Comput Stat (2014) 29:1403–1426
DOI 10.1007/s00180-014-0498-x

ORIGINAL PAPER

A generalized Newton algorithm for quantile regression models

Songfeng Zheng

Received: 1 April 2013 / Accepted: 15 April 2014 / Published online: 30 April 2014

Abstract This paper formulates the quadratic penalty function for the dual problem of the linear programming associated with the L_1 constrained linear quantile regression model. We prove that the solution of the original linear programming can be obtained by minimizing the quadratic penalty function, with the formulas derived. The obtained quadratic penalty function has no constraint, thus could be minimized efficiently by a generalized Newton algorithm with Armijo step size. The resulting algorithm is easy to implement, without requiring any sophisticated optimization package other than a linear equation solver. The proposed approach can be generalized to the quantile regression model in reproducing kernel Hilbert space with slight modification. Extensive experiments on simulated data and real-world data show that, the proposed Newton quantile regression algorithms can achieve performance comparable to state-of-the-art.

Keywords Linear programming · Quadratic penalty function · Armijo step · L_1 constrained model

1 Introduction

The classical least square regression aims to estimate the conditional expectation of the response y given the predictor (vector) $\mathbf{x}$, $\mathrm{E}(y|\mathbf{x})$. However, it is well known that the mean value (or the conditional expectation) is sensitive to outliers of the data set. Therefore, if the data is not homogeneously distributed, we would expect the traditional least square regression to give us a poor prediction.

S. Zheng (✉)
Department of Mathematics, Missouri State University, Springfield, MO 65897, USA
e-mail: SongfengZheng@MissouriState.edu

A τ-th quantile is defined as the value such that there are 100τ % of the data smaller than it, or a value such that there is 100τ % of mass on the left side of it. Compared to the mean value, quantile values are more robust to outliers. Moreover, a series of quantile values can describe the whole data distribution better than a single value (e.g., mean value) does. For a random variable y, it was proved in Hunter and Lange (2000) that

$$Q_\tau = \arg\min_c \mathrm{E}\left[\rho_\tau(y - c)\right], \tag{1}$$

where Q_τ is the τ-th quantile of the random variable y; $\rho_\tau(r)$ is the so-called "check function" which is defined as

$$\rho_\tau(r) = \begin{cases} \tau r, & \text{if } r > 0 \\ -(1-\tau)r, & \text{otherwise} \end{cases} \tag{2}$$

and here $\tau \in (0, 1)$ indicates the quantile of interest.

Similar to least square regression, quantile regression (Koenker 2005; Koenker and Bassett 1978) aims at estimating the conditional quantiles of the response variable given a predictor variable (or vector). Suppose we are given a set of training data $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, with input $\mathbf{x}_i \in \mathbb{R}^p$ and output $y_i \in \mathbb{R}$, and the goal is to recover the τ-th quantile of the response y given the predictor vector $\mathbf{x}$. Assume we denote the conditional quantile as $f(\mathbf{x})$, which is called quantile regressor. In accordance with the definition of quantile in Eq. (1), quantile regressor can be estimated by

$$\hat{f}(\cdot) = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \rho_\tau\left(y_i - f(\mathbf{x}_i)\right), \tag{3}$$

where $\mathcal{F}$ is the space of all the permissible functions. We assume for the time being that the space $\mathcal{F}$ contains all the linear functions of the predictors, i.e., we estimate the τ-th quantile by $f(\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta} + \gamma$, where $\boldsymbol{\beta} \in \mathbb{R}^p$ is the coefficient vector and $\gamma \in \mathbb{R}$ is the intercept.

The linear quantile regression model in Eq. (3) can be solved by linear programming algorithms (Koenker 2005; Koenker and Park 1996) or Majorize-Minimize algorithm (Hunter and Lange 2000) which sequentially minimizes an upper bound of the objective function in Eq. (3). Li et al. (2011) proposed a model for piecewise linear quantile regression function. The idea of support vector regression was introduced for fitting quantile regression model, yielding Support Vector Quantile Regression (SV-QReg) (Hwang and Shim 2005; Li et al. 2007; Takeuchi et al. 2006), which was applied in Sohn et. al (2008a, b) for microarray analysis. SV-QReg can also estimate nonlinear quantile regression models or models in high dimensional spaces, but it is computationally expensive because it needs to solve a quadratic programming problem. Regression forest was used to estimate the conditional quantiles in Meinshausen (2006), but as a nonparametric model, the quantile regression forest is not easy to interpret. Langford et al. (2006) proposed to use classification technique in estimating the conditional quantile. For a given quantile value, their method trains a set of classifiers

$\{c_t\}$ for a series of $t \in [0, 1]$, and the testing stage calculates the average of the outputs of the classifiers. Therefore, this method is time consuming. For linear models, it is well known that an L_1 constraint on the coefficient vector helps select informative variables (Tibshirani 1996). Inspired by this idea, the L_1-constrained linear quantile regression (Li and Zhu 2008; Wu and Liu 2009) was proposed, and Li and Zhu (2008) gives an algorithm which can produce the whole solution path.

To fit the linear quantile regression model, this paper studies the associated linear programming, and formulates the quadratic penalty function of the dual of the linear programming. We further prove that the solution to quantile regression model can be obtained by minimizing the quadratic penalty function with no constraint, and the explicit formulas are also derived. In order to minimize the quadratic penalty function, a generalized Newton algorithm with Armijo step size is applied. The proposed Newton algorithm can be implemented with a few lines of MATLAB code, without requiring any sophisticated optimization software package other than a linear equation solver. In addition, with slight modification, the proposed approach can be easily generalized to fit nonlinear quantile regression model in reproducing kernel Hilbert space.

The proposed approach was tested on various simulated and real-world datasets, and the results show that the proposed approach performs comparable to state-of-the-art. Although the penalty function method has been successfully applied to train variants of Support Vector Machines (Fung and Mangasarian 2004; Mangasarian 2006),[1] to the best of our knowledge, this is the first attempt to apply penalty function method for fitting quantile regression model, and the resulting algorithms are easy to implement with performance comparable to state-of-the-art.

The rest of this paper is organized as following: Sect. 1.1 briefly defines the notations used in this paper; Sect. 2 studies the linear programming associated with linear quantile regression and formulates the quadratic penalty function of the dual, we will prove that a linear quantile regression model can be obtained by minimizing the quadratic penalty function with the explicit formulas derived; Sect. 3 proposes a generalized Newton algorithm to minimize the quadratic penalty function; the generalization of the proposed approach to kernel quantile regression model is presented in Sect. 4; Sect. 5 compares the proposed approach to various alternatives via simulated datasets, and Sect. 6 compares the performances of various algorithms on two real-world datasets; finally, Sect. 7 summarizes this paper.

1.1 Notations

All scalars are represented by lower case letters. All vectors will be denoted by **bold** lower case symbols, and all are column vectors unless transposed to a row vector by a prime superscript $'$. All matrices will be denoted by **bold** upper case symbols. For two vectors $\mathbf{a}$ and $\mathbf{b}$ in the n-dimensional real space $\mathbb{R}^n$, $\mathbf{a} \geq \mathbf{b}$ means $a_i \geq b_i$ for each $i = 1, \ldots, n$. For a vector $\mathbf{x}$ in $\mathbb{R}^n$, $\|\mathbf{x}\|$ stands for the 2-norm of the vector,

[1] The quadratic penalty function was called asymptotic exterior penalty function in Fung and Mangasarian (2004) and Mangasarian (2006). However, we found that "quadratic penalty function" is a more standard terminology, see Ruszczyński (2006, Sect. 6.2.2) and Bertsekas (1999, Sect. 4.2.1).

that is, $\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}$. For a vector $\mathbf{x}$ in $\mathbb{R}^n$, the plus function $\mathbf{x}_+$ is defined as $(\mathbf{x}_+)_i = \max\{0, x_i\}$, for $i = 1, \ldots, n$. The subgradient of $\mathbf{x}_+$ is denoted by $\mathbf{x}_*$, which is a step function defined as $(\mathbf{x}_*)_i = 1$ if $x_i > 0$, $(\mathbf{x}_*)_i = 0$ if $x_i < 0$, and $(\mathbf{x}_*)_i \in [0, 1]$ if $x_i = 0$, for $i = 1, \ldots, n$. Thus, $(\mathbf{x}_*)_i$ is any value in the interval $[0, 1]$ if $x_i = 0$, and we typically take $(\mathbf{x}_*)_i = 0.5$ in this case.

A column vector of ones (zeros) in k-dimensional space will be denoted by $\mathbf{1}_k$ ($\mathbf{0}_k$), and the identity matrix of k-th order will be denoted by $\mathbf{I}_k$. For matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times k}$, the kernel $K(\mathbf{A}, \mathbf{B})$ is a function which maps $\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times k}$ into $\mathbb{R}^{m\times k}$. In particular, if $\mathbf{x}$ and $\mathbf{y}$ are column vectors in $\mathbb{R}^n$, then $K(\mathbf{x}', \mathbf{y})$ is a real number, $K(\mathbf{x}', \mathbf{A}')$ is a row vector in $\mathbb{R}^m$ and $K(\mathbf{A}, \mathbf{A}')$ is an $m \times m$ matrix.

If f is a real-valued function defined on $\mathbb{R}^n$, the gradient of f at $\mathbf{x}$ is denoted by $\nabla f(\mathbf{x})$ which is a column vector in $\mathbb{R}^n$, and the Hessian of f at $\mathbf{x}$ is denoted by $\nabla^2 f(\mathbf{x})$, which is an $n \times n$ matrix. For a piecewise quadratic function $f(\mathbf{x}) = \frac{1}{2}\|(\mathbf{A}\mathbf{x} - \mathbf{b})_+\|^2$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$, the gradient vector is $\nabla f(\mathbf{x}) = \mathbf{A}'(\mathbf{A}\mathbf{x} - \mathbf{b})_+$, which is not differentiable, thus the ordinary Hessian of f does not exist. However, we can define its generalized Hessian which is the $n \times n$ symmetric positive semi-definite matrix

$$\partial^2 f(\mathbf{x}) = \mathbf{A}'\mathrm{diag}(\mathbf{A}\mathbf{x} - \mathbf{b})_*\mathbf{A},$$

where $\mathrm{diag}(\mathbf{A}\mathbf{x} - \mathbf{b})_*$ denotes an $m \times m$ diagonal matrix with diagonal elements $(\mathbf{A}_i\mathbf{x} - b_i)_*$, for $i = 1, \cdots, m$, where $\mathbf{A}_i$ is the i-th row of matrix $\mathbf{A}$. The generalized Hessian has many of the properties of the regular Hessian in relation to $f(\mathbf{x})$ (Hiriart-Urruty et al. 1984).

2 Quadratic penalty function for linear quantile regression

Suppose we are given a set of training data $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, with predictor $\mathbf{x}_i \in \mathbb{R}^p$ and response $y_i \in \mathbb{R}$. Assuming a linear quantile regression function

$$q_\tau(\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta} + \gamma,$$

the problem becomes estimating the regression coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^p$ and intercept $\gamma \in \mathbb{R}$ from the training data. Recently, L_1 quantile regression model (Li and Zhu 2008; Wu and Liu 2009) was proposed which imposes an L_1 constraint on the coefficient vector $\boldsymbol{\beta}$, i.e.,

$$\min_{\boldsymbol{\beta},\gamma} \quad \sum_{i=1}^{n} \rho_\tau\left(y_i - \mathbf{x}_i'\boldsymbol{\beta} - \gamma\right) + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{4}$$

with regularization parameter $\lambda \geq 0$. We note that if we set $\lambda = 0$ in Eq. (4), we will recover the ordinary linear quantile regression problem in Eq. (3). Thus, it suffices to study the problem in Eq. (4).

The minimization problem in Eq. (4) can be formulated as a linear programming

$$\begin{cases} \min_{\boldsymbol{\xi},\boldsymbol{\zeta},\gamma_1,\gamma_2,\mathbf{w}_1,\mathbf{w}_2} & \tau\mathbf{1}_n'\boldsymbol{\xi}+(1-\tau)\mathbf{1}_n'\boldsymbol{\zeta}+\lambda\mathbf{1}_p'\mathbf{w}_1+\lambda\mathbf{1}_p'\mathbf{w}_2 \\ \text{s.t.} & y_i-\mathbf{x}_i'(\mathbf{w}_1-\mathbf{w}_2)-(\gamma_1-\gamma_2)=\xi_i-\zeta_i \quad \text{for} \quad i=1,\cdots,n \\ & \quad \text{and} \quad \boldsymbol{\xi}\geq\mathbf{0}_n,\ \boldsymbol{\zeta}\geq\mathbf{0}_n,\ \gamma_1\geq 0,\ \gamma_2\geq 0,\ \mathbf{w}_1\geq\mathbf{0}_p,\ \mathbf{w}_2\geq\mathbf{0}_p \end{cases} \tag{5}$$

In the formulation of Eq. (5), we decompose the coefficient vector $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta}=\mathbf{w}_1-\mathbf{w}_2, \quad \text{with} \quad \mathbf{w}_1\geq\mathbf{0}_p,\ \mathbf{w}_2\geq\mathbf{0}_p,$$

and we also decompose the intercept γ as

$$\gamma=\gamma_1-\gamma_2, \quad \text{with} \quad \gamma_1\geq 0,\ \gamma_2\geq 0.$$

To simplify the notations, we rewrite the optimization problem in Eq. (5) in matrix form as

$$\begin{cases} \min_{\bar{\mathbf{x}}\geq 0} & \mathbf{c}'\bar{\mathbf{x}} \\ \text{s. t.} & \mathbf{A}\bar{\mathbf{x}}=\mathbf{y}, \end{cases} \tag{6}$$

with

$$\begin{cases} \mathbf{c}=\left[\tau\mathbf{1}_n', (1-\tau)\mathbf{1}_n', 0, 0, \lambda\mathbf{1}_p', \lambda\mathbf{1}_p'\right]', \\ \bar{\mathbf{x}}=\left[\boldsymbol{\xi}', \boldsymbol{\zeta}', \gamma_1, \gamma_2, \mathbf{w}_1', \mathbf{w}_2'\right]', \\ \mathbf{A}=[\mathbf{I}_n, -\mathbf{I}_n, \mathbf{1}_n, -\mathbf{1}_n, \mathbf{X}, -\mathbf{X}], \\ \mathbf{y}=[y_1,\ldots,y_n]', \end{cases}$$

where $\mathbf{X}$ is the data matrix of size $n\times p$, that is, the i-th row of $\mathbf{X}$ is the i-th training example $\mathbf{x}_i'$.

The dual of the linear programming in Eq. (6) is

$$\begin{cases} \max_{\mathbf{u}} & \mathbf{y}'\mathbf{u} \\ \text{s. t.} & \mathbf{u}'\mathbf{A}\leq\mathbf{c}' \end{cases} \tag{7}$$

where $\mathbf{u}$ is the vector of dual variables. We explicitly write the constraint in Eq. (7) as

$$\mathbf{u}'[\mathbf{I}_n, -\mathbf{I}_n, \mathbf{1}_n, -\mathbf{1}_n, \mathbf{X}, -\mathbf{X}]\leq\left[\tau\mathbf{1}_n', (1-\tau)\mathbf{1}_n', 0, 0, \lambda\mathbf{1}_p', \lambda\mathbf{1}_p'\right],$$

in detail, we have

$$-(1-\tau)\mathbf{1}_n\leq\mathbf{u}\leq\tau\mathbf{1}_n, \quad \mathbf{1}_n'\mathbf{u}\leq 0, \quad -\mathbf{1}_n'\mathbf{u}\leq 0, \quad -\lambda\mathbf{1}_p\leq\mathbf{X}'\mathbf{u}\leq\lambda\mathbf{1}_p.$$

For the linear programming in Eq. (7), the quadratic penalty function method[2] (Ruszczyński 2006, Sect. 6.2.2) solves the following unconstrained minimization problem

$$\min_{\mathbf{u}} \quad f(\mathbf{u})$$

where the quadratic penalty function is

$$\begin{aligned} f(\mathbf{u}) = & -\epsilon \mathbf{y}'\mathbf{u} + \frac{1}{2}\|(\mathbf{u} - \tau \mathbf{1}_n)_+\|^2 + \frac{1}{2}\|(-\mathbf{u} - (1-\tau)\mathbf{1}_n)_+\|^2 \\ & + \frac{1}{2}(\mathbf{1}_n'\mathbf{u})_+^2 + \frac{1}{2}(-\mathbf{1}_n'\mathbf{u})_+^2 \\ & + \frac{1}{2}\| \left(\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+ \|^2 + \frac{1}{2}\| \left(-\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+ \|^2, \end{aligned} \tag{8}$$

with $\epsilon > 0$.

Similar to Proposition 2 in Mangasarian (2006), we prove the following property of the quadratic penalty function:

Proposition 1 *There is an $\bar{\epsilon} > 0$, such that for any $\epsilon \in (0, \bar{\epsilon}]$, minimizing $f(\mathbf{u})$ in Eq. (8) provides a solution to the L_1 constrained linear quantile regression model in Eq. (6).*

Proof Minimizing $f(\mathbf{u})$ in Eq. (8) is equivalent to

$$\min_{\mathbf{u}, \mathbf{z}_1, \mathbf{z}_2, z_3, z_4, \mathbf{z}_5, \mathbf{z}_6} \quad -\epsilon \mathbf{y}'\mathbf{u} + \frac{1}{2}\left(\|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 + z_3^2 + z_4^2 + \|\mathbf{z}_5\|^2 + \|\mathbf{z}_6\|^2\right) \tag{9}$$

$$\text{s. t.} \quad \begin{cases} -\mathbf{u} + \tau \mathbf{1}_n + \mathbf{z}_1 \geq 0 \\ \mathbf{u} + (1-\tau)\mathbf{1}_n + \mathbf{z}_2 \geq 0 \\ -\mathbf{1}_n'\mathbf{u} + z_3 \geq 0 \\ \mathbf{1}_n'\mathbf{u} + z_4 \geq 0 \\ -\mathbf{X}'\mathbf{u} + \lambda \mathbf{1}_p + \mathbf{z}_5 \geq 0 \\ \mathbf{X}'\mathbf{u} + \lambda \mathbf{1}_p + \mathbf{z}_6 \geq 0 \end{cases} \tag{10}$$

The justification for this is that at the minimum of Eq. (9), the variables $\mathbf{z}_i$'s are nonnegative; otherwise, if any component of these variables is negative, the objective function can be strictly decreased by setting that component to zero while maintaining the constraints in Eq. (10) feasibility. Hence, at the minima, we clearly have

$$\mathbf{z}_1 = (\mathbf{u} - \tau \mathbf{1}_n)_+, \quad \mathbf{z}_2 = (-\mathbf{u} - (1-\tau)\mathbf{1}_n)_+, \quad z_3 = (\mathbf{1}_n'\mathbf{u})_+, \tag{11}$$

and

$$z_4 = (-\mathbf{1}_n'\mathbf{u})_+, \quad \mathbf{z}_5 = \left(\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+, \quad \mathbf{z}_6 = (-\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p)_+. \tag{12}$$

[2] The penalty function method in optimization explicitly absorbs the constraints, such that an unconstrained optimization problem is obtained.

Applying Lagrange multiplier method with nonnegative multipliers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6$ for the six constraints in Eq. (10), we have the Lagrange function

$$\begin{aligned} L(\mathbf{u}, \mathbf{z}_1, \ldots, \mathbf{z}_6, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6) = & -\epsilon \mathbf{y}'\mathbf{u} + \frac{1}{2}\left(\|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 + z_3^2 + z_4^2 + \|\mathbf{z}_5\|^2 + \|\mathbf{z}_6\|^2\right) \\ & - \boldsymbol{\mu}_1'\left(-\mathbf{u} + \tau \mathbf{1}_n + \mathbf{z}_1\right) - \boldsymbol{\mu}_2'\left(\mathbf{u} + (1-\tau)\mathbf{1}_n + \mathbf{z}_2\right) \\ & - \mu_3\left(-\mathbf{1}_n'\mathbf{u} + z_3\right) - \mu_4\left(\mathbf{1}_n'\mathbf{u} + z_4\right) \\ & - \boldsymbol{\mu}_5'\left(-\mathbf{X}'\mathbf{u} + \lambda \mathbf{1}_p + \mathbf{z}_5\right) - \boldsymbol{\mu}_6'\left(\mathbf{X}'\mathbf{u} + \lambda \mathbf{1}_p + \mathbf{z}_6\right). \end{aligned}$$

At the minimum of L, we must have

$$\frac{\partial L}{\partial \mathbf{u}} = 0 \Rightarrow -\epsilon \mathbf{y} + \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 + \mu_3 \mathbf{1}_n - \mu_4 \mathbf{1}_n + \mathbf{X}\boldsymbol{\mu}_5 - \mathbf{X}\boldsymbol{\mu}_6 = 0, \tag{13}$$

and

$$\frac{\partial L}{\partial \mathbf{z}_i} = 0 \Rightarrow \boldsymbol{\mu}_i = \mathbf{z}_i \geq 0, \qquad \text{for} \qquad i = 1, \cdots, 6.$$

Substituting the above relations to L, we get the Wolfe dual function as

$$\begin{aligned} W = & -(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 + \mu_3 \mathbf{1}_n - \mu_4 \mathbf{1}_n + \mathbf{X}\boldsymbol{\mu}_5 - \mathbf{X}\boldsymbol{\mu}_6)'\mathbf{u} \\ & + \frac{1}{2}\left(\|\boldsymbol{\mu}_1\|^2 + \|\boldsymbol{\mu}_2\|^2 + \mu_3^2 + \mu_4^2 + \|\boldsymbol{\mu}_5\|^2 + \|\boldsymbol{\mu}_6\|^2\right) \\ & - \boldsymbol{\mu}_1'(-\mathbf{u} + \tau \mathbf{1}_n + \boldsymbol{\mu}_1) - \boldsymbol{\mu}_2'(\mathbf{u} + (1-\tau)\mathbf{1}_n + \boldsymbol{\mu}_2) \\ & - \mu_3\left(-\mathbf{1}_n'\mathbf{u} + \mu_3\right) - \mu_4\left(\mathbf{1}_n'\mathbf{u} + \mu_4\right) \\ & - \boldsymbol{\mu}_5'\left(-\mathbf{X}'\mathbf{u} + \lambda \mathbf{1}_p + \boldsymbol{\mu}_5\right) - \boldsymbol{\mu}_6'\left(\mathbf{X}'\mathbf{u} + \lambda \mathbf{1}_p + \boldsymbol{\mu}_6\right) \\ = & -\frac{1}{2}\left(\|\boldsymbol{\mu}_1\|^2 + \|\boldsymbol{\mu}_2\|^2 + \mu_3^2 + \mu_4^2 + \|\boldsymbol{\mu}_5\|^2 + \|\boldsymbol{\mu}_6\|^2\right) \\ & - \left[\tau \mathbf{1}_n'\boldsymbol{\mu}_1 + (1-\tau)\mathbf{1}_n'\boldsymbol{\mu}_2 + 0 \cdot \mu_3 + 0 \cdot \mu_4 + \lambda \mathbf{1}_p'\boldsymbol{\mu}_5 + \lambda \mathbf{1}_p'\boldsymbol{\mu}_6\right]. \end{aligned}$$

The Wolfe dual problem is to maximize W with respect to the nonnegative Lagrange multipliers $\boldsymbol{\mu}_i$'s with the constraint in Eq. (13), which is clearly equivalent to

$$\begin{aligned} \min_{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6} \quad & \frac{1}{2}\left(\|\boldsymbol{\mu}_1\|^2 + \|\boldsymbol{\mu}_2\|^2 + \mu_3^2 + \mu_4^2 + \|\boldsymbol{\mu}_5\|^2 + \|\boldsymbol{\mu}_6\|^2\right) \\ & + \left[\tau \mathbf{1}_n'\boldsymbol{\mu}_1 + (1-\tau)\mathbf{1}_n'\boldsymbol{\mu}_2 + 0 \cdot \mu_3 + 0 \cdot \mu_4 + \lambda \mathbf{1}_p'\boldsymbol{\mu}_5 + \lambda \mathbf{1}_p'\boldsymbol{\mu}_6\right] \qquad (14) \\ \text{s. t.} \quad & \begin{cases} -\epsilon \mathbf{y} + \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 + \mu_3 \mathbf{1}_n - \mu_4 \mathbf{1}_n + \mathbf{X}\boldsymbol{\mu}_5 - \mathbf{X}\boldsymbol{\mu}_6 = 0 \\ \boldsymbol{\mu}_i \geq 0 \qquad \text{for } i = 1, \cdots, 6 \end{cases} \qquad (15) \end{aligned}$$

Let

$$\mathbf{v} = \left[\boldsymbol{\mu}_1', \boldsymbol{\mu}_2', \mu_3, \mu_4, \boldsymbol{\mu}_5', \boldsymbol{\mu}_6'\right]',$$

and recall that

$$\mathbf{c} = \left[\tau \mathbf{1}_n', (1-\tau)\mathbf{1}_n', 0, 0, \lambda \mathbf{1}_p', \lambda \mathbf{1}_p'\right]' \quad \text{and} \quad \mathbf{A} = [\mathbf{I}_n, -\mathbf{I}_n, \mathbf{1}_n, -\mathbf{1}_n, \mathbf{X}, -\mathbf{X}]\,.$$

With these notations, the minimization problem in Eqs. (14) and (15) can be rewritten as

$$\begin{cases} \min_{\mathbf{v} \geq \mathbf{0}} & \frac{1}{2}\|\mathbf{v}\|^2 + \mathbf{c}'\mathbf{v} \\ \text{s. t.} & \mathbf{A}\mathbf{v} = \epsilon \mathbf{y} \end{cases} \tag{16}$$

Define $\bar{\mathbf{w}} = \mathbf{v}/\epsilon$, then Eq. (16) becomes

$$\begin{cases} \min_{\bar{\mathbf{w}} \geq 0} & \mathbf{c}'\bar{\mathbf{w}} + \frac{1}{2}\epsilon\|\bar{\mathbf{w}}\|^2 \\ \text{s. t.} & \mathbf{A}\bar{\mathbf{w}} = \mathbf{y} \end{cases} \tag{17}$$

We notice that the minimization problem in Eq. (17) is the perturbed problem of Eq. (6). By the perturbation theory of linear programming (Mangasarian and Meyer 1979), there exists $\bar{\epsilon} > 0$, such that for $\epsilon \in (0, \bar{\epsilon}]$, the solution of the perturbed problem in Eq. (17) is the solution of the original problem in Eq. (6).

From the above derivation, clearly, minimizing Eq. (8) results in $\mathbf{u}$, which could be used for calculating $\mathbf{z}_i$'s by Eqs. (11) and (12). Further, $\boldsymbol{\mu}_i = \mathbf{z}_i$, which gives $\bar{\mathbf{w}}$, a solution to the original L_1 constrained linear quantile regression problem in Eq. (6). Thus, the claim in the Proposition is established. □

Based on the conclusion of Proposition 1, and using the corresponding relationship between components of $\bar{\mathbf{w}}$ (or $\mathbf{v}$) in Eq. (17) and those of $\bar{\mathbf{x}}$ in Eq. (6), it is not difficult to derive that, as ϵ small enough,

$$\boldsymbol{\xi} = \boldsymbol{\mu}_1/\epsilon = (\mathbf{u} - \tau \mathbf{1}_n)_+/\epsilon, \quad \boldsymbol{\zeta} = \boldsymbol{\mu}_2/\epsilon = (-\mathbf{u} - (1-\tau)\mathbf{1}_n)_+/\epsilon, \tag{18}$$

$$\gamma_1 = \mu_3/\epsilon = (\mathbf{1}_n'\mathbf{u})_+/\epsilon, \quad \gamma_2 = \mu_4/\epsilon = (-\mathbf{1}_n'\mathbf{u})_+/\epsilon, \tag{19}$$

and

$$\mathbf{w}_1 = \boldsymbol{\mu}_5/\epsilon = \left(\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+/\epsilon, \quad \mathbf{w}_2 = \boldsymbol{\mu}_6/\epsilon = \left(-\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+/\epsilon. \tag{20}$$

We summarize the above results as

Proposition 2 *There is an $\bar{\epsilon} > 0$, and let $\mathbf{u}$ be the solution to the quadratic penalty function minimization problem in Eq. (8) for any $\epsilon \in (0, \bar{\epsilon}]$. Then $\mathbf{u}$ provides a solution to the L_1 constrained linear quantile regression model with*

$$\boldsymbol{\beta} = \mathbf{w}_1 - \mathbf{w}_2 = \left(\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+/\epsilon - \left(-\mathbf{X}'\mathbf{u} - \lambda \mathbf{1}_p\right)_+/\epsilon, \tag{21}$$

and

$$\gamma = \gamma_1 - \gamma_2 = (\mathbf{1}_n'\mathbf{u})_+/\epsilon - (-\mathbf{1}_n'\mathbf{u})_+/\epsilon = \mathbf{1}_n'\mathbf{u}/\epsilon. \tag{22}$$

In Eq. (22), we used the relation[3] $(a)_+ - (-a)_+ = a$ for any $a \in \mathbb{R}$.

3 The generalized Newton algorithm

As derived in Eqs. (21) and (22), the problem is summarized as solving for $\mathbf{u}$ by minimizing the function $f(\mathbf{u})$ defined in Eq. (8). For its simplicity, we choose to use Newton's method to minimize $f(\mathbf{u})$. In each iteration, Newton's algorithm searches for the optimal point in the direction of $-(\nabla^2 f(\mathbf{u}))^{-1}\nabla f(\mathbf{u})$.

It is easy to find the gradient vector of $f(\mathbf{u})$ as

$$\begin{aligned}\nabla f(\mathbf{u}) &= -\epsilon\mathbf{y} + (\mathbf{u} - \tau\mathbf{1}_n)_+ - (-\mathbf{u} - (1-\tau)\mathbf{1}_n)_+ \\ &\quad + \mathbf{1}_n(\mathbf{1}_n'\mathbf{u})_+ - \mathbf{1}_n(-\mathbf{1}_n'\mathbf{u})_+ + \mathbf{X}(\mathbf{X}'\mathbf{u} - \lambda\mathbf{1}_p)_+ - \mathbf{X}\left(-\mathbf{X}'\mathbf{u} - \lambda\mathbf{1}_p\right)_+ \\ &= -\epsilon\mathbf{y} + (\mathbf{u} - \tau\mathbf{1}_n)_+ - (-\mathbf{u} - (1-\tau)\mathbf{1}_n)_+ + \mathbf{1}_n(\mathbf{1}_n'\mathbf{u}) \\ &\quad + \mathbf{X}\left[\left(\mathbf{X}'\mathbf{u} - \lambda\mathbf{1}_p\right)_+ - \left(-\mathbf{X}'\mathbf{u} - \lambda\mathbf{1}_p\right)_+\right].\end{aligned} \tag{23}$$

The regular Hessian of $f(\mathbf{u})$ does not exist because the plus function in Eq. (23) is not differentiable, but the generalized Hessian of $f(\mathbf{u})$ is calculated as

$$\begin{aligned}\partial^2 f(\mathbf{u}) &= \text{diag}\,(\mathbf{u} - \tau\mathbf{1}_n)_* + \text{diag}\,(-\mathbf{u} - (1-\tau)\mathbf{1}_n)_* + \mathbf{1}_n\mathbf{1}_n' \\ &\quad + \mathbf{X}\text{diag}(\mathbf{X}'\mathbf{u} - \lambda\mathbf{1}_p)_*\mathbf{X}' + \mathbf{X}\text{diag}\left(-\mathbf{X}'\mathbf{u} - \lambda\mathbf{1}_p\right)_*\mathbf{X}' \\ &= \text{diag}(\mathbf{u} - \tau\mathbf{1}_n)_* + \text{diag}\,(-\mathbf{u} - (1-\tau)\mathbf{1}_n)_* + \mathbf{1}_n\mathbf{1}_n' \\ &\quad + \mathbf{X}\left[\text{diag}\left(|\mathbf{X}'\mathbf{u}| - \lambda\mathbf{1}_p\right)_*\right]\mathbf{X}',\end{aligned} \tag{24}$$

where Eq. (24) follows from Eq. (25) (Please refer to "Appendix" for the proof)

$$(a - \lambda)_* + (-a - \lambda)_* = (|a| - \lambda)_*, \quad \text{with } \lambda \geq 0 \text{ and any } a \in \mathbb{R}. \tag{25}$$

Furthermore, we can prove

Proposition 3 *The generalized Hessian defined in Eq.* (24) *is semi-positive definite.*

Proof By definition, for any $r \in \mathbb{R}$, $(r)_* \geq 0$. Therefore, the matrices $\text{diag}(\mathbf{u} - \tau\mathbf{1}_n)_*$, $\text{diag}(-\mathbf{u} - (1-\tau)\mathbf{1}_n)_*$, and $\text{diag}(|\mathbf{X}'\mathbf{u}| - \lambda\mathbf{1}_p)_*$ all have non-negative elements on the diagonal, thus all of them are semi-positive definite.

For any vector $\mathbf{a} \in \mathbb{R}^n$,

$$\mathbf{a}'\mathbf{1}_n\mathbf{1}_n'\mathbf{a} = (\mathbf{a}'\mathbf{1}_n)^2 \geq 0$$

[3] This relation is easy to check: if $a > 0$, $(a)_+ = a$ while $(-a)_+ = 0$; if $a < 0$, $(a)_+ = 0$ and $(-a)_+ = -a$. Therefore, for any $a \in \mathbb{R}$, we always have $(a)_+ - (-a)_+ = a$.

and

$$\mathbf{a}'\mathbf{X}\left[\text{diag}(|\mathbf{X}'\mathbf{u}| - \lambda\mathbf{1}_p)_*\right]\mathbf{X}'\mathbf{a} = (\mathbf{a}'\mathbf{X})\left[\text{diag}(|\mathbf{X}'\mathbf{u}| - \lambda\mathbf{1}_p)_*\right](\mathbf{a}'\mathbf{X})' \geq 0$$

since the matrix $\text{diag}(|\mathbf{X}'\mathbf{u}| - \lambda\mathbf{1}_p)_*$ is semi-positive definite.

This completes the proof. □

Since the generalized Hessian has similar properties as the regular Hessian (Hiriart-Urruty et al. 1984), in the proposed algorithm, we update the solution in the direction of $-(\partial^2 f(\mathbf{u}) + \delta\mathbf{I}_n)^{-1}\nabla f(\mathbf{u})$, and we call the resulting algorithm as generalized Newton algorithm. Here δ is a small positive number and $\mathbf{I}_n$ is $n \times n$ identity matrix. The term $\delta\mathbf{I}_n$ is added to the generalized Hessian to make the resulting matrix invertible.

The generalized Hessian $\partial^2 f(\mathbf{u})$ is an $n \times n$ matrix, where n is the number of training examples. Let p stand for the dimensionality of the data. If $n \gg p$, inverting $\partial^2 f(\mathbf{u}) + \delta\mathbf{I}_n$ is prune to error due to the finite precision of floating-point numbers, and it is time consuming since the inverting operation has time complexity of $O(n^3)$. In this case, we can employ the Sherman–Morrison–Woodbury identity[4] as follows, define

$$\begin{aligned}\mathbf{E}^2 &= \text{diag}\left(|\mathbf{X}'\mathbf{u}| - \lambda\mathbf{1}_p\right)_*,\\ \mathbf{H} &= [\mathbf{X}\mathbf{E},\ \ \mathbf{1}_n],\end{aligned}$$

and

$$\mathbf{F} = \text{diag}\,(\mathbf{u} - \tau\mathbf{1}_n)_* + \text{diag}\,(-\mathbf{u} - (1-\tau)\mathbf{1}_n)_* + \delta\mathbf{I}_n.$$

It follows that

$$\partial^2 f(\mathbf{u}) + \delta\mathbf{I}_n = \mathbf{H}\mathbf{H}' + \mathbf{F}.$$

Applying the Sherman–Morrison–Woodbury identity, yields

$$\begin{aligned}(\partial^2 f(\mathbf{u}) + \delta\mathbf{I}_n)^{-1} &= (\mathbf{H}\mathbf{H}' + \mathbf{F})^{-1}\\ &= \mathbf{F}^{-1} - \mathbf{F}^{-1}\mathbf{H}(\mathbf{I}_{p+1} + \mathbf{H}'\mathbf{F}^{-1}\mathbf{H})^{-1}\mathbf{H}'\mathbf{F}^{-1}. \qquad (26)\end{aligned}$$

Note that inverting the $n \times n$ matrix $\mathbf{F}$ is trivial since $\mathbf{F}$ is diagonal; and the matrix $\mathbf{I}_{p+1} + \mathbf{H}'\mathbf{F}^{-1}\mathbf{H}$ is a $(p+1) \times (p+1)$ matrix, whose inverse is easier to calculate since $n \gg p$.

[4] The matrix identity is $(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}\mathbf{A}^{-1}$, where $\mathbf{A}$, $\mathbf{U}$, $\mathbf{C}$, and $\mathbf{V}$ denote matrices of appropriate sizes. If $\mathbf{A}^{-1}$ is easy to calculate and $\mathbf{C}$ has a much smaller dimension than $\mathbf{A}$, using this formula is more efficient than inverting $\mathbf{A} + \mathbf{UCV}$ directly. See Higham (2002) for more details.

We summarize the generalized Newton algorithm for fitting linear quantile regression model as following:[5]

Algorithm 1: Generalized Newton Algorithm for Linear Quantile Regression

0. Given the training data $\{(\mathbf{x}_i, y_i),\ i = 1, \cdots, n\}$ with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, the desired quantile value τ, and the regularization parameter λ; reformulate the data matrix $\mathbf{X}$ and the response vector $\mathbf{y}$.
1. Set the parameters ϵ, δ, error tolerance tol, and the maximum number of iteration M.
2. Define function $f(\mathbf{u})$ as in Eq. (8), and its gradient vector and generalized Hessian matrix as in Eq. (23) and Eq. (24), respectively.
3. Initialize any $\mathbf{u}^0 \in \mathbb{R}^n$
4. **for** $i = 1$ to M **do**:
5. Calculate the Newton direction $\mathbf{d}^i = -\left(\partial^2 f(\mathbf{u}^{i-1}) + \delta \mathbf{I}_n\right)^{-1} \bigtriangledown f(\mathbf{u}^{i-1})$.
6. Update the vector $\mathbf{u}$ by $\mathbf{u}^i = \mathbf{u}^{i-1} + \eta_i \mathbf{d}^i$, where η_i is the step size at the i-th iteration.
7. Stop if $\|\mathbf{u}^i - \mathbf{u}^{i-1}\| \leq tol$.
8. **end for**
9. Define the estimated quantile regression function by Eqs. (21) and (22).

In the 5th step of Algorithm 1, when $n \gg p$, the Sherman–Morrison–Woodbury identity in Eq. (26) is invoked for more efficient computation. In the 6th step, the step size η_i could be chosen as

$$\eta_i = \arg\min_{\eta > 0}\ f\left(\mathbf{u}^{i-1} + \eta \mathbf{d}^i\right). \tag{27}$$

The minimization problem in Eq. (27) can be solved by backtracking line search algorithms (Boyd and Vandenberghe 2004, Chap. 9). We choose to use Armijo rule (Armijo 1966) for its simplicity, which is given in Algorithm 2 for completeness.

Algorithm 2: Armijo Rule to Determine a Step Size

0. Given the objective function $f(\mathbf{u})$, its gradient $\nabla f(\mathbf{u})$, the current estimation $\mathbf{u}_c$, and the search direction $\mathbf{d}$.
1. Initialize $\eta = 1$.
2. Calculate $D = f(\mathbf{u}_c + \eta \mathbf{d}) - f(\mathbf{u}_c)$.
3. If $D \leq \frac{\eta}{4} \nabla f(\mathbf{u}_c)' \mathbf{d}$, return the current η as the step size; otherwise, set $\eta = \eta/2$, and go back to step 2.

About the convergence of Algorithm 1, we have

Remark 1 Let $tol = 0$, $M = \infty$, and $\epsilon > 0$ sufficiently small. By the results of penalty function method (Bertsekas 1999; Ruszczyński 2006), each accumulation point $\bar{\mathbf{u}}$ of the sequence $\{\mathbf{u}^i\}$ generated by the generalized Newton algorithm in Algorithm 1 is a solution to the quadratic penalty problem.

[5] Algorithm 1 is similar in spirit to an algorithm for Support Vector Machine studied in Fung and Mangasarian (2004), which is for pattern recognition problem, while our proposed algorithm is for regression problem.

The following remark is with regard to the parameter setting in Algorithm 1:

Remark 2 Theoretically, it is not an easy task to determine the size of ϵ such that the solution $\mathbf{u}$ of minimizing Eq. (8) yields a satisfactory linear quantile regression model. However, computationally, this does not seem to be critical. As we observed in our experiments, the testing errors are very close if ϵ is less than 0.01. The parameter tol in Algorithm 1 was usually set as 10^{-3} in our experiments. We can set the iteration number M as a large number, but we observed empirically that the algorithm usually converges in 50 iterations.

4 Generalize to kernel quantile regression

Suppose that the quantile regression function lies in a reproducing kernel Hilbert space, then according to the representation theorem (Kimeldorf and Wahba 1971), it could be written as

$$q_\tau(\mathbf{x}) = \gamma + \sum_{i=1}^{n} \beta_i K(\mathbf{x}', \mathbf{x}_i), \tag{28}$$

where $K(\cdot, \cdot)$ is the kernel function. In Hwang and Shim (2005), a method based on support vector regression was introduced to fit the kernel quantile regression model. This method is easy to implement based on the readily available SVM toolboxes[6] (Chang and Lin 2011; Gunn 1997; Joachims 1999). However, Hwang and Shim (2005) actually did not impose any constraint on the regression coefficients. While Li et al. (2007) proposed a kernel quantile regression model with the ridge constraint on the regression coefficients in the reproducing kernel Hilbert space.

The basic idea of kernel machine (e.g., support vector machine, support vector regression) is to achieve sparsity in the sample space, that is, we want a large part of the coefficients β_i's in Eq. (28) to be 0. As such, we impose an L_1 constraint on the coefficients β_i's in Eq. (28). Thus, a sparse quantile regression model in reproducing kernel Hilbert space can be obtained by solving the following optimization problem

$$\min_{\boldsymbol{\beta}, \gamma} \quad \sum_{i=1}^{n} \rho_\tau \left(y_i - \sum_{j=1}^{n} \beta_j K(\mathbf{x}_i', \mathbf{x}_j) - \gamma \right) + \lambda \sum_{i=1}^{n} |\beta_i|, \tag{29}$$

where $\lambda \geq 0$ is a regularization parameter.

Comparing Eqs. (4) and (29), it is clear that the optimization problem for kernel quantile regression in Eq. (29) can be obtained by replacing the vector $\mathbf{x}_i$ in Eq. (4) by the $n \times 1$ vector

$$\left(K(\mathbf{x}_i', \mathbf{x}_1), K(\mathbf{x}_i', \mathbf{x}_2), \ldots, K(\mathbf{x}_i', \mathbf{x}_n) \right)'.$$

[6] In our implementation, which is based on the toolbox from Gunn (1997), we assign a very large value (e.g., 5,000) to the penalty parameter C in support vector regression.

This clearly indicates that we can fit the kernel quantile regression model by the method developed in Sects. 2 and 3 with only slight modification. Explicitly, we just make the following replacement in the definition of function $f(\mathbf{u})$ in Eq. (8):

$$\mathbf{X} \to K(\mathbf{X}, \mathbf{X}').$$

We also notice that if we set $\lambda = 0$, we will get a kernel quantile regression model without any constraint, which is similar to the model given by Hwang and Shim (2005).

Computationally, the methods in Hwang and Shim (2005) and Li et al. (2007) need to solve a quadratic programming problem, while the proposed method only needs to solve a linear equation in each iteration (refer to the detailed algorithm in Sect. 3). Thus, the proposed kernel quantile regression is expected to be more computationally efficient.

5 Simulation studies

This section compares the proposed algorithms to alternatives on various simulated datasets. The tested algorithms include the quantile regression algorithm implemented based on an interior point method (IP-QReg) (Koenker 2005; Koenker and Park 1996), Majorize-Minimize based quantile regression (MM-QReg) (Hunter and Lange 2000), the support vector quantile regression (SV-QReg) (Hwang and Shim 2005) with linear kernel, the quantile regression forest (QReg Forest) (Meinshausen 2006), and the proposed Newton quantile regression (N-QReg). We also compare the performance of the proposed Newton kernel QReg to the kernel QReg model in Li et al. (2007). In N-QReg, the parameter ϵ is fixed at 0.01.

As a comparison to state-of-the-art, we also test the performance of the quantile regression function `rq` in the R package `"quantreg"`; for the L_1 constrained model, we use function `rq` with option `method="lasso"`, please refer to the document at http://stuff.mit.edu/afs/athena/software/r_v2.14.1/lib/R/library/quantreg/html/rq.fit.lasso.html for more details and examples. The L_1 constrained quantile regression model was initially proposed in Li and Zhu (2008), and their method produces the whole solution path, that is, the solutions for all possible λ. Since our method fits the model for a particular λ, it is not fair to compare to Li and Zhu (2008). As such, we choose to compare to the function in the R package.

All the experiments presented in Sects. 5 and 6 were performed on a personal computer with Pentium IV CPU 3.00 GHz and 1.00 GB memory, with WinXP operating system. The IP-QReg and MM-QReg were implemented based on the MATLAB code downloaded from http://www.stat.psu.edu/~dhunter/code/qrmatlab/. The SV-QReg was implemented based on the MATLAB SVM toolbox of Gunn (1997) with the quadratic programming solver from the C++ version of LIBSVM (Chang and Lin 2011). QReg Forest was implemented based on the R package `"quantregForest"`. The proposed N-QReg and L_1 N-QReg were implemented using MATLAB, without particular code optimization. In our experiments, we used MATLAB® R2007b and R 2.10.1.

Let the true τ-th quantile function be $q_\tau(\mathbf{x})$ and the estimated τ-th quantile function be $\hat{q}_\tau(\mathbf{x})$. The performance of the fitted model on the testing set $\{(\mathbf{x}_i, y_i), i = 1, \ldots, N_{\text{test}}\}$ is evaluated by the mean absolute deviation which is defined as

$$\text{Mean Absolute Deviation} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} |q_\tau(\mathbf{x}_i) - \hat{q}_\tau(\mathbf{x}_i)|. \tag{30}$$

To select the parameter λ in all the L_1 constrained models (i.e., the proposed method and the R function), we pretend that the true quantile function is not available in the simulations, and select the model parameter which minimizes the average of the "check loss" on the validation set $\{(\mathbf{x}_i, y_i), i = 1, \cdots, N_{\text{val}}\}$, which is defined as

$$L(\tau, \lambda) = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \rho_\tau \left(y_i - \hat{q}_{\tau,\lambda}(\mathbf{x}_i)\right), \tag{31}$$

and select

$$\hat{\lambda} = \arg\min_{\lambda > 0} L(\tau, \lambda). \tag{32}$$

We first compare the prediction accuracy of the proposed algorithms to those of alternative methods by Simulation 1; the second simulation studies the variable selection ability of the proposed method in the case of high dimension but small sample size; in Simulation 3, we study the performance of various algorithms in the case of heteroscedastic error terms; finally, Simulation 4 compares the performance of different kernel quantile regression models. For every algorithm, in each iteration, we used the same training set, the same validation set (if necessary), and the same testing set, in order to make the comparison fair.

Simulation 1 (*Comparing the prediction accuracy of various methods*) We generate data according to the model

$$y = \mathbf{x}'\mathbf{b} + \sigma\epsilon, \tag{33}$$

with $\mathbf{b} = (3, 1.5, 0, 0, 2, 0, 0, 0)'$ and $\mathbf{x} \sim N(\mathbf{0}, \mathbf{\Sigma}_{8\times 8})$. The pairwise correlation between x_i and x_j is given by $r^{|i-j|}$ with $r = 0.5$. The error term ϵ follows standard normal distribution. We choose $\sigma = 2$, resulting the signal-to-noise (S/N) ratio about 5. The S/N ratio is defined as $\text{var}(\mathbf{b}'\mathbf{x})/\text{var}(\sigma\epsilon)$. This model was considered in many works, for example, in Li and Zhu (2008) and Wu and Liu (2009), among others.

We generate 100 training examples and 10,000 testing examples from the model in Eq. (33). The performance of the fitted model on the testing set is evaluated by the mean absolute deviation defined in Eq. (30). Under the assumption of Eq. (33), the true τ-th quantile function can be written out explicitly as

$$q_\tau(\mathbf{x}) = \mathbf{x}'\mathbf{b} + \sigma\, Q_\tau^{\text{Norm}},$$

Table 1 The performances of QReg Forest, IP-QReg, SV-QReg, MM-QReg, the QReg function in the R package, and the proposed N-QReg models

Method	QReg Forest	IP-QReg	SV-QReg	MM-QReg	QReg R	N-QReg
$\tau = .25$	1.801 (.169)	.642 (.166)	.642 (.166)	.640 (.163)	.642 (.166)	**.628 (.161)**
$\tau = .50$	1.604 (.149)	.612 (.128)	.612 (.128)	.609 (.127)	.612 (.128)	**.596 (.123)**
$\tau = .75$	1.747 (.172)	.632 (.178)	.632 (.178)	.632 (.177)	.632 (.178)	**.623 (.174)**

Listed are the mean values of the Mean Absolute Deviation of the 100 runs, and the standard deviations are listed in parentheses. The best performance is marked in bold

Table 2 The performances of L_1 Newton QReg and the L_1 QReg function in R package on the data in Simulation 1

τ	Method	Test error	# of trimmed variables	
			Correct	Wrong
.25	L_1 N-QReg	.556 (.150)	4.79 (0.43)	0.00 (0.00)
	L_1 QReg (R)	.506 (.143)	4.74 (0.50)	0.00 (0.00)
.50	L_1 N-QReg	.500 (.136)	4.83 (0.40)	0.00 (0.00)
	L_1 QReg (R)	.487 (.128)	4.78 (0.44)	0.00 (0.00)
.75	L_1 N-QReg	.552 (.153)	4.79 (0.43)	0.00 (0.00)
	L_1 QReg (R)	.489 (.149)	4.73 (0.47)	0.00 (0.00)

For three τ values, the mean values of the Mean Absolute Deviation of the 100 runs are shown with the standard deviations listed in parentheses. The means and standard deviations of the number of correctly and mistakenly trimmed variables are listed as measures of variable selection performance

where Q_τ^{Norm} is the τ-th quantile of the standard normal distribution.

We use the considered methods (except for QReg Forest) to fit a linear model for three τ values (0.25, 0.50, and 0.75). The generating-training-testing procedure was repeated 100 times. For every τ and each algorithm, we report the average and standard deviation of the obtained 100 mean absolute deviations in Table 1.

From Table 1, we observe that, QReg Forest performs the worst because the underlying true model is linear. SV-QReg, IP-QReg, MM-QReg, and the QReg function in R package perform similarly in terms of mean absolute deviation, while Newton QReg without L_1 constraint yields slightly smaller mean testing error with smaller standard deviations.

To test the performance of the L_1 constrained models, we generate an independent validation set of size 10,000 from the model in Eq. (33), and select the parameter λ which minimizes the check loss function on the validation set. L_1 constrained model is known for being able to select informative variables (Tibshirani 1996). In the obtained linear model, we calculate the sum of the absolute values of all the estimated coefficients. If the absolute value of the estimated coefficient of a predictor is less than 1 % of the total sum, that means the contribution of this variable to the model is small, thus it could be trimmed out. We calculate the average numbers of correctly and mistakenly deleted predictors to measure the variable selection performance.

We test the performance of the developed L_1 Newton QReg and the L_1 QReg function in the R package, and list the testing errors and variable selection measure in Table 2. By comparing to the testing errors in Table 1, we observe that the L_1 constrained models achieve more precise prediction. The reason is that the L_1 constraint helps remove the effect from the noisy predictors by utilizing information from the (large) validation set. Table 2 shows that, compared to the function from the R package, the L_1 Newton QReg has slightly worse performance in testing accuracy while slightly better performance in variable selection, we thus can say that overall, they perform on the same level.

Simulation 2 (*Dimensionality larger than the sample size*) In this simulation, we generate data from the model given in Eq. (33), and augment the predictor vector with 92 noisy variables $x_9, x_{10}, \ldots, x_{100}$, each of which is generated independently from standard normal distribution. The training set consists of 50 examples, in this way, the dimensionality of the data is much larger than the training sample size, which makes the estimation more difficult. For the purpose of parameter selection in the L_1 constrained model, we generate an independent validation set of size 10,000 from the same model. In the L_1 constrained linear quantile regression model, the regularization parameter λ is selected by minimizing the check loss on the validation set. The testing set consists of 10,000 examples. In this experiment, most of the predictors have no contribution to the response, therefore, it is desirable to identify the informative predictors (i.e., x_1, x_2, and x_5) and suppress the noisy ones. In order to make the estimation possible, we set the parameter σ in Eq. (33) as $\sigma = 1$. We adopt the same variable selection rule and performance measure as in Simulation 1.

For three τ values (0.25, 0.5, and 0.75), the experiment was repeated 100 times, for several considered alternative quantile regression algorithms.[7] Table 3 presents the performance measures. It is not clear how to identify noninformative predictors by the QReg Forest, thus the variable selection result of QReg Forest is not reported.

From Table 3, we notice that SV-QReg and Newton QReg without L_1 constraint only delete about 60 % of the 97 noisy variables in all the situations. We would expect that the undeleted noisy variables degrade the performance of the models, as shown in Table 3 that the prediction accuracies are very poor. With L_1 constraint, Newton QReg deletes about 93 % of the noninformative variables in average, which is the best in the considered methods. We also notice that the L_1 constraint also prevents us from deleting the informative predictors, as the average number of mistakenly deleted variables is 0. The variable selection ability of the L_1 constrained models enable us to yield far better prediction accuracy, as shown in Table 3. We finally observe that on this dataset, the proposed method performs better than the R function in terms of both prediction accuracy and the variable selection ability.

[7] In IP-QReg and MM-QReg algorithms, there is a step which needs to invert a $p \times p$ matrix, whose rank is at most n, where p is the dimensionality of the data, and n is the training set size. When the dimensionality is greater than the sample size ($p > n$), the matrix is not invertible, thus the downloaded software package for IP-QReg and MM-QReg gives error message. Therefore, the performances of IP-QReg and MM-QReg are not provided.

Table 3 The performance of several algorithms on datasets with dimensionality larger than the sample size

τ	Method	Test error	# of trimmed variables	
			Correct	Wrong
.25	QReg Forest	2.986 (0.255)	NA	NA
	SV-QReg	2.318 (0.190)	55.27 (2.86)	0.98 (0.82)
	N-QReg	2.283 (0.177)	56.91 (2.93)	0.86 (0.79)
	L_1 N-QReg	**0.540 (0.163)**	**91.09 (2.57)**	**0.00 (0.00)**
	L_1 QReg (R)	0.785 (0.138)	83.07 (2.72)	0.00 (0.00)
.50	QReg Forest	2.345 (0.259)	NA	NA
	SV-QReg	2.254 (0.181)	63.55 (3.06)	0.00 (0.00)
	N-QReg	2.218 (0.177)	63.80 (3.16)	0.00 (0.00)
	L_1 N-QReg	**0.504 (0.111)**	**89.40 (2.64)**	**0.00 (0.00)**
	L_1 QReg (R)	0.807 (0.125)	79.85 (2.59)	0.00 (0.00)
.75	QReg Forest	2.861 (0.259)	NA	NA
	SV-QReg	2.319 (0.197)	55.76 (2.61)	0.75 (0.64)
	N-QReg	2.285 (0.177)	56.88 (3.06)	0.62 (0.60)
	L_1 N-QReg	**0.536 (0.131)**	**90.76 (2.84)**	**0.00 (0.00)**
	L_1 QReg (R)	0.775 (0.111)	82.93 (2.38)	0.00 (0.00)

The best performance is marked in bold

Simulation 3 (*Heteroscedastic random errors*) This experiment considers the case of heteroscedastic random errors to check the robustness of our method. Following Wu and Liu (2009), we generate data from the model

$$y = 1 + x_1 + x_2 + x_3 + (1 + x_3)\epsilon, \tag{34}$$

where x_1 and x_3 are respectively generated from the standard normal distribution and the uniform distribution on [0, 1], $x_2 = x_1 + x_3 + z$ with z being standard normal, and $\epsilon \sim N(0, 1)$. The variables x_1, x_3, z, and ϵ are mutually independent. To study the effect of variable selection, we include five more standard normal noisy variables, $x_4, \ldots, x_8$, independent of each other.

The generating-training-testing process was repeated 100 times with training set size 100, validation set size 10,000, and testing set size 10,000. Table 4 shows the testing accuracy of several considered methods, measured in terms of mean absolute deviation. The results once again indicate that the proposed Newton Qreg algorithm yields similar performance to several alternatives.

We also compared the performances of the L_1 Newton QReg and the L_1 constrained QReg function in R package, in terms of prediction accuracy and variable selection ability. We adopt a strategy similar to Simulation 1 for variable selection, and the results are given in Table 5. First of all, by comparing Tables 4 and 5, we clearly observe that the L_1 constraint has helped improve the prediction accuracy significantly due to suppressing the noisy predictors. Table 5 shows that, compared to the L_1 QReg function in R package, the proposed L_1 Newton QReg achieves slightly greater testing errors; however, it correctly deletes more uninformative variables. Thus, we

Table 4 The testing error of QReg Forest, IP-QReg, SV-QReg, MM-QReg, the proposed Newton QReg, and the QReg function in R package on the dataset with heteroscedastic error terms

Method	QReg Forest	IP-QReg	SV-QReg	MM-QReg	N-QReg	QReg R
$\tau = .25$	.699 (.091)	.496 (.124)	.496 (.124)	.493 (.120)	.489 (.117)	.496 (.124)
$\tau = .50$	.683 (.089)	.460 (.126)	.460 (.126)	.457 (.126)	.452 (.125)	.460 (.126)
$\tau = .75$	.768 (.102)	.502 (.126)	.502 (.126)	.502 (.126)	.494 (.129)	.502 (.126)

Table 5 The performances of L_1 Newton QReg and the L_1 QReg in R package on the data in Simulation 3

τ	Method	Test error	# of trimmed variables	
			Correct	Wrong
.25	L_1 N-QReg	.411 (.139)	2.08 (1.14)	0.53 (0.50)
	L_1 QReg (R)	.362 (.114)	1.93 (0.98)	0.59 (0.49)
.50	L_1 N-QReg	.405 (.114)	2.01 (1.16)	0.34 (0.48)
	L_1 QReg (R)	.376 (.110)	1.77 (1.12)	0.22 (0.41)
.75	L_1 N-QReg	.458 (.130)	1.98 (1.24)	0.20 (0.40)
	L_1 QReg (R)	.421 (.126)	1.63 (1.18)	0.05 (0.22)

can say that the two L_1 constrained models have similar overall performances. In this simulation, both of the algorithms produce more wrong zero coefficients in the final model, compared to Simulation 2. A possible reason is that x_2 is highly correlated with x_1 and x_3 through $x_2 = x_1 + x_3 + z$. Nevertheless, it is clear that the L_1 constrained algorithm can result in a sparse model and improve the prediction accuracy significantly.

Simulation 4 (*Kernel quantile regression models*) In order to test the performances of various kernel quantile regression models, we generate data according to

$$\begin{aligned} y = & 40 \exp\left[8\left((x_1 - 0.5)^2 + (x_2 - 0.5)^2\right)\right] \\ & \times \left(\exp\left[8\left((x_1 - 0.2)^2 + (x_2 - 0.7)^2\right)\right]\right. \\ & \left. + \exp\left[8\left((x_1 - 0.7)^2 + (x_2 - 0.2)^2\right)\right]\right)^{-1} + \epsilon, \end{aligned} \tag{35}$$

where x_1 and x_2 are generated from uniform distribution over (0, 1); we consider two different distributions for the error term ϵ: standard normal distribution and standard Laplace distribution (i.e., double exponential distribution). The model in Eq. (35) was given in Yuan (2006) and was also studied in Li et al. (2007).

With the model in Eq. (35), we generate 200 training examples associated with each error distribution, along with 10,000 observations for validation and 10,000 for

Table 6 Under different error terms, the performances of the SV-QReg, kernel QReg of Li et al. (2007), and the proposed Newton kernel QReg with and without L_1 constraint

Method		Newton L_1 K-QReg	Newton K-QReg	SV-QReg	Li et al. (2007)
Normal	$\tau = .1$	.366 (.060)	.506 (.073)	.816 (.075)	.478 (.072)
	$\tau = .3$	.309 (.048)	.354 (.047)	.685 (.079)	.368 (.052)
	$\tau = .5$	.295 (.046)	.331 (.040)	.655 (.079)	.348 (.048)
Laplace	$\tau = .1$	.418 (.069)	.482 (.064)	.842 (.082)	.672 (.114)
	$\tau = .3$	.261 (.039)	.289 (.040)	.609 (.076)	.423 (.061)
	$\tau = .5$	.228 (.035)	.263 (.036)	.555 (.068)	.355 (.060)

Listed are the mean values of the Mean Absolute Deviation of the 100 runs, and the standard deviations are listed in parentheses

Table 7 Under Normal error and Laplace error, the sparsity of the model obtained by the Newton kernel QReg with L_1 constraint

Error term	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$
Normal	168.31 (17.88)	168.15 (25.19)	168.09 (24.70)
Laplace	171.81 (6.90)	167.15 (18.08)	163.95 (29.77)

For each of the tested τ value, we list the average number of zero coefficients in model (28) with the standard deviation listed in parentheses

testing purpose. In all the algorithms, we used the radial basis kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left\{-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right\}$$

with $\sigma = 0.2$, which is with the same setting as in Li et al. (2007).

For each of the two error distributions, we compare the performance of our proposed Newton kernel QReg with and without L_1 constraint, the SV-QReg of Hwang and Shim (2005), and the kernel QReg of Li et al. (2007). Similar to Li et al. (2007), we consider three different values for τ: 0.1, 0.3, and 0.5. For the models with L_1 constraint, we select the parameter λ that minimizes the check loss on the validation set.

As before, the process of generating-training-testing was repeated 100 times. Table 6 shows the testing accuracies of different models under the two error distributions, where the results of kernel QReg of Li et al. (2007) were obtained directly from Li et al. (2007). It is easily observed that, without the L_1 constraint, the SV-QReg model of Hwang and Shim (2005) performs significantly worse than the Newton kernel QReg without L_1; the model in Li et al. (2007) performs significantly worse than the proposed Newton kernel QReg with L_1 constraint, in terms of mean absolute deviation error.

In the obtained model from the proposed L_1 Newton kernel QReg, if the absolute value of any coefficient β_i in Eq. (28) is less than 10^{-6}, it could be counted as zero. We use the number of zeros as a sparsity measure for the obtained model. Table 7 shows the average number of zero coefficients in Eq. (28) for each error distribution and each tested τ value. Recall that the total number of coefficients is the training set size, which

is 200 in this simulation. We easily observe that, with the L_1 constraint, more than 80 % of the obtained coefficients are actually zero. On the contrary, without the L_1 constraint, after applying the same rule, the obtained model is dense (the results are not listed).

In our simulations, we used the average check loss on the validation set as the parameter selection criterion. It is also possible to use other selection criteria such as the Schwarz information criterion (Schwarz 1978; Koenker et al. 1994) (SIC) and the generalized approximate cross-validation criterion (Yuan 2006) (GACV), which are also used in Li et al. (2007) and Li and Zhu (2008). We notice that the criterion we used (i.e., average check loss) was called gold standard in Li et al. (2007) and Li and Zhu (2008). We also ran the simulations with SIC and GACV criteria, and we observed the similar performance as reported in this section. Due to the space limit, we choose not to present the results with SIC and GACV criteria.

6 Results on real-world data

This section compares the performance of the proposed Newton quantile regression algorithm to alternative quantile regression algorithms on two real-world datasets from UCI Machine Learning Repository. The parameter setting of the algorithms is the same as in Sect. 5. On the real datasets, although the relevant variables are not known to us, a sparse model is easy to understand and interpret, thus we still prefer a model which can delete more variables. In this section, we adopt the same strategy as in Simulation 1 of Sect. 5 for variable selection. The considered algorithms include QReg Forest, IP-QReg, MM-QReg, SV-QReg with linear kernel, the QReg function in R package with and without the L_1 constraint, and the proposed Newton QReg with and without L_1 constraint. For every algorithm, in each iteration, we used the same training set, the same validation set (if necessary), and the same testing set, in order to make the comparison fair.

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g., there is no data about grape types, wine brand, wine selling price, etc.). For more details, please refer to Cortez et al. (2009). There are 11 predictor variables, which are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH value, sulphates, and alcohol. We normalize each of these variables to have mean 0 and unit standard deviation, and augment the predictors by including the squares of the standardized variables, resulting in 22 predictors in total. The response variable is the wine quality which is a score between 0 and 10. We standardized it to have mean 0 and unit standard deviation. There are 1,599 instances for red wine and 4,898 instances for white wine. To numerically evaluate the performances, in lack of the true quantile functions for the datasets, we adopt the average of the "check loss" on the testing set as the error measure, which is defined as

$$L(\tau) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \rho_\tau \left(y_i - \hat{q}_\tau(\mathbf{x}_i) \right), \tag{36}$$

Table 8 The performances of QReg Forest, IP-QReg, SV-QReg, MM-QReg, Newton QReg, and the QReg function in R package on the red wine quality data

Method	QReg Forest	IP-QReg	SV-QReg	MM-QReg	N-QReg	QReg R
$\tau = .25$	.2204 (.0092)	.2535 (.0054)	.2549 (.0058)	.2549 (.0062)	.2532 (.0051)	.2537 (.0057)
$\tau = .50$	.2619 (.0101)	.3129 (.0054)	.3134 (.0068)	.3136 (.0063)	.3114 (.0064)	.3127 (.0054)
$\tau = .75$	.2283 (.0120)	.2558 (.0069)	.2553 (.0070)	.2563 (.0070)	.2548 (.0061)	.2567 (.0085)

Listed are the mean values of the "check loss" of the 100 runs, and the standard deviations are listed in parentheses

Table 9 The performances of the considered algorithms on the white wine quality data

Method	QReg Forest	IP-QReg	SV-QReg	MM-QReg	N-QReg	QReg R
$\tau = .25$	.2162 (.0039)	.2582 (.0030)	.2580 (.0027)	.2584 (.0024)	.2576 (.0026)	.2575 (.0023)
$\tau = .50$	.2829 (0066)	.3264 (.0032)	.3262 (.0029)	.3266 (.0028)	.3256 (.0032)	.3259 (.0029)
$\tau = .75$	.2412 (0.0077)	.2572 (.0026)	.2570 (.0026)	.2567 (.0026)	.2565 (.0026)	.2572 (.0028)

See the caption of Table 8

where N_{test} is the size of the testing set, and $\hat{q}_\tau(\mathbf{x}_i)$ is the estimated τ-th quantile at $\mathbf{x}_i$. By the definition of quantile, the smaller the value of $L(\tau)$, the closer the estimated quantile to the true quantile.

To train the model on the red wine dataset, we randomly select 500 observations, and the rest are used as testing set; on the white wine dataset, the training set size is 1,500 and the rest is for testing. That is, we use about 30 % of the whole dataset for training, and about 70 % for testing. For three τ values (0.25, 0.50, and 0.75), the partition-training-testing process is repeated 100 times for each algorithm. The mean and standard deviation of the 100 check losses are calculated. Tables 8 and 9 show the performances of the considered quantile regression algorithms on the red wine data and white wine data, respectively. We observe that, on both datasets, the quantile regression forest yields the best prediction accuracy. This shows that as a nonparametric model, the regression forest model provides flexibility in model fitting. Compared to alternative parametric methods, the proposed Newton QReg algorithm can achieve smaller testing errors with often smaller standard deviations, although the advantage is not dominating.

We also compared the performance (testing error and noisy predictor suppression) of the proposed L_1 constrained Newton quantile regression to the corresponding function in the R package. In the L_1 constrained models, we need a validation set to select the regularization parameter λ. A validation set of the same size as the training set (i.e., for red wine, it is 500, and 1,500 for the white wine data) is selected randomly. The parameter λ which minimizes the check loss on the validation set is selected. We calculate the average number of trimmed predictors to measure the variable selection performance. Table 10 shows the testing errors and the variable selection performances of the considered algorithms. Compared to the L_1 QReg function in the R package, the proposed Newton L_1 QReg algorithm can achieve smaller testing errors with often smaller standard deviations. We also observe that the Newton L_1 QReg can remove

Table 10 Comparison of the L_1 constrained Newton QReg and the function in the R package

τ	Method	Red wine		White wine	
		Check error	# D. V.	Check error	# D. V.
.25	Newton	.2462 (.0096)	7.49 (1.73)	.2562 (.0042)	6.88 (1.19)
	R package	.2497 (.0101)	6.84 (1.66)	.2573 (.0048)	6.72 (1.70)
.50	Newton	.3049 (.0093)	7.56 (1.78)	.3238 (.0043)	7.18 (1.51)
	R package	.3097 (.0109)	7.19 (1.83)	.3253 (.0051)	6.58 (1.73)
.75	Newton	.2462 (.0073)	8.01 (1.75)	.2554 (.0039)	6.10 (1.37)
	R package	.2520 (.0094)	7.00 (1.72)	.2565 (.0047)	5.97 (1.75)

For three τ values, the mean values of the "check loss" of the 100 runs are shown with the standard deviations listed in parentheses. The means and standard deviations of the number of deleted variables (# D. V.) are listed as a measure of variable selection performance

more predictors in average (with a smaller standard deviation), resulting in sparser models.

7 Conclusions

This paper studies the linear programming associated with linear quantile regression model, and formulates the quadratic penalty function for its dual. We proved that a quantile regression model can be obtained by minimizing the quadratic penalty function without constraint, and derived the explicit formulas. The proposed algorithms are easy to implement, without requiring any additional sophisticated optimization software package other than a linear equation solver. A generalized Newton algorithm with Armijo step size is proposed to minimize the quadratic penalty function. With slight modification, the proposed approach can be generalized to fit quantile regression model in reproducing kernel Hilbert space. Extensive experiments on simulated data and real-world data show that, the proposed Newton quantile regression algorithms can achieve performance comparable to state-of-the-art.

In this paper, we used the quadratic penalty function to absorb the constraints in the linear programming for the linear quantile regression model. In literature, there are other methods to convert a constrained optimization problem to an unconstrained one, for example, the exponential method of multipliers (Bertsekas 1999, Sect. 4.2.5), and the logarithm barrier method (Ruszczyński 2006, Sect. 6.6). It would be interesting to compare different methods from the theoretical and practical aspects.

Acknowledgments The author would like to extend his sincere gratitude to the anonymous reviewers and editors for their constructive suggestions and comments, which have greatly helped improve the quality of this paper. This work was supported by a Faculty Research Grant from Missouri State University.

Appendix: The proof of Eq. (25)

If $a > \lambda > 0$, then

$$a - \lambda > 0, \quad -a - \lambda < 0, \quad \text{and} \quad |a| - \lambda > 0,$$

thus, from the definition of $(\cdot)_*$,

$$(a-\lambda)_* + (-a-\lambda)_* = 1 = (|a|-\lambda)_*.$$

If $a < -\lambda < 0$, we have

$$a-\lambda < 0, \quad -a-\lambda > 0, \quad \text{and} \quad |a|-\lambda > 0,$$

thus,

$$(a-\lambda)_* + (-a-\lambda)_* = 1 = (|a|-\lambda)_*.$$

If $-\lambda < a < \lambda$, it implies that

$$a-\lambda < 0, \quad -a-\lambda < 0, \quad \text{and} \quad |a|-\lambda < 0,$$

thus,

$$(a-\lambda)_* + (-a-\lambda)_* = 0 = (|a|-\lambda)_*.$$

In summary, it is checked for all the cases that $(a-\lambda)_* + (-a-\lambda)_* = (|a|-\lambda)_*$.

References

Armijo L (1966) Minimization of functions having Lipschitz-continuous first partial derivatives. Pac J Math 16:1–3

Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Scientific, Belmont, MA

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York, NY

Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27

Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 47(4):547–553

Fung G, Mangasarian OL (2004) A feature selection Newton method for support vector machine classification. Comput Optim Appl 28(2):185–202

Gunn SR (1997) Support vector machines for classification and regression. Technical Report, Image Speech and Intelligent Systems Research Group, University of Southampton. http://users.ecs.soton.ac.uk/srg/publications/pdf/SVM

Higham N (2002) Accuracy and stability of numerical algorithms, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia, PA

Hiriart-Urruty J-B, Strodiot J-J, Nguyen VH (1984) Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. Appl Math Optim 11(1):43–56

Hunter DR, Lange K (2000) Quantile regression via an MM Algorithm. J Comput Graph Stat 9(1):60–77

Hwang C, Shim J (2005) A simple quantile regression via support vector machine. In: Lecture notes in computer science, vol 3610, pp 512–520

Joachims T (1999) Making large-scale SVM learning practical. In: Schölkopf B, Burges C, Smola A (eds) Advances in kernel methods—support vector learning. MIT Press, Cambridge, MA

Kimeldorf GS, Wahba G (1970) Some results on Tchebycheffian spline functions. J Math Anal Appl 33(1):82–95

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

Koenker R, Bassett G (1978) Regression quantiles. Econometrica 46:33–50

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

Koenker R, Park BJ (1996) An interior point algorithm for nonlinear quantile regression. J Econ 71:265–283
Langford J, Oliveira R, Zadrozny B (2006) Predicting conditional quantiles via reduction to classification. In: Proceedings of the uncertainty in artificical intelligence. Cambridge, MA
Li C, Wei Y, Chappell R, He X (2011) Bent line quantile regression with application to an allometric study of land Mammals' speed and mass. Biometrics 67(1):242–249
Li Y, Liu Y, Zhu J (2007) Quantile regression in reproducing kernel Hilbert spaces. J Am Stat Assoc 102:255–268
Li Y, Zhu J (2008) L_1-norm quantile regression. J Comput Graph Stat 17(1):163–185
Mangasarian OL (2006) Exact 1-norm support vector machines via unconstrained convex differentiable minimization. J Mach Learn Res 7:1517–1530
Mangasarian OL, Meyer RR (1979) Nonlinear perturbation of linear programs. SIAM J Control Optim 17(6):745–752
Meinshausen N (2006) Quantile regression forests. J Mach Learn Res 7:983–999
Ruszczyński A (2006) Nonlinear optimization. Princeton University Press, Princeton, NJ
Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6:461–464
Sohn I, Kim S, Hwang C, Lee JW (2008) New normalization methods using support vector machine quantile regression approach in microarray analysis. Comput Stat Data Anal 52(8):4104–4115
Sohn I, Kim S, Hwang C, Lee JW, Shim J (2008) Support vector machine quantile regression for detecting differentially expressed genes in microarray analysis. Methods Inf Med 47(5):459–467
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc B 58(1):267–288
Takeuchi I, Le QV, Sears TD, Smola AJ (2006) Nonparametric quantile estimation. J Mach Learn Res 7:1231–1264
Wu Y, Liu Y (2009) Variable selection in quantile regression. Stat Sin 19:801–817
Yuan M (2006) GACV for quantile smoothing splines. Comput Stat Data Anal 50(3):813–829