexity and the choice of hyperparameter α. (**.pdf**)

R-package: `R` code for PGQR and datasets can be found at the following URL:

`https://github.com/shijiew97/PGQR`.

References

Allen-Zhu, Z., Li, Y., and Liang, Y. (2019). Learning and generalization in overparameterized neural networks, going beyond two layers. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 6158–6169. Curran Associates, Inc.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232. PMLR.

Arpit, D., Jastrzundefinedbski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. (2017). A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org.

Cannon, A. J. (2018). Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes. *Stochastic Environmental Research and Risk Assessment*, 32(11):3207–3225.

Cassotti, M., Ballabio, D., Todeschini, R., and Consonni, V. (2015). A similarity-based QSAR model for predicting acute toxicity towards the fathead minnow (pimephales promelas). *SAR and QSAR in Environmental Research*, 26(3):217–243.

Chaudhuri, P. and Loh, W.-Y. (2002). Nonparametric estimation of conditional quantiles using quantile regression trees. *Bernoulli*, 8(5):561–576.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1096–1105. PMLR.

Daniels, H. and Velikova, M. (2010). Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917.

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., and Dehmer, M. (2020). An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, 3.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Green, P. J. and Silverman, B. W. (1994). *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman and Hall.

Izbicki, R. and Lee, A. B. (2017). Converting high-dimensional regression to high-dimensional conditional density estimation. *Electronic Journal of Statistics*, 11(2):2800–2831.

Jiang, X., Jiang, J., and Song, X. (2012). Oracle model selection for nonlinear models based on weighted composite quantile regression. *Statistica Sinica*, 22(4):1479–1506.

Kahraman, H. T. (2014). Metaheuristic linear modeling technique for estimating the excitation current of a synchronous motor. *Turkish Journal of Electrical Engineering and Computer Sciences*, 22(6):1637–1652.

Kingma, D. P. and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koenker, R. and Bassett Jr, G. (1982). Robust tests for heteroscedasticity based on regression quantiles. *Econometrica*, 50(1):43–61.

Koenker, R., Chernozhukov, V., He, X., and Peng, L. (2017). *Handbook of Quantile Regression*. CRC press.

Koenker, R. and Hallock, K. F. (2001). Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156.

Koenker, R., Ng, P., and Portnoy, S. (1994). Quantile smoothing splines. *Biometrika*, 81(4):673–680.

Li, D., Li, Q., and Li, Z. (2021). Nonparametric quantile regression estimation with mixed discrete and continuous data. *Journal of Business & Economic Statistics*, 39(3):741–756.

Liu, S., Zhou, X., Jiao, Y., and Huang, J. (2021). Wasserstein generative learning of conditional distribution. *arXiv preprint arXiv:2112.10039*.

Liu, X., Han, X., Zhang, N., and Liu, Q. (2020). Certified monotonic neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15427–15438. Curran Associates, Inc.

Lopez, R., Balsa-Canto, E., and Onate, E. (2008). Neural networks for variational problems in engineering. *International Journal for Numerical Methods in Engineering*, 75(11):1341–1360.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6232—6240. Curran Associates, Inc.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7(35):983–999.

Mikulincer, D. and Reichman, D. (2022). Size and depth of monotone neural networks: interpolation and approximation. *arXiv preprint arXiv:2207.05275*.

Montanari, A. and Zhong, Y. (2022). The interpolation phase transition in neural networks: Memorization and generalization under lazy training. *The Annals of Statistics*, 50(5):2816 – 2847.

Moon, S. J., Jeon, J.-J., Lee, J. S. H., and Kim, Y. (2021). Learning multiple quantiles with neural networks. *Journal of Computational and Graphical Statistics*, 30(4):1238–1248.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA.

Pospisil, T. and Lee, A. B. (2019). RFCDE: Random forests for conditional density estimation and functional data. *arXiv preprint arXiv:1804.05753*.

Sangnier, M., Fercoq, O., and d'Alché-Buc, F. (2016). Joint quantile regression in vector-valued RKHSs. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29, page 3700–3708. Curran Associates, Inc.

Shen, G., Jiao, Y., Lin, Y., Horowitz, J. L., and Huang, J. (2021). Deep quantile regression: Mitigating the curse of dimensionality through composition. *arXiv preprint arXiv:2107.04907*.

Shen, G., Jiao, Y., Lin, Y., Horowitz, J. L., and Huang, J. (2022). Estimation of non-crossing quantile regression process with deep requ neural networks. *arXiv preprint arXiv:2207.10442*.

Shin, M., Wang, S., and Liu, J. S. (2022). Generative multiple-purpose sampler for weighted M-estimation. *arXiv preprint arXiv:2006.00767*.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(1):1–21.

Soltanolkotabi, M., Javanmard, A., and Lee, J. D. (2019). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769.

Tagasovska, N. and Lopez-Paz, D. (2018). Frequentist uncertainty estimates for deep learning. *arXiv preprint arXiv:1811.00908*.

Takeuchi, I., Le, Q., Sears, T., Smola, A., et al. (2006). Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7(45):1231–1264.

van den Burg, G. and Williams, C. (2021). On memorization in probabilistic deep generative models. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27916–27928. Curran Associates, Inc.

Xu, Q., Deng, K., Jiang, C., Sun, F., and Huang, X. (2017). Composite quantile regression neural network with applications. *Expert Systems with Applications*, 76:129–139.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhang, H. and Zhang, Z. (1999). Feedforward networks with monotone constraints. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 3, pages 1820–1823 vol.3.

Zhong, Q. and Wang, J.-L. (2023). Neural networks for partially linear quantile regression. *Journal of Business & Economic Statistics*, pages 1–12.

Zhou, X., Jiao, Y., Liu, J., and Huang, J. (2023). A deep generative approach to conditional sampling. *Journal of the American Statistical Association*, 118(543):1837–1848.

Zou, H. and Yuan, M. (2008). Composite quantile regression and the oracle model selection theory. *The Annals of Statistics*, 36(3):1108–1126.