Communications for Statistical Applications and Methods
2024, Vol. 31, No. 6, 627–644

https://doi.org/10.29220/CSAM.2024.31.6.627
Print ISSN 2287-7843 / Online ISSN 2383-4757

Local quantile ensemble for machine learning methods

Suin Kim[a], Yoonsuh Jung[1,a]

[a]Department of Statistics, Korea University, Korea

Abstract

Quantile regression models have become popular due to their benefits in obtaining robust estimates. Some machine learning (ML) models can estimate conditional quantiles. However, current ML methods mainly focus on just adapting quantile regression. In this paper, we propose a local quantile ensemble based on ML methods, which averages multiple estimated quantiles near the target quantile. It is designed to enhance the stability and accuracy of the quantile fits. This approach extends the composite quantile regression algorithm that typically considers the central tendency under a linear model. The proposed methods can be applied to various types of data having nonlinear and heterogeneous trend. We provide an empirical rule for choosing quantiles around the target quantile. The bias-variance tradeoff inherent in this method offers performance benefits. Through empirical studies using Monte Carlo simulations and real data sets, we demonstrate that the proposed method can significantly improve quantile estimation accuracy and stabilize the quantile fits.

Keywords: ensemble learning, quantile averaging, quantile crossing, tree-based models, variance reduction

1. Introduction

Traditional regression models focus on estimating the conditional mean. However, relying solely on the mean is inadequate since real data often exhibits heteroscedasticity. Instead, there is a growing recognition of the need to comprehend the underlying distribution. Ideally, estimating the entire conditional distribution would be beneficial. Foresi and Peracchi (1995) and Klein *et al.* (2015) suggested distributional regression methods to capture the full conditional distribution. However, they can be computationally demanding and pose challenges in terms of interpretability.

Besides, quantile regression (QR) (Koenker and Bassett, 1978) provides an efficient method to achieve a comprehensive understanding of the data. It is particularly robust in handling data with high skewness, non-Gaussian errors, and outliers. We can explore the conditional distribution through QR without making assumptions regarding the global distribution (Chapter 2.5 in Koenker, 2005). Many fields that require knowledge beyond the conditional mean utilize QR these days. For example, meteorology adopts QR to forecast heavy rainfall, healthcare examines treatment side effects (Powell, 2020), and finance calculates the value-at-risk of a risky investment (Keilbar and Wang, 2022) using QR.

Estimating quantiles for complex data requires nonlinear models. Tree-based models targeting quantiles include quantile regression forest (QRF) (Meinshausen and Ridgeway, 2006) and quantile extremely randomized trees (QERT) (Geurts *et al.*, 2006). Boosting models such as gradient boosting machines (GBM) (Friedman, 2001), LightGBM (Ke *et al.*, 2017), and extreme gradient boosting

[1]Corresponding author: Department of Statistics, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul 02841, Korea. E-mail: yoons77@korea.ac.kr

Published 30 November 2024 / journal homepage: http://csam.or.kr

(XGBoost) allow for quantile estimation by substituting their original loss functions with a check loss. Quantile smoothing splines (QSS) (Koenker and Portnoy, 1994) and support vector quantile regression (SVQR) (Takeuchi *et al.*, 2006; Li *et al.*, 2007) are another variants of nonlinear quantile regression models.

The crossing of the fitted quantiles is often an issue in the quantile regression models. To prevent the fitted quantiles from crossings, researchers often impose constraints (He, 1997; Liu and Wu, 2011; Shin and Jung, 2023). Although the proposed model does not employ certain constraint, we observe that crossings can be much relieved as a byproduct.

In this paper, we aim to apply the concept of composite quantile regression (CQR) (Zou and Yuan, 2008) to various ML models. CQR combines objective functions for multiple quantiles and minimizes them simultaneously. Bloznelis *et al.* (2019) and Zhao and Xiao (2014) showed that the combined estimator has a higher efficiency than the original quantile regression. However, CQR primarily treat the central tendency of linear models, which does not well matched for the terminology 'quantile'. Xie and Wen (2019) and Hatalis *et al.* (2019) integrate CQR concepts with deep learning models, but those algorithms still primarily target the mean or median. One of the assumptions of CQR is that all quantiles share the same regression parameter, and only varying the intercept term. This assumption is restricted when analyzing heterogeneous data. Therefore, our goal is to relieve this assumption and to develop methods that can estimate all range of quantiles for nonlinear and heterogeneous data.

We propose a new methodology called local quantile ensemble (LQE). LQE aims to ensemble models for several quantiles in a straightforward way, averaging multiple quantile fits near a target. This method is not restricted to specific algorithms, making it applicable to any quantile-targeted models. While traditional quantile estimation focus on single quantile, LQE averages multiple quantiles near the target. As a result, its performance may improve through the bias-variance trade-off. Additionally, LQE can reduce quantile crossing, which is one of the main concerns in QR.

Our paper will proceed as follows: First, we review some quantile machine learning algorithms in Section 2.1. In Section 2.2, we introduce the details of LQE, and illustrate the bias-variance trade-off with a pilot example. Sections 3 and 4 show extensive numerical and visual results of the simulated and real data sets, respectively. Monte Carlo simulations and real data analysis reveal the usefulness of the proposed methods.

2. Methodology

2.1. Review of quantile machine learning methods

Consider a univariate response variable $y \in \mathbb{R}$, and let $\mathbf{x}$ represent covariates with p dimensions. Suppose we have independent samples $(\mathbf{x}_i, y_i)$, $i = 1, \dots, n$. The goal is to estimate the conditional quantile of y given $\mathbf{x}$. Denote the true quantile as $Q_\tau(y|\mathbf{x})$, where $\tau \in (0,1)$ indicates the quantile of interest. Then, true conditional quantile is defined as

$$Q_\tau(y \mid \mathbf{x}) = \inf\{y \in \mathbb{R} : F(y \mid \mathbf{x}) \geq \tau\}. \tag{2.1}$$

Here, F denotes a conditional distribution of y given $\mathbf{x}$. To obtain the estimates of (2.1), $\hat{Q}_\tau(y|\mathbf{x})$, there are several classes of machine learning models.

Boosting models (e.g., GBM, LightGBM, and XGBoost) have been used mainly for mean regression, but they can be easily implemented for quantile estimation. Conventional boosting models employ mean squared error (MSE) or mean absolute error as a loss function. We substitute these with the mean check loss function to estimate conditional quantile. The check loss is defined as

$L_\tau(y, \hat{y}) = |y - \hat{y}|\{\tau - I(y < \hat{y})\}$, where I is an indicator function that returns 1 if the condition inside is true or 0 otherwise.

With the mentioned loss function, boosting models learn by gradually integrating weak learners. The weak learner is a model that performs relatively poor. Its performance is just above the random guess. Denote the number of iterations as M. Then, we can formulate boosting models for quantiles as follows:

$$\hat{Q}_\tau(y \mid \mathbf{x}) = \sum_{m=1}^{M} \lambda \hat{g}_m(\mathbf{x}), \quad \text{where} \quad \hat{g}_m(\mathbf{x}) = \underset{g}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L_\tau\left(y_{i,(m)}, g(\mathbf{x}_i)\right). \tag{2.2}$$

The λ represents a learning rate, which is typically a very small constant. $\hat{g}^m(\mathbf{x})$ and $y_{i,(m)}$ are the decision tree and the i^{th} residual in the m^{th} iteration, respectively, for $m = 1, \dots, M$. Each tree is fitted to correct the previous errors. The algorithm stops either when the maximum number of iterations is reached or when a new decision tree no longer improves the loss function.

Meanwhile, QRF and QERT estimate quantiles by obtaining an empirical conditional distribution. These models calculate the weighted average of $I(y_i \leq y)$. Suppose we construct a QRF with H trees, where H is a positive integer. Following the notation of Meinshausen and Ridgeway (2006), let θ_h be a random parameter vector of the h^{th} tree for $h = 1, \dots, H$. Let $R_{l(\mathbf{x},\theta_h)}$ be the rectangular subspace generated by a single leaf $l(\mathbf{x}, \theta_h)$. Then QRF combines all trees by averaging the weight vectors from $h = 1, \dots, H$ trees as follows:

$$\hat{F}(y \mid \mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) I(y_i \leq y), \;\text{ where }\; w_i(\mathbf{x}) = \frac{1}{H} \sum_{h=1}^{H} w_i(\mathbf{x}, \theta_h) \;\text{ and }\; w_i(\mathbf{x}, \theta_h) = \frac{I(\mathbf{x}_i \in R_{l(\mathbf{x},\theta_h)})}{\sum_{j=1}^{n} I(\mathbf{x}_j \in R_{l(\mathbf{x},\theta_h)})}.$$

Note that the weight vector $w_i(\mathbf{x}, \theta_h)$ indicates a proportion of observations that are contained in the leaf for $\mathbf{x}$. We can subsequently obtain the estimates $\hat{Q}_\tau(y|\mathbf{x})$ by substituting the estimates $\hat{F}(y|\mathbf{x})$ in place of $F(y|\mathbf{x})$ in (2.1) as follows:

$$\hat{Q}_\tau(y \mid \mathbf{x}) = \inf\left\{y \in \mathbb{R} : \hat{F}(y \mid \mathbf{x}) \geq \tau\right\}. \tag{2.3}$$

Once the empirical distribution of $\hat{F}(y|\mathbf{x})$ is obtained, one can vary τ in (2.3) to target any quantiles of interest. Thus, we can obtain fits for multiple quantiles from a single fitted model. Furthermore, quantile crossing does not occur in QRF since the obtained empirical distribution is a monotonely increasing function of y.

QERT uses a similar fitting algorithm as QRF but amplifies the randomness. The primary differences are that each tree in QERT completely randomizes the cutpoints of the nodes and uses the entire sample to grow the tree. As a result, we can obtain the estimates using the same method as in (2.3), ensuring they do not exceed the above quantile fit. Further details of the algorithm can be found in Geurts *et al.* (2006).

Many studies compare various ML models to analyze real-world data with nonlinear patterns. To improve the performance ML quantile models, we suggest to combine multiple models with the ideas of CQR. The details of ensemble comes in the next section.

2.2. Local quantile ensemble

The methodology we propose here is local quantile ensemble (LQE). The core idea of LQE is to combine K estimates of quantiles $0 < \tau_1 < \cdots < \tau_K < 1$ near the target quantile τ to yield a more accurate estimate. For simplicity, we refer to the K quantiles as local quantiles where the target quantile is located at the center. LQE combines estimates of the local quantiles around the target quantile as

$$\hat{Q}_{\tau}^{LQE}(y \mid \mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} \hat{Q}_{\tau_k}(y \mid \mathbf{x}). \tag{2.4}$$

We can easily implement LQE for models that return conditional quantile estimates, such as those in (2.2) and (2.3). While (2.4) can be extended to a weighted average, this paper only addresses the equally weighted version.

LQE has two advantages over previous studies. First, it is broadly applicable beyond linear models. Existing methods are largely based on the linear model, where the objective is to estimate the regression parameter. For example, Blozenlis *et al.* (2019) and Zhao and Xiao (2014) compute the average of the coefficient estimators for multiple quantiles. To extend this to ML models, we averages the fitted values instead of averaging the parameters. Second, LQE is capable of targeting any quantiles between 0 and 1. Most studies of CQR focus on estimating the central trend of data using full quantile information, typically limiting the target to the mean or median. In contrast, LQE can aims to estimate all conditional quantiles. As a result, LQE provides a more comprehensive approach to quantile prediction.

The ensemble effect can improve the performance of LQE. Ensemble approaches address problems where models may overfit or become trapped at a local optimum (Dietterich, 2000). Traditional quantile regression utilizes restricted information near the target quantile, making it sensitive to changes in observations above or below the fit (Koenker, 2005, Chapter 2.5). Consequently, estimates tends to be less accurate in regions with few samples or when τ is close to an extreme quantile (Li and Wang, 2019). By combining adjacent quantiles, LQE utilizes a broader range of samples, effectively addressing these issues. The bias-variance trade-off offered by LQE is crucial here. By averaging multiple quantile fits, LQE reduces the variance associated with individual model predictions, leading to more reliable and robust estimates. Despite an increase in bias, the overall performance of LQE often improves due to the substantial reduction in variance.

Next, we detail how to decide the local quantiles $\{\tau_1, \ldots, \tau_K\}$. Two factors need to be specified: The range and the number of local quantiles. Combining quantiles that are far from the target quantile might not be helpful. Therefore, it may be necessary to restrict the range of the local quantiles. Denote the range as band(τ), which stands for bandwidth. The optimal length of band(τ) is influenced by the model and algorithm chosen, but establishing a theoretical optimum for each model is challenging. Therefore, we propose a simple heuristic: Use a wider bandwidth for quantiles with high densities. With the sparse density, each quantile fit is unstable. Even a small change in the target quantile can have a significant impact on the fit. Therefore, it is recommended to set band(τ) narrowly for those spaces. Because typical data sets have more samples near the center, we suggest the following rule for the bandwidth.

(Rule 1) The bandwidth for the target quantile τ, band(τ), is determined as

$$\text{band}(\tau) = [\tau \pm 0.5c_\tau], \qquad \text{where} \qquad c_\tau = -|0.9(\tau - 0.5)| + 0.5.$$

Note that the maximum length of the band(τ) in Rule 1 is 0.5, which is achieved when $\tau = 0.5$.

Once we fix the bandwidth according to the target quantile τ, the next decision is the number of quantiles to combine, denoted by K. This is simpler than determining the range, as Koenker (2005,

Table 1: The mean of estimated squared bias and variance of conditional quantile fits with LightGBM for $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$, and 0.9. All values are multiplied by 10^2

	$\tau = 0.1$		$\tau = 0.3$		$\tau = 0.5$		$\tau = 0.6$		$\tau = 0.8$		$\tau = 0.9$	
	Bias^2	Var	Bias^2	Var	Bias^2	Var	Bias^2	Var	Bias^2	Var	Bias^2	Var
SQ	0.056	0.397	0.035	0.887	0.084	1.260	0.147	1.166	0.225	2.831	0.387	6.404
LQE	0.082	0.378	0.042	0.706	0.175	0.863	0.255	1.042	0.478	2.316	0.983	3.115

Chapter 5.5) notes that selecting a large K value does not significantly affect performance. Previous studies have often set K to 9 or 10 (Zhao and Xiao, 2014; Cannon, 2018). Furthermore, Zou and Yuan (2008) found no significant difference in the performance of the estimates beyond $K = 9$. When too many number of quantiles are averaged, the correlation between local quantile estimates increases. This hinders the variance reduction. Therefore, we fix K at 9 for LQE.

Pilot example: To demonstrate the bias-variance trade-off of the proposed method, we design a pilot experiment. Following Scenario 1 in Section 3, we generate the training data sets for $R = 100$ times. We fit a LightGBM model for target quantile $\tau = 0.1$, 0.3, 0.5, 0.6, 0.8, and 0.9. We decompose MSE into the squared bias and variance with a fixed test data set of 1000 samples. Let $\hat{Q}_\tau^{(r)}(y|\mathbf{x}_i)$ be the τ^{th} conditional quantile estimate given $\mathbf{x}_i$ in the r^{th} training data set. We calculate the squared bias and variance for the i^{th} test sample as follows:

$$\hat{\text{Bias}}_i^2 = \left(\bar{Q}_\tau(y \mid \mathbf{x}_i) - Q_\tau(y \mid \mathbf{x}_i)\right)^2, \quad \hat{\text{Var}}_i = \frac{1}{R}\sum_{r=1}^{R}\left(\hat{Q}_\tau^{(r)}(y \mid \mathbf{x}_i) - \bar{Q}_\tau(y \mid \mathbf{x}_i)\right)^2, \quad \text{for } i = 1, \ldots, n.$$

Here, $Q_\tau(y|\mathbf{x}_i)$ represents a true quantile, and $\bar{Q}_\tau(y|\mathbf{x}_i)$ is the average of the estimated quantile function over the R replication.

Table 1 summarizes the average of the estimated squared bias and variance across 1000 test samples. SQ refers to a single quantile from a model that fits the target quantile only. We observe that LQE reduces the variance while only slightly increasing the bias in general. The larger bias of LQE compared to SQ is a result of employing multiple quanitle fits. Nevertheless, our empirical findings indicate that the reduction in variance outweighs the increase in bias. This reduction in variance is particularly noticeable in the quantiles $\tau = 0.8$ and 0.9, where the corresponding density of the error distribution is low. These results show the favorable effect of ensemble. In the next section, we will see that the variance reduction can lead to better performance in various ML models. Although we no more presents the decomposition of MSE, we believe the reduced MSE by the proposed methods is due to the bias-variance trade-off.

3. Simulation studies

We compare the performance of our method to those of various ML quanitle regression methods. We consider two scenarios:

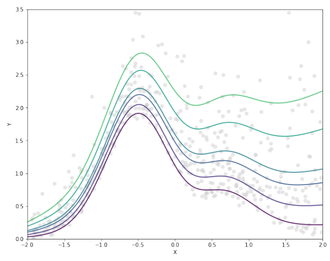
(Scenario 1) We generate data following the Example 2 in Cannon (2018).

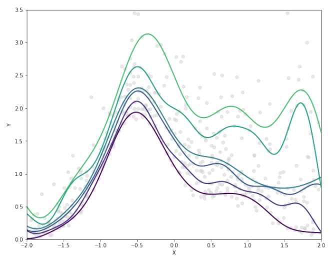
$$y = \left(1 - 2x + 4x^2\right) \exp\left(-2x^2\right) + (0.2 + 0.08x)\,\epsilon,$$

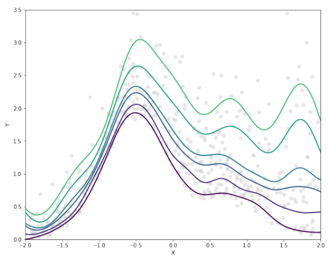
where x is randomly drawn from $U(-2, 2)$. We consider $\chi^2(3)$ for the distribution of ϵ.

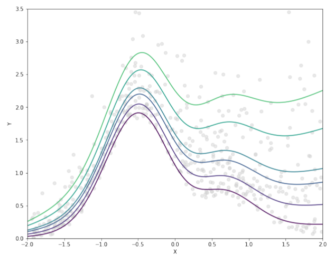
(Scenario 2) We modify the Scenario 4 in Das and Ghosal (2018).

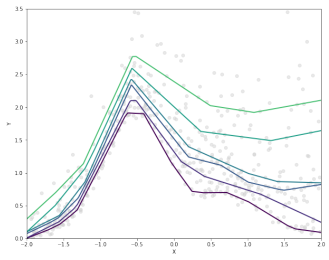
$$y = \sin(\pi x_1) + \cos(\pi x_2) + \left(x_1^2 + 1.5x_2^2 - 0.5x_2\right)\epsilon,$$

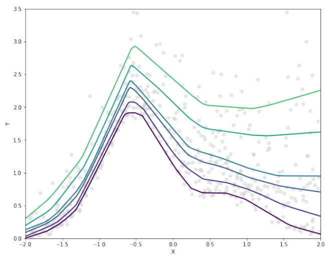

(a) *True quantile;* $\epsilon \sim \chi^2(3)$

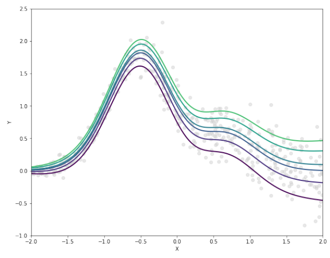

(b) *SVQR-SQ fit*

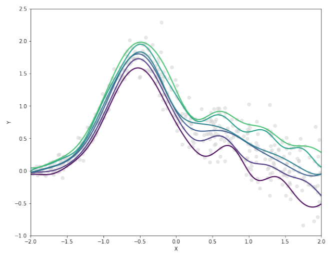

(c) *SVQR-LQE fit*

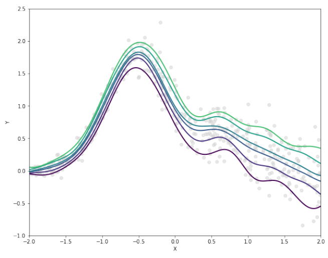

(d) *True quantile;* $\epsilon \sim \chi^2(3)$

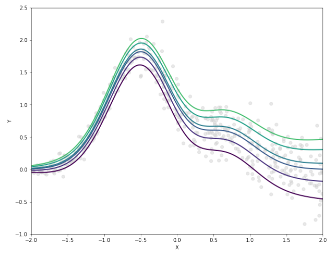

(e) *QSS-SQ fit*

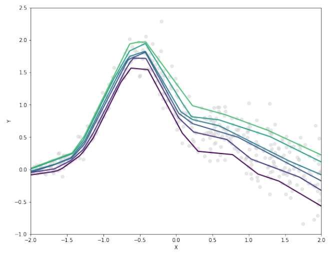

(f) *QSS-LQE fit*


(g) *True quantile;* $\epsilon \sim N(0,1)$


(h) *SVQR-SQ fit*


(i) *SVQR-LQE fit*


(j) *True quantile;* $\epsilon \sim N(0,1)$


(k) *QSS-SQ fit*

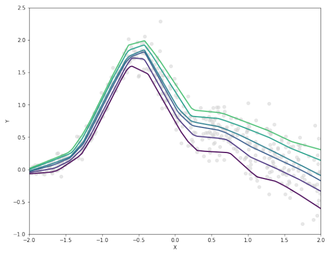

(l) *QSS-LQE fit*

Figure 1: *Fitted curves under Scenario 1 with quantile smoothing spline (QSS) and support vector quantile regression (SVQR). Each curve corresponds to the quantile fits at* $\tau = 0.1, 0.3, 0.5, 0.6, 0.8,$ *and* 0.9 *from bottom to the top.*

where the bivariate covariates $\mathbf{x} = [x_1, x_2]$ are randomly drawn from $U(0,1) \times U(0,1)$. The error term ϵ follows a t-distribution with 4 degrees of freedom.

Note that the generated data have a nonlinear trend and heteroscedastic errors in both scenarios. For the first scenario, we mainly focus on the error with the chi-square distribution, but we additionally

Table 2: The mean of MSEs at $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$, and 0.9 and crossing losses (with their standard error in the parenthesis) for 100 iterations under Scenario 1

Model	Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$	Crossing loss
GBM	SQ	0.9128	0.8247	0.9824	1.1756	2.1746	4.3652	0.5767
		(0.0670)	(0.0313)	(0.0361)	(0.0401)	(0.0961)	(0.3231)	(0.2082)
	LQE	**0.8615**	**0.6329**	0.8872	1.0994	2.0980	**3.7406**	**0.2325**
		(0.0289)	(0.0214)	(0.0348)	(0.0464)	(0.1276)	(0.2817)	(0.0843)
	LQE-S	0.9434	0.6941	**0.8502**	**1.0659**	**1.9340**	3.9925	0.2607
		(0.0346)	(0.0211)	(0.0333)	(0.0440)	(0.0965)	(0.3169)	(0.1137)
XGBoost	SQ	0.6122	2.3963	1.4581	2.2915	4.1964	8.8616	0.1364
		(0.0254)	(0.0990)	(0.0610)	(0.1019)	(0.2721)	(0.5117)	(0.0264)
	LQE	**0.5490**	0.8078	1.4378	**1.7092**	**4.1071**	**5.3745**	**0.0077**
		(0.0168)	(0.0272)	(0.0576)	(0.0775)	(0.2093)	(0.2964)	(0.0029)
	LQE-S	0.7356	**0.7546**	**1.3214**	1.8151	4.3839	6.7670	0.0731
		(0.0192)	(0.0264)	(0.0570)	(0.0889)	(0.2077)	(0.3330)	(0.0460)
LightGBM	SQ	0.5568	0.8507	1.1962	1.6267	3.1880	4.5249	0.2381
		(0.0207)	(0.0298)	(0.0441)	(0.0635)	(0.1663)	(0.2877)	(0.0416)
	LQE	**0.4718**	0.5776	**0.9226**	**1.2125**	**2.1504**	3.9118	**0.0029**
		(0.0177)	(0.0183)	(0.0363)	(0.0487)	(0.0956)	(0.2206)	(0.0014)
	LQE-S	0.5567	**0.5623**	0.9430	1.2563	2.3903	**3.6402**	0.0034
		(0.0214)	(0.0161)	(0.0386)	(0.0515)	(0.1302)	(0.2037)	(0.0018)
QRF	SQ	0.6046	1.0282	1.7211	2.0594	2.7853	6.7809	0.0000
		(0.0224)	(0.0371)	(0.0808)	(0.0921)	(0.1238)	(0.3763)	(0.0000)
	LQE	**0.4912**	0.9490	**1.4449**	**1.7419**	2.7542	**5.6504**	0.0000
		(0.0178)	(0.0338)	(0.0603)	(0.0779)	(0.1480)	(0.3464)	(0.0000)
	LQE-S	0.5124	**0.9459**	1.5895	1.9040	**2.7481**	6.3088	0.0000
		(0.0180)	(0.0339)	(0.0683)	(0.0873)	(0.1446)	(0.3743)	(0.0000)
QERT	SQ	0.4828	0.5415	0.9956	1.4259	3.6139	7.3510	0.0000
		(0.0149)	(0.0185)	(0.0416)	(0.0531)	(0.1696)	(0.3890)	(0.0000)
	LQE	**0.3988**	0.5139	0.9101	**1.2915**	**3.2943**	**7.0852**	0.0000
		(0.0122)	(0.0182)	(0.0407)	(0.0585)	(0.1748)	(0.3762)	(0.0000)
	LQE-S	0.4218	**0.4504**	**0.8337**	1.3196	3.4947	7.2023	0.0000
		(0.0125)	(0.0168)	(0.0397)	(0.0601)	(0.1763)	(0.3730)	(0.0000)

The crossing loss is calculated for τ'=0.1 and τ''=0.3. The smallest values for each model are bolded for clarity. MSEs are multiplied by 10^2, and the crossing loss by 10^3.

consider the standard normal distribution for the graphical demonstration. For the second scenario, the original paper considered Gaussian errors, but we generate errors with a heavy-tailed distribution. The target quantiles are $\tau = 0.1$, 0.3, 0.5, 0.6, 0.8, and 0.9.

Among ML algorithms capable of estimating conditional quantiles, we select GBM, XGBoost, LightGBM, QRF, and QERT. In addition, we fit QSS and SVQR for the visualization of Scenario 1 in Figure 1. We use R 3.5.1 program for QSS and SVQR, and Python 3.11 for the others. QSS is implementable with the `rqss` function in the package `quantreg` (Koenker *et al.*, 2018). The `kqr` function from the package `kernlab` is applicable to SVQR. We can fit QRF and QERT using the functions `RandomForestQuantileRegressor` and `ExtraTreesQuantileRegressor` in the python package `sklearn-quantile` (Roebroek, 2022). We use the `GradientBoostingRegressor` function in `scikit-learn` (Pedregosa *et al.*, 2011) for GBM, the `lightgbm` package (Ke *et al.*, 2017) for LightGBM, and the `xgboost` package (Chen and Guestrin, 2016) for the XGBoost model. The LQE method can be easily implemented using existing packages.

We compare three models: Single quantile (SQ) and two versions of local quantile ensemble (LQE). The SQ is a genuine model with a single target quantile. LQE combines $K = 9$ local quantiles.

To determine the magnitude of the local, we use Rule 1 in Section 2.2. Note that LQE requires to tune each of the K model. This increases the computing cost by K times. To reduce the tuning process, we consider a simpler version. First, we find the optimal parameter for the target quantile only. Then, we apply it to all the other local quantile models. The simpler one, LQE-S, needs a similar tuning time to SQ. Although the tuning time for LQE-S is much shorter than that of LQE, its results are quite comparable.

We split the generated data into training and test samples. The training data is used for model fitting and validation for parameter tuning, and the test data is used for measuring the prediction accuracy. In Scenario 1, the training and test sample sizes are 500 and 1000, respectively. In Scenario 2, we utilize a data set of 2000 samples for training and 5000 samples for the test. The simulations are repeated 100 times.

We use two criteria for evaluation. First, we calculate a mean square error (MSE) between the true quantile function and the predicted quantile fit over the test data, as follows:

$$\text{MSE}(\tau) := \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Q}_{\tau}(y \mid \mathbf{x}_i) - Q_{\tau}(y \mid \mathbf{x}_i)\right)^2. \tag{3.1}$$

This is a well-known indicator of the estimation accuracy. Next, we employ the crossing loss (Sangnier *et al.*, 2016) to measure the undesirable quantile crossings. It is defined as

$$\text{Crossing loss} := \frac{1}{n}\sum_{i=1}^{n}\max\left(0, \hat{Q}_{\tau'}(y \mid \mathbf{x}_i) - \hat{Q}_{\tau''}(y \mid \mathbf{x}_i)\right), \quad \text{where } 0 < \tau' < \tau'' < 1. \tag{3.2}$$

The crossing loss is nonzero when the quantile estimate for τ' exceeds that for τ'' quantile. We record the crossing loss for only boosting models because QRF and QERT innately prevent quantile crossing. The crossing losses for these models are all zero.

We select the parameters through a grid search. The chosen parameters are the empirical minimizers of the mean check loss from 10-fold cross-validation. The range of parameters and their description are given in the appendix. We search for the same parameter space for SQ, LQE, and LQE-S within each ML model for a fair comparison.

Tables 2 and 3 show the results for Scenarios 1 and 2. Regardless of the ML model employed, LQE generally exhibits a lower MSE compared to SQ. It suggests that the ensemble effect of LQE results in improved performance. LQE-S, while slightly behind LQE in some cases, still often outperforms SQ, making it a viable option for scenarios where computational resources and time are constrained. The benefit of the proposed method is reassured in crossing losses. In most cases, LQE and/or LQE-S show much smaller crossing losses than SQ. In general, LQE provide a stabler fit which tends to keep the fits from crossing.

For graphical illustration, we apply quantile smoothing splines (QSS) and support vector quantile regression (SVQR) to Scenario 1. The algorithms for them are in Koenker and Portnoy (1994), Takeuchi *et al.* (2006), and Li *et al.* (2007). Figure 1 displays the fitted curves of SQ and LQE. Quantile crossings sometimes occur in SQ near the boundary, whereas they are rare in the proposed method. The numerical summary of Figure 1 is detailed in the appendix. Figure 2 illustrates the fitted curves of ML models in Scenario 2. LQE displays a slightly stabler surface than SQ.

4. Real data analysis

In this section, we analyze three real data sets. We select data from three different fields. All data sets used here are available in the UCI machine learning repository. We briefly introduce each data set.

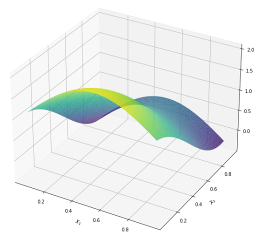
Table 3: The mean of MSEs at $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$, and 0.9 and crossing losses (with their standard error in the parenthesis) for 100 iterations under Scenario 2. The crossing loss is calculated for $\tau' = 0.1$ and $\tau'' = 0.3$. The smallest values for each model are bolded for clarity. MSEs are multiplied by 10^2, and the crossing loss by 10^3.

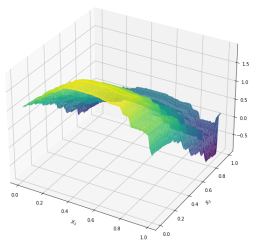
Model	Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$	Crossing Loss
GBM	SQ	0.7085	0.2768	0.1832	0.1758	0.2183	0.4162	5.4292
		(0.0266)	(0.0079)	(0.0054)	(0.0057)	(0.0097)	(0.0160)	(0.4368)
	LQE	**0.6078**	**0.2197**	0.1546	0.1650	0.2229	**0.3960**	**0.9908**
		(0.0234)	(0.0063)	(0.0046)	(0.0055)	(0.0096)	(0.0168)	(0.0866)
	LQE-S	0.6178	0.2246	**0.1513**	**0.1647**	**0.2220**	0.4083	1.0558
		(0.0241)	(0.0064)	(0.0047)	(0.0055)	(0.0095)	(0.0178)	(0.0946)
XGBoost	SQ	0.8727	0.4128	0.3267	0.3052	0.3692	0.6839	7.9461
		(0.0313)	(0.0129)	(0.0108)	(0.0112)	(0.0185)	(0.0408)	(0.6017)
	LQE	0.8157	0.3315	0.2598	0.2522	0.3336	0.5120	1.3189
		(0.0265)	(0.0082)	(0.0068)	(0.0073)	(0.0132)	(0.0286)	(0.0679)
	LQE-S	**0.7101**	**0.3049**	**0.2299**	**0.2336**	**0.2858**	**0.5117**	**1.1192**
		(0.0216)	(0.0081)	(0.0068)	(0.0072)	(0.0118)	(0.0294)	(0.0714)
LightGBM	SQ	0.8783	0.3640	0.2837	0.2920	0.2385	0.4435	5.3904
		(0.0306)	(0.0103)	(0.0071)	(0.0079)	(0.0082)	(0.0151)	(0.2655)
	LQE	0.7154	**0.2793**	**0.1975**	**0.2114**	0.2502	0.4177	**0.8135**
		(0.0224)	(0.0071)	(0.0053)	(0.0060)	(0.0082)	(0.0169)	(0.0563)
	LQE-S	**0.6936**	0.2888	0.2030	0.2289	**0.2175**	**0.4027**	0.8455
		(0.0208)	(0.0073)	(0.0052)	(0.0063)	(0.0079)	(0.0156)	(0.0621)
QRF	SQ	**1.7994**	0.5771	0.4259	0.5258	0.8837	**1.5439**	0.0000
		(0.0529)	(0.0165)	(0.0100)	(0.0138)	(0.0290)	(0.0515)	(0.0000)
	LQE	2.1930	0.5625	0.4467	**0.4789**	**0.7876**	1.8936	0.0000
		(0.0640)	(0.0147)	(0.0098)	(0.0110)	(0.0226)	(0.0577)	(0.0000)
	LQE-S	2.0693	**0.5581**	**0.4185**	0.5352	0.9277	1.7124	0.0000
		(0.0605)	(0.0147)	(0.0091)	(0.0131)	(0.0276)	(0.0557)	(0.0000)
QERT	SQ	**1.1710**	0.3560	0.2741	0.2916	0.4417	1.2037	0.0000
		(0.0309)	(0.0088)	(0.0067)	(0.0065)	(0.0107)	(0.0845)	(0.0000)
	LQE	1.8633	**0.3027**	**0.2249**	0.2605	0.4820	1.9316	0.0000
		(0.0470)	(0.0064)	(0.0053)	(0.0063)	(0.0118)	(0.0459)	(0.0000)
	LQE-S	1.2676	0.3063	0.2260	**0.2574**	**0.4209**	**0.9353**	0.0000
		(0.0345)	(0.0071)	(0.0057)	(0.0062)	(0.0103)	(0.0275)	(0.0000)

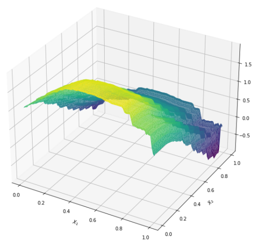
Bike: The `bike-sharing` data contains information on the number of bike rentals and the weather by day or hour gathered from the bike-sharing system. Among these, we use the data collected per day. Fanaee-T and Gama (2014) first analyzed this data, and Schallhorn *et al.* (2017) used D-vine quantile regression with it. The objective is to solve a regression problem predicting the total number of bikes rented per day. After excluding variables linearly dependent on the total number of rentals and the date variable, there are $p = 11$ variables and $n = 731$ observations. We randomly select 500 samples as training data and the others as test data.

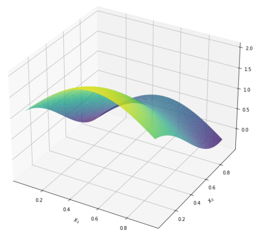
Concrete: We analyze the `concrete` data to predict the compressive strength of the concrete based on its age and ingredients. The data was first proposed by Yeh (1998). Romano *et al.* (2019) conducted conformal inference using quantile regression. It contains $n = 1030$ samples, with $p = 7$ covariates. All covariates are continuous variables, such as the amount of fly ash and water in the concrete. We randomly divide the data into 700 training samples and 330 test samples.

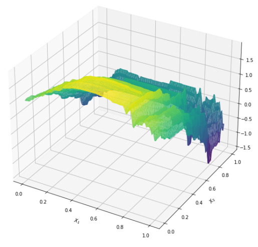
Blog Feedback: The `blog-feedback` data was gathered by crawling and analyzing the raw HTML documents of blog posts. Buza (2013) initially collected the posts from 2010 to March 2012. Feldman *et al.* (2021) examined it to create prediction intervals using orthogonal quantile regression, and Alaa

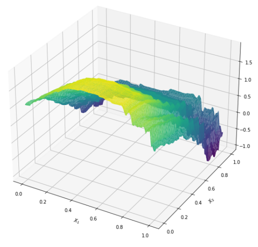

(a) *True quantile at* $\tau = 0.6$

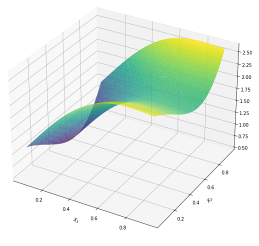

(b) *GBM-SQ fit at* $\tau = 0.6$

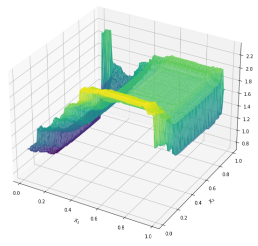

(c) *GBM-LQE fit at* $\tau = 0.6$

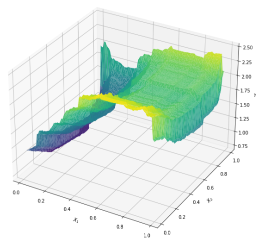

(d) *True quantile at* $\tau = 0.6$

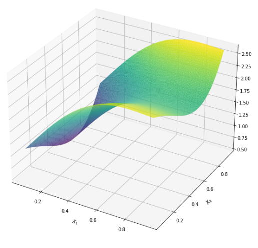

(e) *LGBM-SQ fit at* $\tau = 0.6$

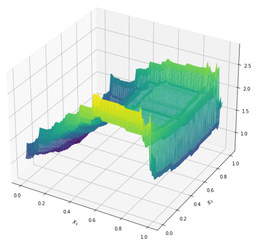

(f) *LGBM-LQE fit at* $\tau = 0.6$


(g) *True quantile at* $\tau = 0.9$


(h) *GBM-SQ at* $\tau = 0.9$


(i) *GBM-LQE at* $\tau = 0.9$


(j) *True quantile at* $\tau = 0.9$


(k) *LGBM-SQ at* $\tau = 0.9$

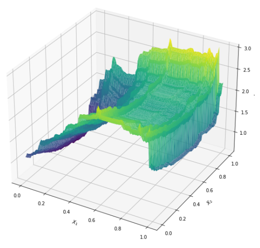

(l) *LGBM-LQE at* $\tau = 0.9$

Figure 2: *True conditional quantile surfaces and fitted ones for* $\tau = 0.6$ *and 0.9 under Scenario 2.*

et al. (2023) analyzed it with conformalized unconditional quantile regression. We aim to predict the number of comments each post will receive during the next day. The data was already split into training and test sets, but we combine them. Final data has a dimension of $n = 60,021$ and $p = 280$. Examples of covariates include the number of characters in a blog post and the date and time of publication. We randomly select 10,000 observations as training data and used the remaining

Table 4: The mean of check losses at $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$ and 0.9 and crossing losses (with the standard error in the parenthesis) for 100 iterations with `bike-sharing` data

Model	Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$	Crossing loss
GBM	SQ	1.4109 (0.0138)	1.9793 (0.0150)	2.0331 (0.0191)	1.9369 (0.0114)	1.4588 (0.0100)	0.9714 (0.0088)	0.6532 (0.0344)
	LQE	**1.3878** (0.0131)	1.9768 (0.0144)	2.0222 (0.0123)	**1.9103** (0.0114)	**1.4198** (0.0096)	0.9550 (0.0084)	0.1821 (0.0139)
	LQE-S	1.3881 (0.0130)	**1.9679** (0.0148)	**2.0120** (0.0120)	1.9151 (0.0116)	1.4425 (0.0097)	**0.9540** (0.0084)	**0.1445** (0.0109)
XGBoost	SQ	1.4411 (0.0174)	2.0528 (0.0162)	2.1270 (0.0120)	2.0197 (0.0105)	1.5304 (0.0097)	1.0386 (0.0086)	2.1641 (0.0772)
	LQE	**1.3503** (0.0147)	**1.9798** (0.0151)	**2.0358** (0.0123)	**1.9461** (0.0109)	**1.4528** (0.0092)	**0.9765** (0.0083)	**0.3310** (0.0216)
	LQE-S	1.3713 (0.0158)	1.9853 (0.0156)	2.0478 (0.0120)	1.9530 (0.0107)	1.4716 (0.0097)	0.9964 (0.0079)	0.4635 (0.0287)
LightGBM	SQ	1.4514 (0.0172)	2.1141 (0.0159)	2.2010 (0.0135)	2.0838 (0.0133)	1.5113 (0.0102)	1.0144 (0.0086)	1.5259 (0.0547)
	LQE	**1.3994** (0.0149)	**2.0800** (0.0149)	**2.1565** (0.0135)	**2.0405** (0.0124)	**1.4799** (0.0098)	**0.9737** (0.0082)	**0.2694** (0.0175)
	LQE-S	1.4222 (0.0164)	2.0832 (0.0149)	2.1582 (0.0136)	2.0473 (0.0125)	1.4845 (0.0098)	0.9837 (0.0085)	0.3926 (0.0225)
QRF	SQ	1.3385 (0.0107)	2.1736 (0.0150)	2.2686 (0.0150)	2.1528 (0.0132)	1.5254 (0.0092)	0.9540 (0.0058)	0.0000 (0.0000)
	LQE	**1.3378** (0.0098)	**2.1500** (0.0143)	**2.2159** (0.0143)	**2.1038** (0.0131)	**1.5116** (0.0091)	**0.9536** (0.0057)	0.0000 (0.0000)
	LQE-S	1.3403 (0.0096)	2.1514 (0.0142)	2.2197 (0.0144)	2.1069 (0.0132)	1.5141 (0.0091)	0.9551 (0.0056)	0.0000 (0.0000)
QERT	SQ	1.3224 (0.0105)	2.2035 (0.0124)	2.3237 (0.0121)	2.1968 (0.0102)	1.5522 (0.0071)	0.9686 (0.0051)	0.0000 (0.0000)
	LQE	1.3041 (0.0088)	2.1618 (0.0118)	2.2528 (0.0114)	2.1407 (0.0103)	1.5235 (0.0071)	0.9582 (0.0048)	0.0000 (0.0000)
	LQE-S	**1.2734** (0.0111)	**2.1062** (0.0148)	**2.2014** (0.0143)	**2.0907** (0.0129)	**1.4892** (0.0094)	**0.9325** (0.0063)	0.0000 (0.0000)

The check loss values are divided by 10^2 and crossing losses are divided by 10.

observations as test data.

We fit the quantile ML models discussed in the previous section. The number of local quantiles, K, is 9 for LQE and LQE-S. All model fitting procedures are similar to those in Section 3. We search for the parameters in each model using a grid search. The details of the parameter spaces we search is in the appendix.

We measure the mean check loss, defined as

$$\frac{1}{n}\sum_{i=1}^{n} L_\tau(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \{\tau - I(y_i < \hat{y}_i)\}, \quad \text{where} \quad \hat{y}_i = \hat{Q}_\tau(y \mid \mathbf{x}_i). \tag{4.1}$$

Tables 4, 5 and 6 summarize the check loss and crossing loss of the compared models from 100 random splits. The performance of LQE and LQE-S are frequently superior to that of SQ. These results demonstrate the proposed method can enhance the accuracy of predictions. LQE yields much lower crossing losses in most cases. LQE seems to suppress quantile crossing without additional constraints.

To demonstrate the computation efficiency of LQE-S, we also measure the computation time required for the parameter tuning process using the GBM model and the `bike` dataset. Because both

Table 5: The mean of check losses at $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$ and 0.9 and crossing losses (with the standard error in the parenthesis) for 100 iterations with `concrete` data

Model	Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$	Crossing loss
GBM	SQ	0.9393	1.3099	**1.3541**	1.3378	1.2397	0.9691	0.7125
		(0.0059)	(0.0089)	(0.0093)	(0.0098)	(0.0106)	(0.0081)	(0.0233)
	LQE	**0.9130**	1.3149	1.3681	1.3469	**1.2070**	**0.9319**	**0.2193**
		(0.0056)	(0.0079)	(0.0087)	(0.0094)	(0.0096)	(0.0080)	(0.0094)
	LQE-S	0.9394	**1.3096**	1.3576	**1.3345**	1.2213	0.9521	0.3727
		(0.0054)	(0.0085)	(0.0088)	(0.0097)	(0.0101)	(0.0078)	(0.0140)
XGBoost	SQ	0.9269	1.3585	1.4433	1.4573	1.2710	0.9873	1.4716
		(0.0073)	(0.0100)	(0.0095)	(0.0098)	(0.0115)	(0.0103)	(0.0441)
	LQE	**0.8794**	**1.3208**	**1.4166**	**1.4015**	**1.2100**	**0.9300**	**0.3018**
		(0.0063)	(0.0086)	(0.0088)	(0.0094)	(0.0103)	(0.0090)	(0.0203)
	LQE-S	0.8890	1.3382	1.4244	1.4028	1.2168	0.9491	0.3693
		(0.0065)	(0.0088)	(0.0090)	(0.0093)	(0.0104)	(0.0097)	(0.0134)
LightGBM	SQ	0.9037	1.4068	1.5828	1.5761	1.2871	0.9699	1.1566
		(0.0068)	(0.0088)	(0.0090)	(0.0102)	(0.0107)	(0.0096)	(0.0315)
	LQE	**0.8718**	1.3946	**1.5679**	**1.5457**	1.2633	**0.9314**	**0.1341**
		(0.0059)	(0.0082)	(0.0090)	(0.0095)	(0.0099)	(0.0085)	(0.0073)
	LQE-S	0.8825	**1.3933**	1.5704	1.5473	**1.2598**	0.9514	0.3752
		(0.0067)	(0.0082)	(0.0090)	(0.0095)	(0.0101)	(0.0090)	(0.0143)
QRF	SQ	**0.9412**	1.6482	**1.8027**	1.7678	1.3877	0.9458	0.0000
		(0.0056)	(0.0088)	(0.0102)	(0.0095)	(0.0088)	(0.0058)	(0.0010)
	LQE	0.9422	1.6412	1.8115	1.7577	**1.3768**	0.9460	0.0000
		(0.0053)	(0.0087)	(0.0098)	(0.0094)	(0.0079)	(0.0053)	(0.0000)
	LQE-S	0.9422	**1.6393**	1.8092	**1.7573**	1.3774	**0.9458**	0.0000
		(0.0053)	(0.0086)	(0.0096)	(0.0093)	(0.0080)	(0.0053)	(0.0000)
QERT	SQ	0.8872	1.5602	1.7124	1.6650	1.3060	0.9068	0.0000
		(0.0059)	(0.0082)	(0.0101)	(0.0099)	(0.0092)	(0.0069)	(0.000)
	LQE	0.8872	1.5435	**1.6739**	1.6268	**1.2880**	0.9032	0.0000
		(0.0057)	(0.0081)	(0.0095)	(0.0099)	(0.0086)	(0.0061)	(0.0000)
	LQE-S	**0.8869**	**1.5432**	1.6751	**1.6265**	1.2883	**0.9031**	0.0000
		(0.0057)	(0.0081)	(0.0095)	(0.0099)	(0.0086)	(0.0061)	(0.0000)

All crossing loss values are multiplied by 10.

SQ and LQE-S tune the target quantile only, their tuning times are roughly equivalent; thus, we only compare LQE and LQE-S. As shown in Table 7, LQE-S reduces the tuning time by approximately $K = 9$ times compared to LQE. This result highlights the substantial reduction in computing time achieved by LQE-S.

5. Conclusion

We introduce an ensemble learning method to improve quantile estimation and prediction of the existing machine learning (ML) models. LQE fits multiple local quantiles around a target and averages them. One can adapt LQE to a broad class of quantile regression models. It enhances performance by mitigating the large variance inherent in QR through the ensemble effect. Furthermore, LQE prevents overfitting and quantile crossing, as well as providing smoother fits.

However, LQE can be time-consuming due to tuning multiple parameters. To alleviate computing cost, we propose LQE-S, which tunes the parameter for the target quantile only and applies them to all other local quantiles. Our numerical analysis demonstrates that the practical performance of LQE and LQE-S are similar. In many cases, they achieve lower losses than single quantile (SQ) models.

Table 6: The mean of check losses at $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$ and 0.9 and crossing losses (with the standard error in the parenthesis) for 100 iterations with `blog-feedback` data

Model	Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$	Crossing loss
	SQ	0.6587	1.8637	2.6291	2.8722	2.8243	2.3605	0.0419
		(0.0007)	(0.0038)	(0.0040)	(0.0053)	(0.0055)	(0.0058)	(0.0010)
GBM	LQE	0.6594	**1.8384**	2.6264	2.8433	**2.7719**	2.3181	**0.0120**
		(0.0007)	(0.0025)	(0.0043)	(0.0051)	(0.0047)	(0.0045)	(0.0003)
	LQE-S	**0.6585**	1.8410	**2.6173**	**2.8325**	2.7943	**2.3046**	0.0217
		(0.0007)	(0.0032)	(0.0045)	(0.0049)	(0.0047)	(0.0052)	(0.0005)
	SQ	0.6472	1.7298	**2.4075**	2.5948	2.5811	2.0892	0.0042
		(0.0007)	(0.0028)	(0.0035)	(0.0038)	(0.0050)	(0.0050)	(0.0002)
XGBoost	LQE	**0.6447**	**1.7147**	2.4136	2.5957	**2.4986**	**2.0545**	**0.0008**
		(0.0007)	(0.0023)	(0.0033)	(0.0035)	(0.0042)	(0.0047)	(0.0000)
	LQE-S	0.6451	1.7168	2.4099	**2.5916**	2.5394	2.0566	0.0009
		(0.0007)	(0.0023)	(0.0033)	(0.0035)	(0.0045)	(0.0049)	(0.0000)
	SQ	0.6476	1.7415	2.4428	2.6361	2.5567	2.1662	0.0199
		(0.0009)	(0.0025)	(0.0031)	(0.0042)	(0.0044)	(0.0051)	(0.0005)
LightGBM	LQE	**0.6441**	**1.7219**	**2.4136**	**2.5923**	**2.4956**	**2.0526**	0.0081
		(0.0008)	(0.0022)	(0.0031)	(0.0035)	(0.0042)	(0.0045)	(0.0002)
	LQE-S	0.6447	1.7235	2.4153	2.5979	2.4960	2.0611	**0.0080**
		(0.0008)	(0.0025)	(0.0031)	(0.0036)	(0.0041)	(0.0047)	(0.0002)
	SQ	0.6562	1.7598	2.5075	**2.6787**	2.7937	2.3508	0.0000
		(0.0006)	(0.0035)	(0.0044)	(0.0045)	(0.0057)	(0.0044)	(0.0000)
QRF	LQE	**0.6507**	**1.7417**	**2.4938**	2.7328	**2.7427**	**2.3342**	0.0000
		(0.0007)	(0.0027)	(0.0038)	(0.0042)	(0.0044)	(0.0033)	(0.0000)
	LQE-S	0.6562	1.7727	2.5376	2.7047	2.7953	2.3549	0.0000
		(0.0007)	(0.0031)	(0.0046)	(0.0040)	(0.0050)	(0.0040)	(0.0000)
	SQ	0.6572	1.8804	**2.7714**	**3.0800**	**3.2064**	**2.6951**	0.0000
		(0.0006)	(0.0030)	(0.0048)	(0.0052)	(0.0052)	(0.0046)	(0.0000)
QERT	LQE	**0.6566**	**1.8664**	2.8146	3.1106	3.2092	2.7455	0.0000
		(0.0006)	(0.0026)	(0.0041)	(0.0047)	(0.0049)	(0.0038)	(0.0000)
	LQE-S	0.6571	1.8687	2.8138	3.1087	3.2153	2.7524	0.0000
		(0.0066)	(0.0027)	(0.0041)	(0.0046)	(0.0045)	(0.0037)	(0.0000)

All crossing loss values are multiplied by 10^2.

Table 7: The mean of computation time (with the standard error in the parenthesis) for 5 iterations

Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$
LQE	32.82 (0.09)	38.30 (0.10)	39.94 (0.12)	40.92 (0.07)	38.53 (0.09)	35.42 (0.09)
LQE-S	3.66 (0.02)	4.32 (0.01)	4.60 (0.01)	4.61 (0.03)	4.33 (0.01)	3.97 (0.01)

Moreover, the reduction in quantile crossing is evident. The wiggly fits and quantile crossings from the existing methods often significantly reduced.

This paper highlights the effect of ensembles near the target quantiles. We anticipate that LQE will perform effectively for another machine learning and deep learning models as well, which is an interesting area of our future research. In fact, the idea of LQE can be applied to the linear quantile regression models, and we developed early stage of theories.

Acknowledgement

Yoonsuh Jung's work was partially supported by National Research Foundation of Korea (NRF) grant funded by Korea government (MIST)(No. 2022M3J6A1063595 and No. 2022R1F1A1071126).

Appendix A:

Table A.1: The mean of MSEs at $\tau = 0.1, 0.3, 0.5, 0.6, 0.8$, and 0.9 and crossing losses (with their standard error in the parenthesis) for 100 iterations under Scenario 1

Error	Model	Method	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 0.9$	Crossing loss
$\chi^2(3)$	QSS	SQ	0.2052 (0.0121)	0.4101 (0.0210)	0.6479 (0.0345)	0.7735 (0.0429)	1.9008 (0.1272)	3.3899 (0.2110)	0.0149 (0.0052)
		LQE	0.1623 (0.0078)	0.3019 (0.0172)	0.5834 (0.0338)	0.8083 (0.0479)	1.7572 (0.1289)	3.3624 (0.2347)	0.0000 (0.0000)
		LQE-S	0.1609 (0.0075)	0.3332 (0.0155)	0.5848 (0.0337)	0.7888 (0.0470)	1.8121 (0.1257)	3.3386 (0.2308)	0.0000 (0.0000)
	KQR	SQ	0.2136 (0.0164)	0.6109 (0.0330)	1.0233 (0.0554)	0.9323 (0.0537)	3.2260 (0.0961)	4.4850 (0.2688)	0.0243 (0.0125)
		LQE	0.1811 (0.0128)	0.4587 (0.0253)	0.7616 (0.0439)	1.0328 (0.0589)	2.1058 (0.1260)	4.3697 (0.2677)	0.0000 (0.0000)
		LQE-S	0.1578 (0.0107)	0.4846 (0.0255)	0.8431 (0.0502)	0.8401 (0.0481)	2.7638 (0.1600)	3.9785 (0.2732)	0.0003 (0.0002)
$N(0, 1)$	QSS	SQ	0.3305 (0.0136)	0.2083 (0.0103)	0.1939 (0.0091)	0.1993 (0.0078)	0.2296 (0.0100)	0.3078 (0.0182)	0.0038 (0.0025)
		LQE	0.2642 (0.0122)	0.1621 (0.0078)	0.1423 (0.0058)	0.1529 (0.0060)	0.1913 (0.0081)	0.2645 (0.0148)	0.0000 (0.0000)
		LQE-S	0.2661 (0.0122)	0.1606 (0.0079)	0.1631 (0.0064)	0.1535 (0.0061)	0.1928 (0.0085)	0.2687 (0.0153)	0.0000 (0.0000)
	KQR	SQ	0.5124 (0.0243)	0.2645 (0.0136)	0.1623 (0.0083)	0.2261 (0.0114)	0.3335 (0.0131)	0.4203 (0.0215)	0.0040 (0.0018)
		LQE	0.3561 (0.0179)	0.2138 (0.0115)	0.1417 (0.0070)	0.1451 (0.0067)	0.1999 (0.0092)	0.3056 (0.0144)	0.0000 (0.0000)
		LQE-S	0.1578 (0.0107)	0.4846 (0.0255)	0.8431 (0.0502)	0.8401 (0.0481)	2.7638 (0.1600)	3.9785 (0.2732)	0.0003 (0.0002)

The crossing loss is calculated for τ'=0.1 and τ''=0.3. The smallest values for each model are bolded for clarity. MSEs are multiplied by 10^2, and the crossing loss by 10^3.

Table A.2: The description of the parameters used for tuning in each ML models

Model	Parameter	Descriptions
GBM	n_estimators	The number of boosting stages to perform
	max_depth	Maximum depth of a tree
	subsample	The fraction of samples for fitting the individual trees
	learning_rate	Step size shrinkage used in update
	max_features	The number of features to consider in each splits
XGBoost	nrounds	The number of boosting stages to perform
	max_depth	Maximum depth of a tree
	min_child_weight	Minimum sum of instance weight
	gamma	Minimum loss reduction required to make a partition
	lambda	L2 regularization term on weights
	eta	Step size shrinkage used in update
LightGBM	nrounds	The number of boosting stages to perform
	max_depth	Maximum depth of a tree
	num_leaves	Maximum number of leaves in one tree
	learning_rate	Step size shrinkage used in update
QRF & QERT	n_estimators	The number of trees in the forest
	min_samples_split	Minimum number of samples required to split
	max_depth	Maximum depth of a tree
	max_features	The number of features to consider in each splits

Table A.3: The parameter search range for simulations

Model	Parameter	Scenario 1	Scenario 2
GBM	n_estimators	(100, 1700)	(100, 1500)
	max_depth	[1, 2, 3, 4]	[1, 2, 3, 4]
	subsample	[0.5, 0.7, 0.9]	[0.5, 0.7, 0.9]
	learning_rate	[0.01]	[0.01]
	max_features	[1]	[2]
XGBoost	nrounds	(2000, 5000)	(2000, 5000)
	max_depth	[1, 2, 3, 4]	(1, 2, 3, 4)
	min_child_weight	[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
	gamma	[0, 0.001, 0.01, 0.1]	[0, 0.001, 0.01, 0.1]
	lambda	[0, 0.01, 0.1, 1]	[0, 0.01, 0.1, 1]
	eta	[0.01]	[0.01]
LightGBM	nrounds	(200, 1800)	(400, 1600)
	max_depth	[2, 4, 6, 8, 10]	[1, 2, 3, 4, 5, 6]
	num_leaves	[2, 4, 8, 16, 32]	[2, 4, 6, 8, 10, 12, 14]
	learning_rate	[0.01]	[0.01]
QRF	n_estimators	(200, 1200)	(800, 3000)
	min_samples_split	[1, 10, 20, 30, 40]	[10, 20, 30, 40]
	max_depth	[2, 4, 6, 8, 10]	[2, 6, 10, 14, 18]
	max_features	[1]	[1, 2]
QERT	n_estimators	(200, 1200)	(800, 3000)
	min_samples_split	[1, 10, 20, 30, 40]	[10, 20, 30, 40]
	max_depth	[2, 4, 6, 8, 10]	[2, 6, 10, 14, 18]
	max_features	[1]	[1, 2]

The terms inside square brackets indicate discrete values, and the terms in the round brackets indicate a sequence of numbers.

Table A.4: The parameter search range for the data used in real data analysis

Model	Parameter	Bike	Concrete	Blog feedback
GBM	n_estimators	[1000, 2000, 3000, 4000]	[2000, 4000, 6000, 8000]	[1000, 2000, 3000, 4000]
	max_depth	[2, 4, 6]	[2, 4, 6, 8]	[2, 4, 6]
	subsample	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]
	learning_rate	[0.01]	[0.01]	[0.1]
	max_features	[4, 6, 8]	[4, 6, 8]	[4, 6, 8]
XGBoost	nrounds	(2000, 6000)	(2000, 6000)	(3000, 7000)
	max_depth	[2, 6, 10]	[2, 6, 10]	[2, 4, 8]
	min_child_weight	[2, 6, 10]	[2, 6, 10]	[2, 6, 10]
	gamma	[0, 0.001, 0.01, 0.1]	[0, 0.001, 0.01, 0.1]	[0, 0.001, 0.01, 0.1]
	lambda	[0, 0.01, 0.1, 1]	[0, 0.01, 0.1, 1]	[0, 0.01, 0.1]
	eta	[0.01]	[0.01]	[0.01]
LightGBM	nrounds	(500, 3500)	(500, 3500)	(3000, 7000)
	max_depth	[2, 4, 6, 8, 10]	[2, 4, 6, 8, 10]	[2, 4, 8]
	num_leaves	[10, 20, 30, 40]	[10, 20, 30, 40]	[10, 20, 30]
	bagging_fraction	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]
	feature_fraction	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]
	learning_rate	[0.01]	[0.01]	[0.1]
	lambda_l1	[1, 0.1, 0.01]	[1, 0.1, 0.01]	[1, 0.1, 0.01]
QRF	n_estimators	(500, 2000)	(1000, 6000)	[500, 1000, 2000]
	min_samples_split	[1, 5, 10, 20]	[5, 10, 15]	[5, 10, 20]
	max_depth	[5, 10, 20, 30]	[5, 10, 15]	[10, 20, 30]
	max_features	[3, 5, 10]	[3, 5, 10]	[5, 10, 20]
QERT	n_estimators	(500, 2000)	(1000, 4000)	[500, 1000, 2000]
	min_samples_split	[1, 5, 10, 20]	[5, 10, 20]	[5, 10, 20]
	max_depth	[5, 10, 20, 30]	[5, 10, 20]	[10, 20, 30]
	max_features	[3, 5, 10]	[3, 5, 10]	[5, 10, 20]

The terms inside square brackets indicate discrete values, and the terms in the round brackets indicate a sequence of numbers.

References

Alaa AM, Hussain Z, and Sontag D (2023). Conformalized unconditional quantile regression, *International Conference on Artificial Intelligence and Statistics*, **206**, 10690–10702.

Bloznelis D, Claeskens G, and Zhou J (2019). Composite versus model-averaged quantile regression, *Journal of Statistical Planning and Inference*, **200**, 32–46.

Buza K (2013). Feedback prediction for blogs. In *Proceedings of Spiliopoulou M, Schmidt-Thieme L, Janning R (eds) Data analysis, machine learning and knowledge discovery, Springer International Publishing, Cham*, 45–152.

Cannon AJ (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**, 3207–3225.

Chen T and Guestrin C (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, 785–794.

Das P and Ghosal S (2018). Bayesian non-parametric simultaneous quantile regression for complete and grid data, *Computational Statistics & Data Analysis*, **127**, 172–186.

Dietterich TG (2000). Ensemble methods in machine learning, In *International Workshop on Multiple Classifier Systems* (pp. 1–15), Berlin, Heidelberg.

Fanaee-TH and Gama J (2014). Event labeling combining ensemble detectors and background knowledge, *Progress in Artificial Intelligence*, **2**, 113–127.

Feldman S, Bates S, and Romano Y (2021). Improving conditional coverage via orthogonal quantile regression, *Advances in Neural Information Processing Systems*, **34**, 2060–2071.

Foresi S and Peracchi F (1995). The conditional distribution of excess returns: An empirical analysis, *Journal of the American Statistical Association*, **90**, 430, 451–466.

Friedman JH (2001). Greedy function approximation: A gradient boosting machine, *Annals of Statistics*, **29**, 1189–1232.

Geurts P, Ernst D, and Wehenkel L (2006). Extremely randomized trees, *Machine Learning*, **63**, 3–42.

Hatalis K, Lamadrid AJ, Scheinberg K, and Kishore S (2019). A novel smoothed loss and penalty function for noncrossing composite quantile estimation via deep neural networks, Available from: arXiv preprint arXiv:1909.12122

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

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

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, and Liu T-Y (2017). Lightgbm: A highly efficient gradient boosting decision tree, *Advances in Neural Information Processing Systems*, **30**, 3146–3154.

Keilbar G and Wang W (2022). Modelling systemic risk using neural network quantile regression, *Empirical Economics*, **62**, 93–118.

Klein N, Kneib T, KLang S, and Sohn A (2015). Bayesian structured additive distributional regression with an application to regional income inequality in Germany, *The Annals of Applied Statistics*, **9**, 1024–1052.

Koenker R and Bassett Jr G (1978). Regression quantiles, *Econometrica: Journal of the Econometric Society*, **46**, 33–50.

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

Koenker R, Portnoy S, Ng PT, Zeileis A, Grosjean P, and Ripley BD (2018). Package quantreg, Reference manual available at R-CRAN: `https://cran.rproject.org/web/packages/quantreg/quantreg.pdf`. Accessed 12 Jan 2023

Koenker R (2005). Quantile regression, Cambridge University Press, Cambridge.

Li D and Wang HJ (2019). Extreme quantile estimation for autoregressive models, *Journal of Business & Economic Statistics*, **37**, 661–670.

Li Y, Liu Y, and Zhu J (2007). Quantile regression in reproducing kernel Hilbert spaces, *Journal of the American Statistical Association*, **102**, 255–268.

Liu Y and Wu Y (2011). Simultaneous multiple non-crossing quantile regression estimation using kernel constraints, *Journal of Nonparametric Statistics*, **23**, 415–437.

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

Pedregosa F, Varoquaux G, Gramfort A *et al.* (2011). Scikit-learn: Machine learning in Python, *The Journal of Machine Learning Research*, **12**, 2825–2830.

Powell D (2020). Quantile treatment effects in the presence of covariates, *Review of Economics and Statistics*, **102**, 994–1005.

Roebroek J (2022). Sklearn-quantile, Available from: `https://github.com/jasperroebroek/sklearn-quantile`. Accessed 10 Aug 2023

Romano Y, Patterson E, and Candes E (2019). Conformalized quantile regression, *Advances in Neural Information Processing Systems*, **32**, Available from: https://proceedings.neurips.cc/paper_files/paper/2019/file/5103c3584b063c431bd1268e9b5e76fb-Paper.pdf.

Sangnier M, Fercoq O, and d'Alché-Buc F (2016). Joint quantile regression in vector-valued RKHSs, *Advances in Neural Information Processing Systems*, **29**, Available from: https://proceedings.neurips.cc/paper_files/paper/2016/file/dfce06801e1a85d6d06f1fdd4475dacd-Paper.pdf.

Schallhorn N, Kraus D, Nagler T, and Czado C (2017). D-vine quantile regression with discrete variables, Available from: arXiv preprint arXiv:1705.08310

Shin W and Jung Y (2023). Deep support vector quantile regression with non-crossing constraints, *Computational Statistics*, **38**, 1947–1976.

Takeuchi I, Le QV, Sears TD, and Smola AJ (2006). Nonparametric quantile estimation, *Journal of Machine Learning Research*, **7**, 1231–1264.

Xie Z and Wen H (2019). Composite quantile regression long short-term memory network. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks* (pp. 513–524), Munich, Germany.

Yeh IC (1998). Modeling of strength of high-performance concrete using artificial neural networks, *Cement and Concrete Research*, **28**, 1797–1808.

Zhao Z and Xiao Z (2014). Efficient regressions via optimally combining quantile information, *Econometric Theory*, **30**, 1272–1314.

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

Received July 03, 2024; Revised July 29, 2024; Accepted August 01, 2024