Expert Systems with Applications 39 (2012) 1687–1697

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

QBoost: Predicting quantiles with boosting for regression and binary classification

Songfeng Zheng *

Department of Mathematics, Missouri State University, 901 S. National Ave., Springfield, MO 65897, USA

ARTICLE INFO

Keywords:
Quantile regression
Boosting
Functional gradient algorithm
Binary classification

ABSTRACT

In the framework of functional gradient descent/ascent, this paper proposes Quantile Boost (QBoost) algorithms which predict quantiles of the interested response for regression and binary classification. Quantile Boost Regression performs gradient descent in functional space to minimize the objective function used by quantile regression (QReg). In the classification scenario, the class label is defined via a hidden variable, and the quantiles of the class label are estimated by fitting the corresponding quantiles of the hidden variable. An equivalent form of the definition of quantile is introduced, whose smoothed version is employed as the objective function, and then maximized by functional gradient ascent to obtain the Quantile Boost Classification algorithm. Extensive experimentation and detailed analysis show that QBoost performs better than the original QReg and other alternatives for regression and binary classification. Furthermore, QBoost is capable of solving problems in high dimensional space and is more robust to noisy predictors.

1. Introduction

Classical least square regression aims to estimate the conditional expectation of the response Y given the predictor (vector) $\mathbf{x}$, i.e., $E(Y|\mathbf{x})$. However, the mean value (or the conditional expectation) is sensitive to the outliers of the data (Koenker, 2005). Therefore, if the data is not homogeneously distributed, we expect the traditional least square regression to give us a poor prediction.

The τth quantile of a distribution is defined as the value such that there is $100\tau\%$ of mass on its left side. Compared to the mean value, quantiles are more robust to outliers (Koenker, 2005). Let $I(\cdot)$ be the indicator function with $I(\cdot)=1$ if the condition is true, otherwise $I(\cdot)=0$. Let $Q_\tau(Y)$ be the τth quantile of random variable Y. It can be proved (Hunter & Lange, 2000) that

$$Q_\tau(Y) = \arg\min_c E_Y[\rho_\tau(Y-c)],$$

where $\rho_\tau(r)$ is the "check function" (Koenker, 2005) defined by

$$\rho_\tau(r) = rI(r \geqslant 0) - (1-\tau)r. \tag{1}$$

Given training data $\{(\mathbf{x}_i, Y_i),\ i=1,\ldots,n\}$, with predictor vector $\mathbf{x}_i \in \mathbf{R}^d$ and response $Y_i \in \mathbf{R}$, let the τth conditional quantile of Y given $\mathbf{x}$ be $f(\mathbf{x})$. Similar to the least square regression, quantile regression (QReg) (Koenker, 2005; Koenker & Bassett, 1978) aims at estimating the conditional quantiles of the response given a predictor vector $\mathbf{x}$ and can be formulated as

$$f^*(\cdot) = \arg\min_f \frac{1}{n}\sum_{i=1}^{n}\rho_\tau(Y_i - f(\mathbf{x}_i)). \tag{2}$$

Compared to least square regression, quantile regression is robust to outliers in observations, and can give a more complete view of the relationship between predictor and response. Furthermore, least square regression implicitly assumes normally distributed errors, while such an assumption is not necessary in quantile regression. Since being introduced in Koenker and Bassett (1978), quantile regression has become a popular and effective approach to statistical analysis with wide applications in economics (Hendricks & Koenker, 1992; Koenker & Hallock, 2001), survival analysis (Koenker & Geling, 2001), and ecology (Cade & Noon, 2003), to name a few.

The quantile regression model in Eq. (2) can be solved by linear programming algorithms (Koenker, 2005) or Majorize-Minimize algorithms (Hunter & Lange, 2000). However, when the predictor $\mathbf{x}$ is in high dimensional space, the aforementioned optimization methods for QReg might be inefficient. High dimensional problems are ubiquitous in applications such as image analysis, gene sequence analysis, etc. To the best of our knowledge, the problem of high dimensional predictor is not sufficiently addressed in QReg literature, and this paper proposes a method for QReg which can work in high dimensional spaces.

The proposed algorithm for QReg is based on boosting (Freund & Schapire, 1997), which is well known for its simplicity and good performance. The powerful feature selection mechanism of boosting makes it suitable to work in high dimensional spaces. Friedman, Hastie, and Tibshirani (2000) developed a general statistical framework which yields a direct interpretation of boosting as a

* Tel.: +1 417 836 6037; fax: +1 417 836 6966.
E-mail address: SongfengZheng@MissouriState.edu

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.06.060

method for function estimation, which is a "stage-wise, additive model".

Consider the problem of function estimation

$$f^*(\mathbf{x}) = \arg\min_f E[l(Y, f(\mathbf{x}))|\mathbf{x}],$$

where $l(\cdot,\cdot)$ is a loss function which is typically differentiable and convex with respect to the second argument. Estimating $f^*(\cdot)$ from the given data, $\{(\mathbf{x}_i, Y_i), i = 1, \ldots, n\}$, can be performed by minimizing the empirical loss, $n^{-1}\sum_{i=1}^n l(Y_i, f(\mathbf{x}_i))$, and pursuing iterative steepest descent in functional space. This leads us to the generic functional gradient descent algorithm (Friedman, 2001; Mason, Baxter, Bartlett, & Frean, 2000). Fig. 1 shows the version summarized in Bühlmann and Hothorn (2007).

Many boosting algorithms can be understood as functional gradient descent with appropriate loss function. For example, if we choose $l(Y, f) = \exp(-(2Y-1)f)$, we would recover AdaBoost (Friedman et al., 2000), and L_2 Boost (Bühlmann & Yu, 2003) corresponds to $l(Y, f) = (Y-f)^2/2$.

Motivated by the gradient boosting algorithms (Friedman, 2001; Mason et al., 2000), this paper estimates the quantile regression function by minimizing the objective function in Eq. (2) with functional gradient descent. In each step, we approximate the negative gradient of the objective function by a base function, and grow the model in that direction. This results in the Quantile Boost Regression (QBR) algorithm. In the binary classification scenario, we define the class label via a hidden variable, and the quantiles of the class label can then be estimated by fitting the corresponding quantiles of the hidden variable. An equivalent form of the definition of quantile is introduced, whose smoothed version is employed as the objective function for classification. Similar to QBR, functional gradient ascent is applied to maximize the objective function, which yields the Quantile Boost Classification (QBC) algorithm. The obtained Quantile Boost (QBoost) algorithms are computationally efficient and converge to local optima. More importantly, they enable us to solve high dimensional problems efficiently.

The rest of this paper is organized as follows: in Section 2, we first apply the functional gradient descent to QReg, yielding the QBR algorithm, and then we proceed to propose the approximation of the objective function for binary classification and to maximize the objective function with functional gradient ascent in order to obtain the QBC algorithm; Section 3 discusses some computational issues in the proposed QBR and QBC algorithms and introduces implementation details; Section 4 presents the experimental results of the proposed QBR and QBC algorithms on benchmark regression and binary classification datasets, and in-depth discussions of the results are also presented in Section 4; finally, Section 5 summarizes this paper with a brief discussion for future research directions.

2. Methods

The methods proposed in our research are presented in this section. We first directly apply the functional gradient descent to the quantile regression model, yielding the quantile boost regression algorithm. We then propose a smooth approximation to the optimization problem for the quantiles of binary response, and based on this we further propose the quantile boost classification algorithm with some discussions of the related methods.

2.1. Quantile boost regression

We consider the problem of estimating quantile regression function in the general framework of functional gradient descent with the loss function

$$l(Y, f) = \rho_\tau(Y-f) = (Y-f)I(Y-f \geqslant 0) - (1-\tau)(Y-f).$$

A direct application of the algorithm in Fig. 1 yields the Quantile Boost Regression (QBR) algorithm, which is shown in Fig. 2.

Similar to AdaBoost, QBR enables us to select most informative predictors if an appropriate base learner is employed, and this will be demonstrated experimentally in Section 4.1.1.

There is a large volume of literature applying boosting to regression problems, for example in Duffy and Helmbold (2002), Freund and Schapire (1997), and Zemel and Pitassi (2001). However, all these methods estimate the mean value of the response, not quantiles. Langford, Oliveira, and Zadrozny (2006) proposed to use classification technique for estimating the conditional

0. Set iteration number $m = 0$, initialize $f^{[0]}(\cdot) = 0$ or

$$f^{[0]}(\cdot) = \arg\min_c \frac{1}{n}\sum_{i=1}^n l(Y_i, c).$$

1. Increase m by 1. Compute the negative gradient $-\frac{\partial}{\partial f} l(Y, f)$ at $f^{[m-1]}(\mathbf{x}_i)$:

$$U_i = -\left.\frac{\partial l(Y_i, f)}{\partial f}\right|_{f=f^{[m-1]}(\mathbf{x}_i)}, \quad i = 1, \cdots, n.$$

2. Fit the negative gradient vector $U_1, \cdots, U_n$ to $\mathbf{x}_1, \cdots, \mathbf{x}_n$ by a base procedure (e.g., the weak learner in AdaBoost):

$$\{(\mathbf{x}_i, U_i), i = 1, \cdots, n\} \to g^{[m]}(\cdot).$$

3. Update the estimation by $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \eta_m g^{[m]}(\cdot)$, where η_m is a step size factor.

4. Check the stopping criterion; if not satisfied, go to step 1.

Fig. 1. Generic functional gradient descent.

0. Given the training data $\{(\mathbf{x}_i, Y_i),\ i = 1, \cdots, n\}$, the desired quantile value τ, and the total number of iterations M, initialize $f^{[0]}(\cdot) = 0$ or set $f^{[0]}(\cdot) = \tau$-th quantile of $(Y_1, \cdots, Y_n)$.
1. **for** $m = 1$ to M **do**:
2. Compute the negative gradient $-\frac{\partial}{\partial f}\rho_\tau(Y - f)$ and evaluate at $f^{[m-1]}(\mathbf{x}_i)$:

$$U_i = I(Y_i - f^{[m-1]}(\mathbf{x}_i) \geq 0) - (1 - \tau), \quad i = 1, \cdots, n.$$

3. Fit the negative gradient vector $U_1, \cdots, U_n$ to $\mathbf{x}_1, \cdots, \mathbf{x}_n$ by the base procedure

$$\{(\mathbf{x}_i, U_i), i = 1, \cdots, n\} \to g^{[m]}(\cdot).$$

4. Update the estimation by $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \eta_m g^{[m]}(\cdot)$, where η_m is a step size factor.
5. **end for**
6. Output the estimated τ-th quantile function $f^{[M]}(\mathbf{x})$.

Fig. 2. Quantile boost regression (QBR) algorithm.

quantile. For each 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, compared to the proposed QBR algorithm, this method is time consuming. Furthermore, it is not clear how to perform variable selection using the method in Langford et al. (2006). Takeuchi, Le, Sears, and Smola (2006) and Li, Liu, and Zhu (2007) perform quantile regression in Reproducing Kernel Hilbert Spaces (RKHS). Although RKHS might be of very high dimensionality, the models in Takeuchi et al. (2006) and Li et al. (2007) require quadratic programming which is computationally more expensive than the proposed QBR model. Moreover, the models in Takeuchi et al. (2006) and Li et al. (2007) cannot perform variable selection as does QBR. In an independent study which applied the functional gradient method, Kriegler & Berk (2007) weighted the absolute loss function according to overestimates and underestimates, resulting an algorithm similar to QBR. However, the method in (Kriegler & Berk, 2007) uses multi-node regression tree as weak learner, and needs to solve multiple optimization problems in each iteration, therefore, it is more time consuming compared to QBR.

2.2. Quantile boost classification

2.2.1. Predicting quantiles of binary response

Consider the following model:

$$Y^* = h(\mathbf{x}) + \varepsilon \quad \text{and} \quad Y = I\{Y^* \geqslant 0\}, \tag{3}$$

where Y^* is a continuous hidden variable, $h(\mathbf{x})$ is the true model for Y^*, ε is a disturb, and $Y \in \{0, 1\}$ is the binary class label for the observation $\mathbf{x}$. The model in Eq. (3) was also considered in Koenker (2005), Kordas (2002, 2006), and Manski (1985).

Let $Q_\tau(Y^*|\mathbf{x})$ be the τth conditional quantile of Y^* given $\mathbf{x}$, and let $g(\cdot)$ be a real monotone increasing function. Clearly

$$P(Y^* \geqslant y|\mathbf{x}) = P(g(Y^*) \geqslant g(y)|\mathbf{x}),$$

so it follows that

$$g(Q_\tau(Y^*|\mathbf{x})) = Q_\tau(g(Y^*)|\mathbf{x}). \tag{4}$$

Since the indicator function, $I(t \geqslant 0)$, is monotone increasing with respect to t, Eq. (4) indicates that

$$I(Q_\tau(Y^*|\mathbf{x}) \geqslant 0) = Q_\tau(I(Y^* \geqslant 0)|\mathbf{x}) = Q_\tau(Y|\mathbf{x}),$$

that is, the conditional quantile function of Y can be obtained by fitting the corresponding conditional quantile function of Y^* (Koenker, 2005; Kordas, 2006). Let the τth conditional quantile function of the latent variable Y^* be $f(\mathbf{x}, \boldsymbol{\beta})$ with $\boldsymbol{\beta}$ as the parameter vector, i.e.,

$$Q_\tau(Y^*|\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\beta}). \tag{5}$$

It follows that the conditional quantile function of the binary variable Y can be modeled as

$$Q_\tau(Y|\mathbf{x}) = I(f(\mathbf{x}, \boldsymbol{\beta}) \geqslant 0). \tag{6}$$

From the relation $Y = I\ (Y^* \geqslant 0)$, we have

$$P(Y = 1|\mathbf{x}) = P(Y^* \geqslant 0|\mathbf{x}), \tag{7}$$

while Eq. (5) implies that

$$P(Y^* \geqslant f(\mathbf{x}, \boldsymbol{\beta})|\mathbf{x}) = 1 - \tau. \tag{8}$$

Thus, if $f(\mathbf{x}, \boldsymbol{\beta}) = 0$, combining Eqs. (7) and (8) yields

$$P(Y = 1|\mathbf{x}) = P(Y^* \geqslant f(\mathbf{x}, \boldsymbol{\beta})|\mathbf{x}) = 1 - \tau.$$

If $f(\mathbf{x}, \boldsymbol{\beta}) > 0$,

$$P(Y = 1|\mathbf{x}) > P(Y^* \geqslant f(\mathbf{x}, \boldsymbol{\beta})|\mathbf{x}) = 1 - \tau.$$

In summary, we have the following relation:

$$P(Y = 1|\mathbf{x}) =<> 1 - \tau \quad \text{for} \quad f(\mathbf{x}, \boldsymbol{\beta}) =<> 0, \tag{9}$$

which is an inequality of the posterior probability of the binary class label given the predictor vector. Hence, if we make decision with cut-off posterior probability value $1 - \tau$, we need to fit the τ quantile regression function of the response. Once the model is fitted, i.e., the parameter vector $\boldsymbol{\beta}$ is estimated as $\mathbf{b}$, we can make prediction by

$$\hat{Y} = I(f(\mathbf{x}, \mathbf{b}) \geqslant 0),$$

where $\hat{Y}$ is the predicted class label for the predictor vector $\mathbf{x}$.

2.2.2. Quantile boost for binary classification

To fit the model for the τth quantile of Y, i.e., to estimate the parameter vector $\boldsymbol{\beta}$ in Eq. (6), in a way similar to the method

applied to QBR, the estimated parameter vector $\mathbf{b}$ can be obtained by solving:

$$\mathbf{b} = \arg\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \rho_\tau [Y_i - I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0)] \right\}. \tag{10}$$

In Appendix A, it is proved that the above minimization problem is equivalent to a maximization problem:

$$\mathbf{b} = \arg\max_{\boldsymbol{\beta}} \left\{ S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [Y_i - (1-\tau)] I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) \right\}. \tag{11}$$

However, the function $S(\boldsymbol{\beta})$ is not differentiable, because of the use of the indicator function $I(f(\mathbf{x}, \boldsymbol{\beta}) \geqslant 0)$. To apply gradient based optimization methods, we replace $I(f(\mathbf{x}, \boldsymbol{\beta}) \geqslant 0)$ by its smoothed version, and solve

$$\mathbf{b} = \arg\max_{\boldsymbol{\beta}} \left\{ S(\boldsymbol{\beta}, h) = \sum_{i=1}^{n} [Y_i - (1-\tau)] K\left(\frac{f(\mathbf{x}_i, \boldsymbol{\beta})}{h}\right) \right\}, \tag{12}$$

where h is a small positive value, and $K(t)$ is smoothed version of the indicator function, $I(t \geqslant 0)$, with the following properties (Kordas, 2006):

$$K(t) > 0, \quad \forall t \in \mathbf{R}, \quad \lim_{t\to\infty} K(t) = 1, \quad \lim_{t\to-\infty} K(t) = 0.$$

In this paper, we take $K(\cdot)$ as the standard normal cumulative distribution function

$$\Phi(z) = \int_{-\infty}^{z} \phi(t)dt \quad \text{with} \quad \phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}. \tag{13}$$

Let each term in the objective function (12) be

$$l(Y, f) = [Y - (1-\tau)] K(f/h).$$

Following the general steps of functional gradient ascent, we obtain the Quantile Boost Classification (QBC) algorithm as shown in Fig. 3.

2.2.3. Relationship between Eq. (11) and error rate

Maximizing Eq. (11) has a close relation to the performance of the final classifier $I(f(\mathbf{x}, \boldsymbol{\beta}) \geqslant 0)$. Let each term in Eq. (11) be

$$\begin{aligned} S_i = [Y_i - (1-\tau)] I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) &= \tau I(Y_i = 1) I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) \\ &\quad - (1-\tau) I(Y_i = 0) I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0), \end{aligned} \tag{14}$$

then Eq. (11) becomes

$$\begin{aligned} S(\boldsymbol{\beta}) &= \tau \sum_{i=1}^{n} I(Y_i = 1) I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) \\ &\quad - (1-\tau) \sum_{i=1}^{n} I(Y_i = 0) I(f(\mathbf{x}_i, \beta) \geqslant 0) \\ &= \tau TP - (1-\tau) FP = \tau(Pos - FN) - (1-\tau) FP \\ &= \tau Pos - [\tau FN + (1-\tau) FP], \end{aligned} \tag{15}$$

where TP, FP, and FN are the true positive, false positive, and false negative numbers of the classifier $I(f(\mathbf{x}, \boldsymbol{\beta}) \geqslant 0)$, respectively; Pos is the number of positive training examples.

Eq. (15) implies that maximizing Eq. (11) is equivalent to minimizing a convex combination of FP and FN, $\tau FN + (1-\tau) FP$. If $\tau = 0.5$, this is equivalent to minimizing the training error number, whereas if $\tau > 0.5$, the minimization puts more weight on FN, while FP is more emphasized if $\tau < 0.5$. This analysis is consistent with Eq. (9): if $\tau > 0.5$, the classifier makes decision at posterior cut-off probability $1 - \tau$ which is less than 0.5, and this will make FN small; similarly, for $\tau < 0.5$, the posterior cut-off probability will be greater than 0.5, which will lower the FP.

2.2.4. Related works

Kordas et al. (2002, 2006) proposed binary quantile regression for the purpose of classification using quantiles. However, in binary QReg, the simulated annealing algorithm is employed to perform the optimization task. Although a local optimum is guaranteed, simulated annealing is well known for its expensive computation, and it is usually difficult to tell when the algorithm converges. While QBC is based on gradient ascent, it yields a local maximum and converges fast. Due to the expensive computation of simulated annealing, binary QReg can only work in very low dimensional spaces. However, in application, we frequently face hundreds or even thousands of predictors, and it is often desired to find out the informative predictors. Clearly, in this case, binary QReg is not applicable. On the contrary, QBC is designed to work in high

0. Given training data $\{(\mathbf{x}_i, Y_i), i = 1, \cdots, n\}$ with $Y_i \in \{0,1\}$, the desired quantile value τ, and the total number of iterations M, initialize $f^{[0]}(\cdot)$ with $f^{[0]}(\cdot) = 0$.
1. **for** $m = 1$ to M **do**
2. Compute the gradient $\frac{\partial}{\partial f} l(Y, f)$ and evaluate at $f^{[m-1]}(\mathbf{x}_i)$:

$$U_i = \frac{Y_i - (1-\tau)}{h} K'\left(\frac{f^{[m-1]}(\mathbf{x}_i)}{h}\right), \quad i = 1, \cdots, n.$$

3. Fit the gradient vector $U_1, \cdots, U_n$ to $\mathbf{x}_1, \cdots, \mathbf{x}_n$ by the base procedure:

$$\{(\mathbf{x}_i, U_i), i = 1, \cdots, n\} \to g^{[m]}(\cdot).$$

4. Update the estimation by $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \eta_m g^{[m]}(\cdot)$, where η_m is a step size factor.
5. **end for**
6. Output the obtained classifier $I(f^{[M]}(\mathbf{x}) \geq 0)$.

Fig. 3. Quantile boost classification (QBC) algorithm.

dimensional spaces, and it enables us to select the most informative variables by using certain types of base learner.

Hall, Titterington, and Xue (2009) proposed a median based classifier which works in high dimensional space. For a given predictor vector, (Hall et al., 2009) compares the L_1 distances from the new predictor vector to the component-wise median vectors of the positive and negative examples in the training set, and assigns class label as the class with the smaller L_1 distance. Although computationally efficient, this simple nearest-neighbor-like algorithm cannot perform variable selection as the proposed QBC algorithm. Mease, Wyner, and Buja (2007) combines AdaBoost and sampling techniques to predict the class quantiles, but the sampling technique might cause under utilization of the available information. The method proposed in Langford et al. (2006) can also be applied to classification tasks, however it is computationally more expensive than QBC, and it is not clear how to select most informative predictors as well.

3. Calculation

In the proposed QBR and QBC algorithms (as shown in Figs. 2 and 3, respectively), in a manner similar to the work in Friedman (2001), let the base procedure be $h(\mathbf{x}, \mathbf{a})$ with $\mathbf{a}$ being a parameter vector. The third step of QBR and QBC can be performed by an ordinary least square regression:

$$\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} [U_i - h(\mathbf{x}_i, \mathbf{a})]^2,$$

hence the function $g^{[m]}(\mathbf{x}) = h(\mathbf{x}, \mathbf{a}_m)$ may be regarded as an approximation to the negative gradient by the base procedure for QBR, and in the QBC algorithm, the function $g^{[m]}(\mathbf{x}) = h(\mathbf{x}, \mathbf{a}_m)$ is an approximation to the gradient.

In step 4 of QBR, the step size factor can be determined by a line search algorithm (e.g., Fibonacci search or Golden section search (Walsh, 1975, chap. 3)) as:

$$\eta_m = \arg\min_{\gamma} \sum_{i=1}^{n} \rho_\tau [Y_i - f^{[m-1]}(\mathbf{x}_i) - \gamma g^{[m]}(\mathbf{x}_i)].$$

For QBC, the line search algorithm for step size is

$$\eta_m = \arg\max_{\gamma} \sum_{i=1}^{n} [Y_i - (1-\tau)] K\left(\frac{f^{[m-1]}(\mathbf{x}_i) + \gamma g^{[m]}(\mathbf{x}_i)}{h}\right).$$

Alternatively, for simplicity, in each iteration, we could update the fitted function $f^{[m-1]}(\cdot)$ by a fixed, but small, step in the negative gradient direction for QBR and in the gradient direction for QBC. It is verified in Bühlmann and Hothorn (2007) that the size of η_m is less important as long as it is sufficiently small. A smaller value of fixed step size η_m typically requires a larger number of boosting iterations and thus more computing time, while the predictive accuracy has been empirically found to be good when choosing η_m "sufficiently small", e.g., $\eta_m = 0.1$ (Friedman, 2001). To balance the predictive accuracy and computational burden, Bühlmann and Hothorn (2007) suggested a step size of 0.1 be chosen. By simulations, (Friedman, 2001) showed that the performance is similar, if the step size parameter is less than 0.125. Thus, in our experiments, the step size parameter of QBR and QBC is fixed at $\eta_m = 0.1$ for all m, as suggested by Bühlmann and Hothorn (2007), Friedman (2001).

In our implementation of the QBC algorithm, the standard normal cumulative distribution function given in Eq. (13) was used as an approximation of the indicator function with $h = 0.1$. From our experience, QBC is not sensitive to the value of h as long as $h < 0.5$.

The proposed QBR and QBC algorithms, the median classifier in Hall et al. (2009), and the AdaBoost face detector (Viola & Jones, 2001) were implemented using the programming language MATLAB®; all other algorithms including the original quantile regression, L_1 norm quantile regression (Li & Zhu, 2008; Wu & Liu, 2009), AdaBoost (Freund & Schapire, 1997), LogitBoost (Friedman et al., 2000), and L_2-Boost (Bühlmann & Yu, 2003) were implemented in statistics programming language R. All the experiments were conducted on an ordinary PC with 3.0 GHz CPU and 3.25 GB memory.

4. Results and discussion

We applied the proposed QBR and QBC algorithms on extensive regression and pattern classification problems, and this section will present the experimental results with in-depth discussions.

4.1. Experiments on regression problems

This subsection tests the proposed QBR algorithm on various regression problems with benchmark datasets in Section 4.1.1, and compares QBR to L_1 norm quantile regression (Li & Zhu, 2008; Wu & Liu, 2009) on simulated dataset in Section 4.1.2.

4.1.1. Results of QBR on benchmark datasets

Since the purpose of this paper is to propose an alternative algorithm to the original QReg, we choose only to compare the performance of QBR to that of the original QReg. Five regression datasets from UCI machine learning repository were used in our experiment: Red Wine Quality, White Wine Quality, Forest Fire, Concrete Slump, and Concrete Compressive; see Table 1 for the basic information of the datasets. The predictor and the response variables were normalized to be in $[-1, 1]$. To make a fair comparison between QBR and QReg, we used simple linear regression with only one predictor as base procedure in QBR.

In evaluation, we use the sum of the "check losses" on the testing set as the error measure which is defined by

$$L(\tau) = \sum_{i=1}^{N_{\text{test}}} \rho_\tau (Y_i - \hat{f}(\mathbf{x}_i)),$$

where N_{test} is the size of the testing set, and $\hat{f}(\mathbf{x}_i)$ is the predicted τth quantile at $\mathbf{x}_i$ by the original QReg or QBR. By the definition of quantile, a smaller value of $L(\tau)$ gives an estimated quantile closer to the true value.

For each dataset, we randomly select 80% of the examples as training set and use the remaining 20% as the testing set. For three τ values (0.25, 0.5, and 0.75), QBR and QReg are separately trained and evaluated on these subsets. The partition-training-testing procedure is repeated 500 times. The means and the standard deviations of the 500 "check losses" are reported in Table 2. It is readily observed that in all of the conducted experiments, QBR uniformly achieves smaller average check loss compared to QReg, which indicates that QBR estimates more accurately.

To statistically compare the results, we conduct a z-test with

$$H_0: L_{\text{QBR}} - L_{\text{QReg}} \geqslant 0 \quad \text{vs.} \quad H_1: L_{\text{BQR}} - L_{\text{QReg}} < 0, \tag{16}$$

where L_{QBR} and L_{QReg} are the check losses of QBR and QReg, respectively. By a standard z-test procedure (please refer to some elementary statistics textbooks, e.g., Navidi, 2010, chap. 6), we obtain the p-values for each pair of comparison, as listed in Table 3. We note that on 11 out of 15 comparative experiments, the p-values are less than 2%, which indicates that in most cases, QBR performs *significantly* better than QReg. The reason might be that in QBR, we implicitly regularize the regression coefficients by applying a small learning rate, i.e., using small step size η_m.

In applications, variable selection is often needed, since it helps us get a simpler model, making the model easier to interpret/understand and identifying informative variables. Although the relevant variables are usually unknown, we none the less prefer a simpler model which is capable of deleting more variables: Furthermore, a simpler model is more time efficient in making

Table 1
Basic information about the five datasets used for regression problems.

Dataset	Red wine	White wine	Forest fire	Concrete slump	Concrete comp.
Size	1599	4898	517	103	1030
Dim.	11	11	12	7	9

Table 2
The comparison between QReg and QBR on the five datasets from UCI machine learning repository. The values listed are the mean values of the check loss of the 500 runs and the standard deviations are listed in parentheses.

Dataset	$\tau = 0.25$		$\tau = 0.50$		$\tau = 0.75$	
	QReg	QBR	QReg	QBR	QReg	QBR
Red wine	19.71 (1.11)	19.41 (1.11)	24.30 (1.06)	24.07 (1.09)	19.40 (0.81)	19.16 (0.79)
White wine	62.40 (1.84)	62.23 (1.91)	78.69 (1.78)	78.60 (1.92)	62.03 (1.50)	61.78 (1.64)
Forest Fire	4.97 (0.54)	4.93 (0.54)	10.04 (0.82)	9.62 (0.78)	9.56 (0.92)	9.14 (0.73)
Conc. slump	0.74 (0.14)	0.71 (0.12)	1.00 (0.17)	0.92 (0.16)	0.95 (0.19)	0.84 (0.17)
Conc. comp.	15.02 (0.77)	14.91 (0.79)	18.21 (1.08)	18.17 (1.00)	13.63 (0.83)	13.52 (0.77)

Table 3
The p-values for the hypothesis testing problem in Eq. (16) on the five datasets for each quantile. 0+ indicates the value is less than 0.0001.

Dataset	Red wine	White wine	Forest fire	Concrete slump	Concrete comp.
$\tau = 0.25$	0+	0.0769	0.1132	0+	0.0158
$\tau = 0.50$	0+	0.2054	0+	0+	0.2596
$\tau = 0.50$	0+	0.0044	0+	0+	0.0188

Table 4
Comparison between QReg and QBR on variable selection on concrete slump data. The experiment was repeated 500 times. The listed are the average number of selected noisy variables (N.V.), the mean of check loss error with the standard deviation in parentheses.

Method	$\tau = 0.25$		$\tau = 0.50$		$\tau = 0.75$	
	# of N.V.	Error	# of N.V.	Error	# of N.V.	Error
QReg	8.13	0.99 (0.16)	6.97	1.38 (0.21)	9.69	1.28 (0.24)
QBR	1.04	0.97 (0.19)	0.63	1.14 (0.21)	0.98	1.27 (0.30)

prediction. In our experiments, both QReg and QBR obtain a linear function of the predictors. Since the predictors and response are normalized to be in $[-1, 1]$, it makes sense if we delete the variables, when their coefficients are too small, thus performing variable selection. Twenty noisy predictors are added to the Concrete Slump data, each of which is generated uniformly at random from $[-1, 1]$. Thus, in the noisy dataset, only a small fraction (7 out of 27) of the variables are informative. Then we repeat the above experiment. In the linear model obtained, we calculate the sum of the absolute values of all the coefficients. If for any predictor, the absolute value of its estimated coefficient is less than 1% of the total sum, it is trimmed. We calculate the average number of the selected noisy predictors of QReg and QBR from the 500 runs. The means and standard deviations of the check losses on testing set are also calculated. Table 4 summarizes the results, from which it is evident that for each τ value, compared to the original QReg, QBR selects far fewer noisy predictors while achieving smaller mean error. This experiment shows that compared to QReg, QBR is more robust to noisy predictors.

The procedure of QBR enables us to use other forms of base learner, for example, regression trees (Friedman, 2001; Friedman et al., 2000), and this provides us flexibility in certain applications. Contrarily, it is not clear how to make use of other forms of regression in the framework of the original QReg.

4.1.2. Comparing to L_1 norm quantile regression

QBR belongs to the stage-wise additive model, which has a close connection to L_1 constrained models (Efron, Hastie, Johnstone, & Tibshirani, 2004). Recently, L_1 quantile regression (L_1-QReg) (Li & Zhu, 2008; Wu & Liu, 2009) was proposed, which imposes L_1 constraint on the coefficient vector in linear quantile regression. Therefore, it is natural to investigate the relationship between QBR and L_1-QReg.

Given data $\{(\mathbf{x}_i, Y_i),\ i = 1, \ldots, n\}$, assuming the use of the linear model $\beta_0 + \boldsymbol{\beta}^T\mathbf{x}$ for estimating the τth quantile of the response, the L_1-QReg can be formulated as

$$\min_{\beta_0, \boldsymbol{\beta}} \quad \sum_{i=1}^{n} \rho_\tau(Y_i - \beta_0 - \boldsymbol{\beta}^T\mathbf{x}_i),$$

$$\text{subject to} \quad |\beta_1| + \cdots + |\beta_p| \leqslant s.$$

The solution of L_1-QReg depends on the penalty parameter s, and changing the parameter s gives us the evolution of the solution, which is called the solution path (Efron et al., 2004; Li & Zhu, 2008). Similarly, in the QBR algorithm with simple linear regression as base procedure, we can observe the evolution of the regression coefficients with the iteration number.

The relationship between L_1-QReg and QBR was investigated by experimentation with simulated data generated from the model

$$Y = \boldsymbol{\beta}^T\mathbf{x} + \varepsilon,$$

where $\boldsymbol{\beta} = (3, -1.5, 0, 0, 2, 0, 0, 0)^T$, the error term ε follows standard double exponential distribution, and $\mathbf{x} \in \mathbf{R}^8$ with each component $x_i \sim N(0, 1)$ and $x_1, \ldots, x_8$ are independent. Five hundred training examples and 10,000 testing observations were generated from the same model. Define the closeness measure between the estimated model and the true model as $\sum_{i=1}^{8}|\beta_i - \hat{\beta}_i|$, where $\hat{\beta}_1, \cdots, \hat{\beta}_8$ are the estimated regression coefficients.

Fig. 4 shows the solution paths and the check losses of the L_1-QReg and QBR using simple linear regression as the base procedure. Firstly, we observe that both L_1-QReg and QBR select the informative predictors, i.e., x_1, x_2, and x_5, and all other predictors have coefficients very close to zero. The final coefficients estimated by L_1-QReg and QBR are listed in Table 5. The two estimates are clearly very similar with the QBR solution slightly closer to the true model. The closeness measures are 0.3813 for L_1-QReg and 0.3184 for QBR, respectively. Secondly, it is observed that the shape of the solution paths are strikingly similar, which means that each of the solutions of L_1-QReg corresponds to a solution of QBR with certain iterations. Thirdly, the average check loss curves show that the trends in the check loss are very similar. The final average check losses are 0.5142 for L_1-QReg and 0.5132 for QBR.

This comparison demonstrates that the L_1-QReg and the proposed QBR have similar properties and similar performances in variable selection and prediction accuracy. Despite the similarity when QBR employs simple linear regression as the base learner, as previously mentioned, QBR enjoys the flexibility of being able to use other forms of base learners, while the L_1-QReg in Li and Zhu (2008), Wu and Liu (2009) can only be used for linear model.

4.2. Experiments on binary classification problems

In this subsection, three experiments were conducted to study the performance of the proposed QBC algorithm. In the first two experiments, we made decision at the cut-off posterior probability 0.5, and therefore we fit QBC with $\tau = 0.5$. Since QBC is a boosting

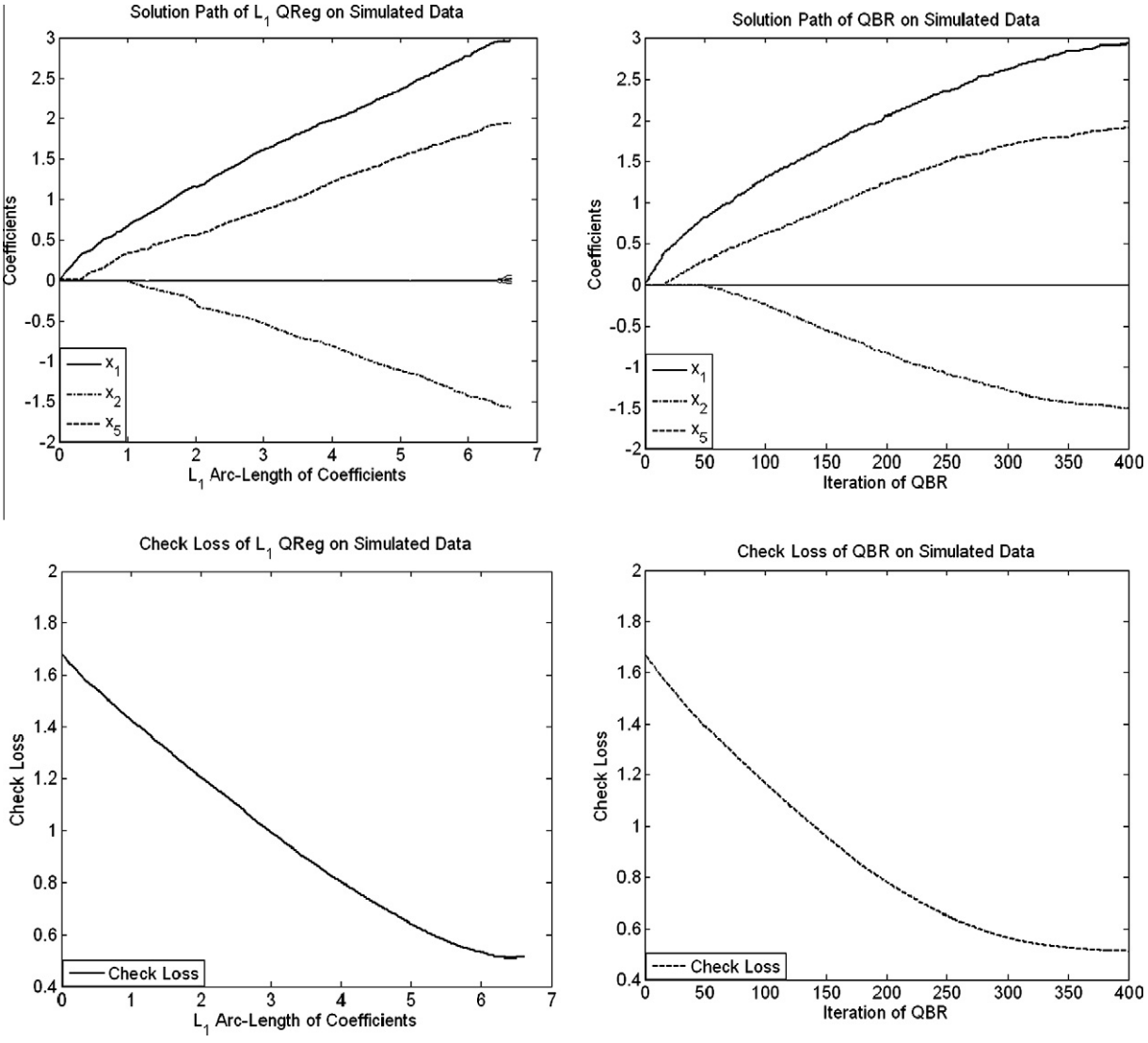

Fig. 4. Solution paths and the average check losses on testing data of L_1-QReg and QBR. In the plots for L_1-QReg, the x-axis is the sum of the absolute values of the solution, i.e., $\sum_{i=1}^{8}|\hat{\beta}_i|$.

Table 5
The final estimated regression coefficients by L_1-QReg and QBR.

Coefficients	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	$\hat{\beta}_4$	$\hat{\beta}_5$	$\hat{\beta}_6$	$\hat{\beta}_7$	$\hat{\beta}_8$
L_1-QReg	2.9633	−1.5773	0.0611	−0.0403	1.9287	0.0314	−0.0119	0.0147
QBR	2.9619	−1.5749	0.0516	−0.0283	1.9318	0.0189	−0.0004	0.0142

procedure, for the comparative purpose, we, for the most part, considered only several of the most popular boosting based classifiers. These include AdaBoost (Freund & Schapire, 1997), LogitBoost (Friedman et al., 2000), and L_2-Boost (Bühlmann & Yu, 2003). We also tested the Median Classifier (Hall et al., 2009) and compared the performance. The third experiment compared the performance of AdaBoost to those of QBC, with $\tau = 0.5$ and $\tau = 0.25$ on face detection problem. The role of τ in QBC algorithm was also illustrated.

4.2.1. Results on credit score data

We compare the result of QBC to that of the binary QReg (Kordas et al., 2002) on the German bank credit score dataset which is available from UCI machine learning repository. The dataset is of size 1000 with 300 positive examples, each has 20 predictors normalized to be in $[-1, 1]$. In Kordas et al. (2002), only 8 predictors were preselected to fit the binary QReg due to the expensive simulated annealing algorithm it employs in the optimization stage.

Without preselecting variables, we randomly selected a training set of size 800 from the dataset and evaluated the learned QBC classifier on the other 200 examples. The QBC training algorithm was ran for 100 iterations using simple linear regression with only one predictor as base learner. It is therefore fair to compare the performance of QBC to that of the binary QReg. The splitting-training-testing process was repeated 500 times, and the mean training and testing error rates were reported in Table 6, which also lists the performance of the binary QReg from Kordas et al. (2002). It is clear that QBC outperforms binary QReg on both training and testing sets. This is due to the efficient computation of QBC, which allows the algorithm to explore more predictors and thus select more informative ones.

Fig. 5 shows the training error and the average objective function (see Eq. (12)) as the QBC training algorithm proceeds. As we can see, the objective function increases monotonically and meanwhile, the training error decreases. This is consistent with our theoretical analysis in Section 2.2.3. We can also see that the training algorithm converges with about 50 iterations.

Table 6
Comparison between binary QReg and QBC on credit score dataset. "Clean" means the original dataset, and "Noisy" means 20 noisy predictors are added to the dataset.

Dataset	QBC		Binary QReg	
	Training (%)	Testing (%)	Training (%)	Testing (%)
Clean	19.84	24.47	21.9	26.5
Noisy	22.53	25.64	NA	NA

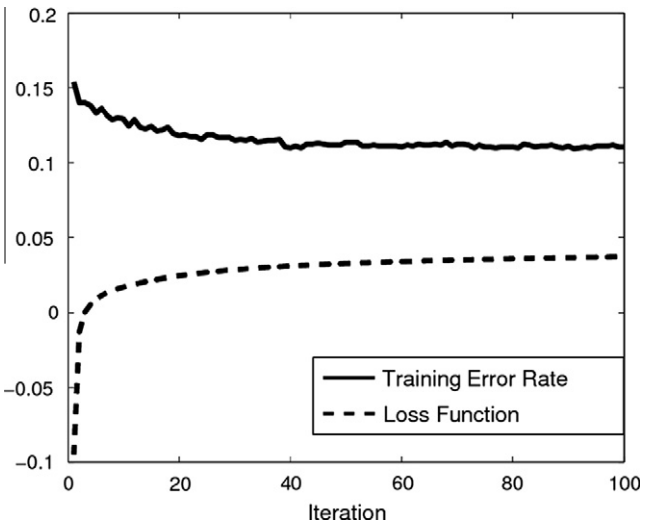

Fig. 5. The average objective function and the training error for a particular run of QBC on the credit score dataset. The training error is scaled by 0.5 to make the curves comparable.

Table 7
The testing errors of different algorithms on the credit score dataset.

Data	L_2Boost (%)	AdaBoost (%)	LogitBoost (%)	QBC (%)	Median classifier (%)
Clean	28.68	29.99	28.81	**28.50**	35.04
Noisy	31.92	30.05	32.68	**28.55**	38.94

To test the variable selection ability of QBC, twenty noisy predictors were generated from the uniform distribution on $[-1, 1]$ and were added to the original dataset. The above procedure was repeated 500 times. Table 6 also lists the average training and testing errors of QBC with, on average, only one noisy variable selected. Our result demonstrates that QBC performs only slightly worse on noisy data, indicating that QBC is robust to noise. Due to the expensive computation of binary QReg, its performance on noisy data is not provided.

Unlike binary QReg, QBC enjoys the flexibility of using other forms of weak learners. We also compared QBC to the alternatives mentioned at the beginning of Section 4.2 on the credit score dataset with and without noisy predictors. All the boosting based algorithms used the decision stump (Dettling & Bühlmann, 2003; Viola & Jones, 2001) as the base procedure for fair comparison, and all algorithms were ran 500 times on sets of 800 training and 200 testing examples, all randomly selected. The average testing error rates are listed in Table 7, and show that compared to the alternative methods, QBC achieves the best performance on both clean and noisy data. Again, we observe that QBC deteriorates only slightly on the noisy data, verifying its robustness to noisy predictors.

4.2.2. Results on gene expression data

We compare QBC to the alternative methods on three publicly available datasets in bioinformatics (Dettling & Bühlmann, 2003): Estrogen, Nodal, and Colon. In these datasets, each predictor is the measure of the gene expression level, and the response is a binary variable. Although we can access thousands of genes, due to the high cost of experiment, we usually can only get a very limited number (50–100) of examples. See Table 8 for the sizes and dimensions of these datasets. The high dimensionality makes binary QReg in Kordas et al. (2002, 2006) unaffordable. All the boosting-based algorithms used the decision stump as base learner for fair comparison.

Since all the datasets have small size n, the leave-one-out (LOO) cross validation is carried out to estimate the classification accuracy. That is, we put aside the ith observation and trained the classifier on the remaining $(n-1)$ data points. We then applied the learned classifier to get $\hat{Y}_i$, the predicted class label of the ith observation. This procedure is repeated for all the n observations in the dataset, so that each one is held out and predicted exactly once. The LOO error number (N_{LOO}) and rate ($\mathrm{E}_{\mathrm{LOO}}$) are determined by

$$N_{\mathrm{LOO}} = \sum_{i=1}^{n} I(\hat{Y}_i \neq Y_i) \quad \text{and} \quad E_{\mathrm{LOO}} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{Y}_i \neq Y_i).$$

Table 8 summarizes the LOO classification error numbers and rates of the considered classifiers on the three datasets, with the best performance marked in bold. From Table 8, it is readily seen that QBC has the best LOO performance. It can also be observed that the Median Classifier (Hall et al., 2009) is not stable – sometimes the performance is very good, sometimes the performance is very poor.

In LOO cross validation, the training and testing sets are highly unbalanced, which will affect the evaluation result. To provide more thorough results, we have also conducted 5-fold cross validation (5-CV), in which each dataset was randomly partitioned into

Table 8
The leave-one-out error numbers and rates of the considered algorithms on the three gene expression datasets. For each algorithm, the misclassification numbers are provided and the error rates are listed in parentheses. The best performances are displayed in **bold**. The size and dimension of each dataset are also given.

Dataset	Size	Dim	L_2-Boost	AdaBoost	LogitBoost	QBC	Median classifier
Estrogen	49	7129	**4 (8.16%)**	5 (10.20%)	6 (12.24%)	**4 (8.16%)**	6 (12.24%)
Colon	62	2000	10 (16.13%)	11 (17.74%)	10 (16.13%)	**9 (14.52%)**	**9 (14.52%)**
Nodal	49	7129	11 (22.45%)	12 (24.49%)	9 (18.37%)	**8 (16.33%)**	18 (36.73%)

Table 9
The 5-fold cross validation mean error rates and the standard deviations (in parentheses) of the considered algorithms on the three gene expression datasets. The best mean error rates for a dataset are displayed in **bold**.

Dataset	L_2-Boost (%)	AdaBoost (%)	LogitBoost (%)	QBC (%)	Median classifier (%)
Estrogen	21.11 (10.91)	17.11 (10.50)	15.11 (12.87)	**13.33** (9.30)	13.38 (11.04)
Colon	24.62 (5.95)	21.41 (7.68)	21.41 (7.68)	21.67 (9.50)	**14.58** (9.20)
Nodal	31.78 (17.67)	24.89 (8.52)	20.44 (13.67)	**20.00** (9.30)	42.84 (15.73)

five parts of roughly equal size. In every experiment, one part was used as the testing set, and the other four parts were used as the training set. Table 9 lists the mean and the standard deviation of the error rates for the five runs of each algorithm on every dataset. We observe from Table 9 that QBC yields the best performance on two out of the three datasets. On the Colon dataset, QBC performs better than L_2-Boost, similar to LogitBoost and AdaBoost, but worse than the Median Classifier. However, due to the instability of the Median Classifier, we can still safely conclude that QBC performs better than or similar to several of the alternatives.

4.2.3. Face detection

Face detection is a classical problem in computer vision, and many machine learning algorithms have been successfully applied to face detection, e.g., Support Vector Machines (Osuna, Freund, & Girosit, 1997), and AdaBoost (Viola & Jones, 2001). We applied the proposed QBC algorithm to a dataset with 3000 face and 3000 nonface images, each of size 16×16. The training set includes 1500 face and 1500 nonface images randomly selected from the whole dataset and the rest of the images were used for testing. Emulating (Viola & Jones, 2001), we extracted different types of Haar features from the images, taking a minimum rectangular size of 4 by 4 (for example: 4 by 5, 8 by 4, 10 by 11), resulting in a total of around 6000 features. The AdaBoost face detector follows (Viola & Jones, 2001), and QBC employs regression stump (Torralba, Murphy, & Freeman, 2004) as the weak learner since it is similar to the weak learner used in Viola and Jones (2001).

We first compare the classification error rates of AdaBoost to that of QBC with $\tau = 0.5$ and show the testing error curves in Fig. 6. It is observed that AdaBoost slightly outperforms QBC with $\tau = 0.5$. To have a closer look, we plot out the curves of false positive rates and false negative rates for the two methods in Fig. 7. It is clear that after 40 iterations, AdaBoost and QBC (with $\tau = 0.5$) have similar false positive rates, but AdaBoost has a smaller false negative rate, and this is the reason why AdaBoost has a slightly better performance in total error rate.

We further notice in Fig. 7 that the false positive rates are much higher than the false negative rates. Therefore, on this dataset, in order to improve the classification accuracy of QBC, we have to decrease the false positive rate. According to the theoretical analysis in Section 2.2.3, if $\tau < 0.5$, the algorithm will pay more attention to decreasing the false positive rate. As such, we fit another QBC classifier with $\tau = 0.25$, and Fig. 7 also shows the corresponding false positive and false negative rates. It is observed that although the false negative rate of QBC with $\tau = 0.25$ is a little bit higher than those of QBC with $\tau = 0.5$ and AdaBoost, the false positive rate is significantly lower than those of QBC with $\tau = 0.5$ and AdaBoost. As a consequence, the overall testing error rate is significantly dropped, as shown in Fig. 6. We further notice that the performance curves of AdaBoost are much more erratic than those of QBC, which implies that the performance of AdaBoost is less stable.

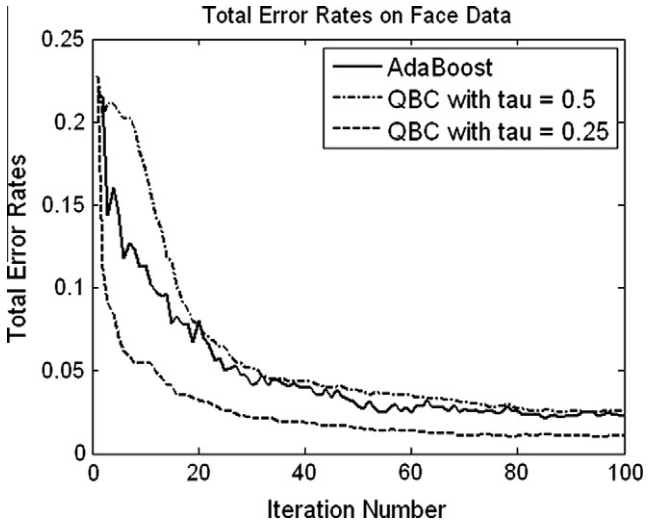

Fig. 6. On the face detection problem, the testing error rates of AdaBoost, QBC with $\tau = 0.5$, and QBC with $\tau = 0.25$.

There are modifications to AdaBoost given in literature for balancing *FP* and *FN*, e.g., in Masnadi-Shirazi and Vasconcelos (2007) and Sun, Kamel, Wong, and Wang (2007). This experiment was intended to demonstrate the role of τ in the QBC algorithm, thus we choose not to compare QBC with different τ to different versions of AdaBoost for imbalanced penalties.

As a comparison to AdaBoost and QBC, the median based classifier reported in Hall et al. (2009) was also tested on the face detection problem. The test was repeated 20 times, since this algorithm is time efficient. In each repetition, we randomly split the whole dataset so that both the training set and testing set include 1500 face and 1500 nonface images. The mean error rate of the 20 tests is 14.42%, the mean false positive rate is 21.78%, and the mean false negative rate is 7.07%. Comparing these numbers to Figs. 6 and 7, we see that the performance of the Median Classifier in Hall et al. (2009) is much worse than those of AdaBoost and QBC.

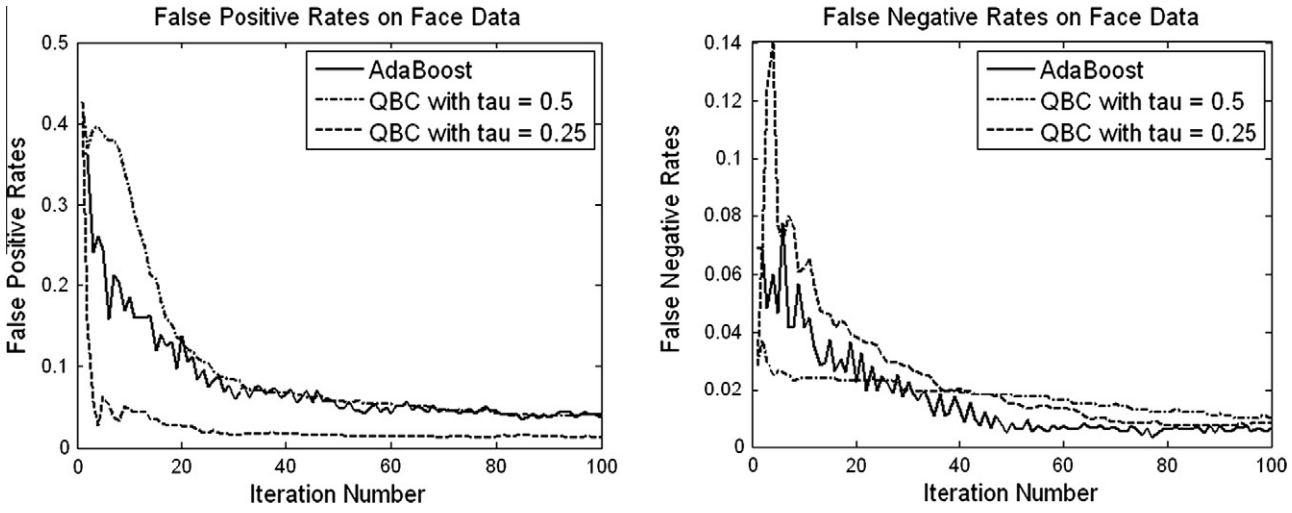

Fig. 7. On the face testing set, false positive and false negative rates of AdaBoost, QBC with $\tau = 0.5$, and QBC with $\tau = 0.25$. Note that the y-axis have different scales in the two figures.

5. Conclusions

Motivated by the gradient boosting framework, this paper applies functional gradient descent to fit quantile regression (QReg), resulting Quantile Boost Regression (QBR) algorithm. In binary classification problems, the class label is defined via a hidden variable, and predicting the quantiles of the binary response is reduced to predicting the corresponding quantiles of the hidden variable. An equivalent form of the definition of quantile was introduced, and its smoothed version was further employed as the objective function. Functional gradient ascent was applied to maximize the objective function for fitting the classifier, yielding the Quantile Boost Classification (QBC) algorithm. The proposed Quantile Boost (QBoost) algorithms yield local optima, and more importantly, they enable us to solve problems in high dimensional spaces and to select informative variables/features.

The QBoost algorithms were tested extensively on various regression and binary classification problems with benchmark datasets. On most of the regression experiments, QBR performs significantly better than the original QReg, in terms of check loss function. Moreover, the comparative experiment on noisy data indicates that QBR is more robust to noise. The comparative result on a credit score dataset shows that QBC outperforms binary QReg (Kordas et al., 2002) and is more robust to noisy predictors. The experiments on credit score dataset and gene expression data analysis demonstrate that QBC performs better than or similar to other alternatives in terms of leave-one-out and five fold cross validation error rates. QBC was compared to the popular AdaBoost classifier (Viola & Jones, 2001) on face detection problem and we demonstrated the role of the desired τ value in QBC algorithm. On face detection and gene expression data analysis, binary QReg (Kordas et al., 2002) is not applicable due to the high dimensionality of the datasets.

The current version of QBC is for two-class problems, and we plan to develop the multi-class version of QBC by reducing it to a series of two-class tasks, for example, by one-versus-all (Freund and Schapire, 1997) or Error-Correcting Output Codes (Dietterich & Bakiri, 1995). A more thorough comparison of the proposed QBR/QBC to other quantile based regression/classification algorithms on more datasets is also desired.

Acknowledgments

The author gratefully acknowledges the distinction and support provided by *Sociedad Mexicana de Inteligencia Artificial* (*SMIA*) and the *9th Mexican International Conference on Artificial Intelligence* (*MICAI-2010*) in order to enhance, improve, and publish this work. The author also thanks the reviewers of MICAI and *Journal of Expert Systems with Applications* for their constructive comments and would like to extend his gratitude to Prof. Alejandro Peña-Ayala for his excellent work in coordinating the preparation, the reviewing, and the revising of the manuscript. Prof. Shelby J. Kilmer provided a great deal of help with the English in this manuscript. This work was partially supported by 2011 Summer Faculty Fellowship of Missouri State University.

Appendix A. The equivalence between Eqs. (10) and (11)

In Eq. (10), let

$$L_i = \rho_\tau(Y_i - I(f(x_i, \boldsymbol{\beta}) \geqslant 0)) = \begin{cases} \rho_\tau(Y_i - 1), & \text{if } f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0 \\ \rho_\tau(Y_i), & \text{if } f(\mathbf{x}_i, \boldsymbol{\beta}) < 0 \end{cases},$$

while

$$\rho_\tau(Y_i) = \tau I(Y_i = 1) \quad \text{and} \quad \rho_\tau(Y_i - 1) = (1 - \tau) I(Y_i = 0).$$

Thus,

$$\begin{aligned} L_i &= (1-\tau) I(Y_i = 0) I(f(\mathbf{x}_i, \beta) \geqslant 0) + \tau I(Y_i = 1) I(f(\mathbf{x}_i, \beta) < 0) \\ &= (1-\tau) I(Y_i = 0) I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) + \tau I(Y_i = 1)[1 - I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0)] \\ &= -\tau I(Y_i = 1) I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) + (1-\tau) I(Y_i = 0) I(f(\mathbf{x}_i, \boldsymbol{\beta}) \geqslant 0) \\ &\quad + \tau I(Y_i = 1) = -S_i + \tau I(Y_i = 1), \end{aligned}$$

where S_i is defined in Eq. (14). Therefore, Eq. (10) reads,

$$\begin{aligned} L(\boldsymbol{\beta}) &= \sum_{i=1}^{n} L_i = \sum_{i=1}^{n} [-S_i + \tau I(Y_i = 1)] = -S(\boldsymbol{\beta}) + \tau \sum_{i=1}^{n} I(Y_i = 1) \\ &= -S(\boldsymbol{\beta}) + \tau Pos, \end{aligned} \tag{A.1}$$

where *Pos* is the number of positive examples, which is a constant. Eq. (A.1) shows the equivalence between Eqs. (10) and (11).

References

Bühlmann, P., & Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science, 22*, 477–505.

Bühlmann, P., & Yu, B. (2003). Boosting with the L_2-loss: Regression and classification. *Journal of the American Statistical Association, 98*, 324–340.

Cade, B. S., & Noon, B. R. (2003). A gentle introduction to quantile regression for ecologists. *Frontiers in Ecology and the Environment, 1*(8), 412–420.

Dettling, M., & Bühlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics, 19*, 1061–1069.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research, 2*, 263–286.
Duffy, N., & Helmbold, D. (2002). Boosting methods for regression. *Machine Learning, 47*, 153–200.
Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics, 32*, 407–499.
Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*, 119–139.
Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*, 1189–1232.
Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics, 28*, 337–407.
Hall, P., Titterington, D. M., & Xue, J. H. (2009). Median-based classifiers for high-dimensional data. *Journal of the American Statistical Association, 104*, 597–1608.
Hendricks, W., & Koenker, R. (1992). Hierarchical spline models for conditional quantiles and the demand for electricity. *Journal of the American Statistical Association, 93*, 58–68.
Hunter, D. R., & Lange, K. (2000). Quantile regression via an MM algorithm. *Journal of Computational and Graphical Statistics, 9*, 60–77.
Koenker, R. (2005). *Quantile regression* (1st ed.). New York: Cambridge University Press.
Koenker, R., & Bassett, G. (1978). Regression quantiles. *Econometrica, 46*, 33–50.
Koenker, R., & Geling, R. (2001). Reappraising medfly longevity: A quantile regression survival analysis. *Journal of the American Statistical Association, 96*, 458–468.
Koenker, R., & Hallock, K. (2001). Quantile regression. *Journal of Economic Perspectives, 15*, 143–156.
Kordas, G. (2002). Credit scoring using binary quantile regression. In Y. Dodge, (Ed.), *Statistical data analysis based on the l_1-norm and related methods*.
Kordas, G. (2006). Smoothed binary regression quantiles. *Journal of Applied Econometrics, 21*, 387–407.
Kriegler, B., & Berk, R. (2007). *Boosting the quantile distribution: A cost-sensitive statistical learning procedure*. Technical Report, Department of Statitics, University of California, Los Angeles.
Langford, J., Oliveira, R., & Zadrozny, B. (2006). Predicting conditional quantiles via reduction to classification. In *Proceedings of uncertainty in artificial intelligence*.
Li, Y., & Zhu, J. (2008). L_1-Norm quantile regression. *Journal of Computational and Graphical Statistics, 17*, 163–185.
Li, Y., Liu, Y., & Zhu, J. (2007). Quantile regression in reproducing kernel Hilbert spaces. *Journal of the American Statistical Association, 102*, 255–268.
Manski, C. F. (1985). Semiparametric analysis of discrete response: Asymptotic properties of the maximum score estimator. *Journal of Economics, 27*, 313–333.
Masnadi-Shirazi, H., & Vasconcelos, N. (2007). Asymmetric boosting. In *Proceedings of international conference on machine learning (ICML)*.
Mason, L., Baxter, J., Bartlett, P. L., & Frean, M. (2000). Boosting algorithms as gradient descent. In *Proceedings of advances in neural information processing systems (NIPS)*.
Mease, D., Wyner, A. J., & Buja, A. (2007). Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research, 8*, 409–439.
Navidi, W. (2010). *Statistics for engineers and scientists* (3rd ed.). New York: McGraw-Hill.
Osuna, E., Freund, R., & Girosit, F. (1997). Training support vector machines: An application to face detection. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*.
Sun, Y., Kamel, M., Wong, A., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition, 40*, 3358–3378.
Takeuchi, I., Le, Q. V., Sears, T. D., & Smola, A. J. (2006). Nonparametric quantile estimation. *Journal of Machine Learning Research, 7*, 1231–1264.
Torralba, A., Murphy, K. P., & Freeman, W. T. (2004). Sharing features: Efficient boosting procedures for multiclass object detection. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*.
Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*.
Walsh, G. R. (1975). *Methods of optimization*. London: John Wiley and Sons.
Wu, Y., & Liu, Y. (2009). Variable selection in quantile regression. *Statistica Sinica, 19*, 801–817.
Zemel, R., & Pitassi, T. (2001). A gradient-based boosting algorithm for regression problems. In *Proceedings of advances in neural information processing systems (NIPS)*.