Component-wise Discrete Asymmetric AdaBoost for High-dimensional Binary Quantile Regression[*]

Jianghao Chu[†] Tae-Hwy Lee[‡] Aman Ullah[§]

Abstract
We generalize Adaptive Boosting or AdaBoost, introduced by Freund and Schapire (1996), to solve the high-dimensional binary quantile regression problems. The existing AdaBoost may be understood as an algorithm to solve the high-dimensional binary median regression problem. We extend the theory of Friedman, Hastie, and Tibshirani (2000) who show that AdaBoost builds an additive logistic regression model via minimizing the 'exponential loss'. Generalizing the exponential loss function to an asymmetric exponential loss function, we introduce "Asymmetric AdaBoost". While the exisitng (symmetric) AdaBoost penalizes false positive (FP) prediction and false negative (FN) prediction equally, we show that the Asymmetric AdaBoost algorithm penalizes them differently (asymmetrically).

Key Words: Adaboost, exponential loss, Asymmetric Adaboost, asymmetric exponential loss, binary quantiles, false negative, false positive.

1. Introduction

Consider the binary variable y that takes a value 1 with probability $\pi(x) \equiv P(y = 1|x)$ and -1 with probability $1 - \pi(x)$. We wish to make predition on y using the high-dimensional predictor x. A traditional econometric model is to estimate $\hat{\pi}(x)$ via logit and probit models, then make the prediction on y being 1 if $\hat{\pi}(x) > 0.5$ using the estimated probability model conditional on a number of predictors (x).

Freund and Schapire (1996) introduce the (symmetric) AdaBoost algorithm which takes a functional descent procedure and selects the predictor variables sequentially in a forward stagewide fashion when x is of high dimension. This method is widely used and performs well in practice. However, it can be shown that it takes 0.5 as cutoff for $\hat{\pi}(x)$ to make the binary prediction. In other words, wrong predictions are given the same penalty regardless of whether it is a false positive (type I error) or false negative (type II error) prediction. Hence we call it a symmetric AdaBoost algorithm.

The goal of this paper is to introduce an asymmetric AdaBoost algorithm, which gives different penalty depending on whether it is a false positive (type I error) or false negative (type II error) prediction. In the boosting literature, algorithms have also been proposed to use asymmetric loss functions, including AdaCost (Fan et al. 1999), CSB0, CSB1, CSB2 (Ting 2000), asymmetric-AdaBoost (Viola and Jones 2002) and AdaC1, AdaC2, AdaC3 (Sun et al. 2005), and asymmetric Boosting (Masnadi-Shirazi and Vasconcelos 2007). Most of the literature only focus on manipulating the weights and/or step-size of the boosting algorithm and lack a sound statistical fundation. These algorithms lost the nice statistical property of AdaBoost and have no theoretical support of its correctness.

[*]We thank seminar participants at USC (Econ), UCR (Econ), UCR (CS), PKU (Business School), CAS (AMSS), CUFE (Economics, Statistics), CMES2018 (Shanghai), AMES2018 (Seoul), and JSM2018 (Vancouver).

[†]Department of Economics, University of California, Riverside, CA 92521. E-mail: jianghao.chu@email.ucr.edu

[‡]Department of Economics, University of California, Riverside, CA 92521. E-mail: tae.lee@ucr.edu

[§]Department of Economics, University of California, Riverside, CA 92521. E-mail: aman.ullah@ucr.edu

We note that the name, "Asymmetric AdaBoost" has been used in Viola and Jones (2002). Although we will use the same name in this paper, it is noted that they are different as theirs use a different methodology for the optimal step size in each iteration. As a result, the first classifier absorbs the entire effect of the asymmetric weights and the remaining rounds are entirely symmetric (Viola and Jones 2002). Moreover, their asymmetry is based merely on a simple belief that the loss of a false positive error is k times the loss of a false negative one and cannot be applied to complex situations in prediction of economic behavior.

This paper is concerned with getting the optimal binary classification/prediction algorithm with a loss function which is asymmetric. The proposed new algorithm produces an additive model by iteratively maximizing an asymmetric "exponential" loss function. In each iteration, we put more weight on the observations that cannot be predicted correctly using the previous predictors, with a particularly designed weight that is consistent with the asymmetric exponential loss based on a sound mathematical derivation. It is therefore able to solve not just the binary median regression but also binary quantile problems with high-dimensional covariates.

The rest of the paper is organized as follow. In Section 2 we provide a brief introduction of the symmetric AdaBoost as minimizing an exponential loss function. When the dependent variable and the forecast function are binary, the exponential loss function gives the same solution as the ordinary symmetric least squares loss function. This equivalence indicates the possibility to use the exponential loss in place of the least square loss especially when the data are high-dimensional. In Section 3 we introduce a new "asymmetric exponential loss" function, based on which we derive a new component-wise discrete asymmetric adaptive boosting algorithm. We also relate the asymmetric exponential loss function with asymmetric L_2 or L_1 loss functions. When the target variable is binary, L_2 and L_1 loss functions are equivalent up to a constant scale. Thus, the Asymmetric AdaBoost will produce the usual binary quantiles of the binary target variable. What is new here is the ability of Asymmetic AdaBoost to handle high-dimensional predictor space to obtain the conditional binary quantiles. When we make binary prediction under asymmetric loss, the Asymmetric AdaBoost algorithm can provide a reliable algorithm when the predictor space is of high dimension. Section 4 concludes.

2. Component-wise Discrete Symmetric AdaBoost Algorithm

The algorithm of the conventional (symmetric) AdaBoost is as shown in Algorithm 1. Let y be the binary class taking a value in $\{-1, 1\}$ that we wish to predict. Let $f_m(x)$ be the weak learner (weak classifier) for the binary target y that we fit to predict using the high-dimensional covariates x in the mth iteration. Let err_m denote the error rate of the weak learner $f_m(x)$, and $E_w(\cdot)$ denote the weighted expectation (to be defined below) of the variable in the parenthesis with weight w. Note that the error rate $E_w\left[1_{(y \neq f_m(x))}\right]$ is estimated by $err_m = \sum_{i=1}^{n} w_i 1_{(y_i \neq f_m(x_i))}$ with the weight w_i given by step 2(d) from the previous iteration. n is the number of observations. The symbol $1_{(\cdot)}$ is the indicator function which takes the value 1 if a logical condition inside the parenthesis is satisfied and takes the value 0 otherwise. The symbol sign$(z) = 1$ if $z > 0$, sign$(z) = -1$ if $z < 0$, and hence sign$(z) = 1_{(z>0)} - 1_{(z<0)}$.

Algorithm 1 Component-wise Discrete Symmetric AdaBoost

```
1. Start with weights $w_i = \frac{1}{n}, i = 1, \ldots, n$.
2. For $m = 1$ to $M$
    (a) For $j = 1$ to $k$ (for each variable)
        i. Fit the classifier $f_{mj}(x_{ij}) \in \{-1, 1\}$ using weights $w_i$ on the training data.
        ii. Compute $err_{mj} = \sum_{i=1}^{n} w_i 1_{(y_i \neq f_{mj}(x_{ji}))}$.
    (b) Find $\hat{j}_m = \arg\min_j err_{mj}$
    (c) Compute $c_m = \log\left(\frac{1-err_{m,\hat{j}_m}}{err_{m,\hat{j}_m}}\right)$.
    (d) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_{m,\hat{j}_m}(x_{\hat{j}_m,i}))}]$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^{n} w_i = 1$.
3. Output the classifier sign$[F_M(x)]$ where $F_M(x) = \sum_{m=1}^{M} c_m f_{m,\hat{j}_m}(x_{\hat{j}_m})$.
```

To fit the classifier $f_{mj}(x_{ij}) \in \{-1, 1\}$ in step 2(a)i, let us consider two examples of the weak classifier. The first example is the most widely used weak learner, namely the classification tree. The simplest classification tree is the stump, which takes the following functional form

$$f(x_j, a) = \begin{cases} 1 & x_j > a \\ -1 & x_j < a \end{cases}$$

where the parameter a is found by minimizing the error rate

$$\min_a \sum_{i=1}^{n} w_i 1\left(y_i \neq f(x_{ji}, a)\right), \tag{1}$$

where the weight w_i is computed according to the steps in Algorithm 1.

As an alternative to the commonly used classification tree weak learners described above, the second example of the weak learner is a logistic weak learner that considers one variable at a time. In the logistic weak learner, we assume the probability $\pi(x_j) \equiv P(y = 1|x_j) = \frac{e^{x_j\beta}}{1+e^{x_j\beta}}$. Let $Y = \frac{y+1}{2}$. We estimate the parameter β by maximizing the weighted logistic log-likelihood function

$$\max_\beta \log L = \log \prod_{i=1}^{n} \left[\left(\frac{e^{x_{ji}\beta}}{1+e^{x_{ji}\beta}}\right)^{Y_i} \left(\frac{1}{1+e^{x_{ji}\beta}}\right)^{1-Y_i}\right]^{w_i}, \tag{2}$$

where the weight w_i here is also computed according to the steps in Algorithm 1, just like for the classification tree learner in (1). Then the resulting logistic classifier for the weak learner will be

$$f(x_j, \beta, \tau) = \begin{cases} 1 & \pi(x_j, \beta) > 0.5 \\ -1 & \pi(x_j, \beta) < 0.5 \end{cases}$$

where τ takes the value 0.5 in symmetric case. However, in asymmetric case, we suggest to set τ other than 0.5, e.g. the conditional quantile in interest for quantile prediction.

2.1 Understanding Algorithm 1

Friedman et. al. (2000) show that AdaBoost builds an additive logistic regression model

$$F_M(x) = \sum_{m=1}^{M} c_m f_m(x) \tag{3}$$

via Newton-like updates for minimizing the exponential loss

$$J(F) = E\left(e^{-yF(x)}|x\right). \tag{4}$$

We use greedy method to minimize the exponential loss function iteratively.

2.1.1 Looking for the optimal weak learner f_{m+1}

After m iterations, the current classifier is denoted as $F_m(x) = \sum_{s=1}^{m} c_s f_s(x)$. In the next iteration, we are seeking an update $c_{m+1}f_{m+1}(x)$ for the function fitted from previous iterations $F_m(x)$. The updated classifier would take the form

$$F_{m+1}(x) = F_m(x) + c_{m+1}f_{m+1}(x).$$

The loss for $F_{m+1}(x)$ will be

$$\begin{aligned} J(F_{m+1}(x)) &= J(F_m(x) + c_{m+1}f_{m+1}(x)) \\ &= E\left[e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))}\right]. \end{aligned} \tag{5}$$

Expand w.r.t. $f_{m+1}(x)$

$$\begin{aligned} J(F_{m+1}(x)) &\approx E\left[e^{-yF_m(x)}\left[1 - yc_{m+1}f_{m+1}(x) + \frac{y^2c_{m+1}^2f_{m+1}^2(x)}{2}\right]\right] \\ &= E\left[e^{-yF_m(x)}\left(1 - yc_{m+1}f_{m+1}(x) + \frac{c_{m+1}^2}{2}\right)\right]. \end{aligned}$$

The last equality holds since $y \in \{-1, 1\}$, $f_{m+1}(x) \in \{-1, 1\}$, and $y^2 = f_{m+1}^2(x) = 1$. $f_{m+1}(x)$ only appears in the second term in the parenthesis, so minimizing the loss function (5) w.r.t. $f_{m+1}(x)$ is equivalent to maximizing the second term in the parenthesis which results in the following conditional expectation

$$\max_f E\left[e^{-yF_m(x)}yc_{m+1}f_{m+1}(x)|x\right].$$

For any $c > 0$ (we will prove this later), we can omit c_{m+1} in the above objective function

$$\max_f E\left[e^{-yF_m(x)}yf_{m+1}(x)|x\right].$$

To compare it with the AdaBoost algorithm, here we define weight $w = w(y, x) = e^{-yF_m(x)}$. Later we will see that this weight w is equivalent to that shown in the AdaBoost algorithm in the previous section. So the above optimization can be seen as maximizing a weighted conditional expectation

$$\max_f E_w[yf_{m+1}(x)|x] \tag{6}$$

where

$$E_w\left(y|x\right) := \frac{E\left(wy|x\right)}{E\left(w|x\right)} \tag{7}$$

refers to a weighted conditional expectation. Note that (6) can be written as

$$\begin{aligned}
& E_w\left[yf_{m+1}\left(x\right)|x\right] \\
= \; & P_w\left(y=1|x\right)f_{m+1}\left(x\right) - P_w\left(y=-1|x\right)f_{m+1}\left(x\right) \\
= \; & \left[P_w\left(y=1|x\right) - P_w\left(y=-1|x\right)\right]f_{m+1}\left(x\right) \\
= \; & E_w\left(y|x\right)f_{m+1}\left(x\right).
\end{aligned} \tag{8}$$

where

$$P_w\left(y|x\right) := \frac{E(w|y,x)P\left(y|x\right)}{E\left(w|x\right)}. \tag{9}$$

Solve the maximization problem (6). Since $f_{m+1}\left(x\right)$ only takes 1 or -1, it should be positive whenever $E_w\left(y|x\right)$ is positive and -1 whenever $E_w\left(y|x\right)$ is negative. The solution for $f_{m+1}\left(x\right)$ is

$$f_{m+1}\left(x\right) = \begin{cases} 1 & E_w\left(y|x\right) > 0 \\ -1 & \text{otherwise.} \end{cases} \tag{10}$$

2.1.2 Looking for the optimal learning rate c_{m+1}

Minimize the loss function (5) w.r.t. c_{m+1}

$$c_{m+1} = \arg\min_{c_{m+1}} E_w\left(e^{-c_{m+1}yf_{m+1}(x)}\right), \tag{11}$$

where

$$E_w\left(e^{-c_{m+1}yf_{m+1}(x)}\right) = P_w\left(y = f_{m+1}\left(x\right)\right)e^{-c_{m+1}} + P_w\left(y \neq f_{m+1}\left(x\right)\right)e^{c_{m+1}}. \tag{12}$$

Then the first order condition is

$$\frac{\partial E_w\left(e^{-c_{m+1}yf_{m+1}(x)}\right)}{\partial c_{m+1}} = -P_w\left(y = f_{m+1}\left(x\right)\right)c_{m+1}e^{-c_{m+1}} + P_w\left(y \neq f_{m+1}\left(x\right)\right)c_{m+1}e^{c_{m+1}} = 0. \tag{13}$$

Thus we have

$$P_w\left(y = f_{m+1}\left(x\right)\right)c_{m+1}e^{-c_{m+1}} = P_w\left(y \neq f_{m+1}\left(x\right)\right)c_{m+1}e^{c_{m+1}}, \tag{14}$$

where

$$P_w\left(y\right) := \frac{E(w|y)P(y)}{E(w)}. \tag{15}$$

Solving it for c_{m+1}, we obtain

$$c_{m+1} = \frac{1}{2}\log\frac{P_w\left(y = f_{m+1}\left(x\right)\right)}{P_w\left(y \neq f_{m+1}\left(x\right)\right)} = \frac{1}{2}\log\left(\frac{1 - err_{m+1}}{err_{m+1}}\right),$$

where

$$err_{m+1} := P_w\left(y \neq f_{m+1}\left(x\right)\right) \tag{16}$$

is the error rate of $f_{m+1}\left(x\right)$. Note that $c_{m+1} > 0$ as long as the error rate is smaller than 50%. Our assumption $c_{m+1} > 0$ holds for any learner that is better than random guessing.

2.1.3 Updating the weight w_m and the strong learner F_m

Now we have finished the steps of one iteration and can get our updated classifier by

$$F_{m+1}(x) \leftarrow F_m(x) + \left(\frac{1}{2}\log\left(\frac{1-err_{m+1}}{err_{m+1}}\right)\right) f_{m+1}(x). \tag{17}$$

Note that in the next iteration, the weight we defined w_{m+1} will be

$$w_{m+1} = e^{-yF_{m+1}(x)} = e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))} = w_m \times e^{-c_{m+1}f_{m+1}(x)y}. \tag{18}$$

Since $-yf_{m+1}(x) = 2 \times 1_{\{y \neq f_{m+1}(x)\}} - 1$, the update is equivalent (up to the normalization) to

$$w_{m+1} = w_m \times e^{\left(\log\left(\frac{1-err_{m+1}}{err_{m+1}}\right)1_{[y\neq f_{m+1}(x)]}\right)} = w_m \times \left(\frac{1-err_{m+1}}{err_{m+1}}\right)^{1_{[y\neq f_{m+1}(x)]}}.$$

Thus the function and weights update are of an identical form to those used in AdaBoost. AdaBoost could do better than any single weak classifier since it iteratively minimizes the loss function via a Newton-like procedure.

2.2 Understanding AdaBoost as a Logistic Regression

Interestingly, the function $F(x)$ from minimizing the exponential loss is the same as maximizing a logistic log-likelihood. Let

$$J(F(x)) = E\left[E\left(e^{-yF(x)}|x\right)\right] \tag{19}$$

$$= E\left[P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)}\right]. \tag{20}$$

Taking derivative w.r.t. $F(x)$ and making it equal to zero, we obtain

$$\frac{\partial E\left(e^{-yF(x)}|x\right)}{\partial F(x)} = -P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)} = 0$$

$$F^*(x) = \frac{1}{2}\log\left[\frac{P(y=1|x)}{P(y=-1|x)}\right].$$

Moreover, if the true probability

$$P(y=1|x) = \frac{e^{2F(x)}}{1+e^{2F(x)}},$$

for $Y = \frac{y+1}{2}$, the log-likelihood is

$$E(\log L|x) = E\left[2YF(x) - \log\left(1+e^{2F(x)}\right)|x\right].$$

The solution $F^*(x)$ that maximize the log-likelihood must equals the $F(x)$ in the true model $P(y=1|x) = \frac{e^{2F(x)}}{1+e^{2F(x)}}$. Hence,

$$e^{2F^*(x)} = P(y=1|x)\left(1+e^{2F^*(x)}\right)$$

$$e^{2F^*(x)} = \frac{P(y=1|x)}{1-P(y=1|x)}$$

$$F^*(x) = \frac{1}{2}\log\left[\frac{P(y=1|x)}{P(y=-1|x)}\right].$$

AdaBoost that minimizes the exponential loss yield the same solution as the logistic regression that maximizes the logistic log-likelihood.

2.3 Understanding AdaBoost as a Symmetric Algorithm

Now we expand the results of Friedman et. al. (2000) and show that AdaBoost is a symmetric algorithm which is equivalent (up to scale) to the least square. At the end of the algorithm, AdaBoost yields a binary prediction rule which is the sign of $F(x)$. The predictor from AdaBoost is equivalent to a binary function $G(x)$ from minimizing the exponential loss function $E\left(e^{-yG(x)}\right)$ with respect to $G(x)$.

Theorem 1. *If $F(x) \in \mathbb{R}$ is the solution for*

$$\min_{F(x)} E\left(e^{-yF(x)}|x\right)$$

and G$(x) \in \{1, -1\}$ *is the solution for*

$$\min_{G(x)} E\left(e^{-yG(x)}|x\right),$$

we have

$$\textit{sign}\left[F(x)\right] = G(x).$$

Proof. The first order condition for

$$\min_{F(x)} E\left(e^{-yF(x)}|x\right)$$

is

$$E\left(-ye^{-yF(x)}|x\right) = -P(y=1|x)\, e^{-F(x)} + P(y=-1|x)\, e^{F(x)} = 0$$

Hence

$$F(x) = \frac{1}{2}\log\left(\frac{P(y=1|x)}{P(y=-1|x)}\right)$$

and

$$\text{sign}\left[F(x)\right] = \begin{cases} 1 & P(y=1|x) > P(y=-1|x) \\ -1 & otherwise. \end{cases}$$

Moreover,

$$E\left(e^{-yG(x)}|x\right) = \begin{cases} P(y=1|x)\, e^{-1} + P(y=-1|x)\, e & G(x) = 1 \\ P(y=1|x)\, e + P(y=-1|x)\, e^{-1} & G(x) = -1, \end{cases}$$

Hence,

$$G(x) = \begin{cases} 1 & P(y=1|x) > P(y=-1|x) \\ -1 & otherwise. \end{cases}$$

Thus, the proof is complete. □

Relating the predictor sign$\left[F(x)\right]$ yields from AdaBoost with a binary function $G(x)$ can give us better insights of the symmetry of AdaBoost.

Remark 1. The exponential loss function $E\left[e^{-yG(x)}\right]$ is equivalent to maximizing the maximum score.

Proof. Taking Taylor expansion of $E\left[e^{-yG(x)}\right]$ around $G(x)=0$, since $y^2=1$ and $G^2(x)=1$, we have that

$$\begin{aligned} E\left[e^{-yG(x)}\right] &= E\left[1-yG(x)+\frac{y^2G^2(x)}{2}-\frac{y^3G^3(x)}{6}+\frac{y^4G^4(x)}{24}+\cdots\right] \\ &= E\left[1-yG(x)+\frac{1}{2}-\frac{yG(x)}{6}+\frac{1}{24}+\cdots\right] \\ &= E\left[a-byG(x)\right] \end{aligned}$$

The last equality holds since that the odd order terms in the Taylor expansion can be written as constant multiples of $-yG(x)$ and the even order terms are constant.

Therefore, minimizing the exponential loss function $E\left[e^{-yG(x)}\right]$ is equivalent to

$$\min_{G(x)} E\left(-yG(x)\right) \tag{21}$$

which has been introduced by Manski (1975) in economic literature long before AdaBoost has been invented. □

Remark 2. Minimizing the exponential loss is equivalent to minimizing the squared error loss $E\left[(y-G(x))^2\right]$.

Proof. Minimizing the exponential loss $E\left[t(y,x)\,e^{-yG(x)}\right]$ is equivalent to

$$\min_{G(x)} E\left(-t(y,x)\,yG(x)\right).$$

Furthermore, we have

$$\begin{aligned} E\left[-yG(x)\right] &= \frac{1}{2}\times\left\{2E\left[-yG(x)\right]+2\right\}-1 \\ &= \frac{1}{2}\times E\left[y^2-2yG(x)+G^2(x)\right]-1 \\ &= \frac{1}{2}\times E\left[y-G(x)\right]^2-1 \end{aligned}$$

Hence, minimizing $E\left[e^{-yG(x)}\right]$ is equivalent to minimizing the squared error loss function $E\left[y-G(x)\right]^2$ which is symmetric. □

Therefore, AdaBoost yield same solution as minimizing a squared errors loss. Moreover, in binary settings, this result can be directly extended to symmetric check loss.

Remark 3. For $y\in\{-1,1\}$ and $G(x)\in\{-1,1\}$, minimizing the L_2 loss is equivalent to minimizing the L_1 loss.

Proof. we have that the L_2 loss function

$$L_2(y-G)=\begin{cases}(y-G)^2=4, & y>f \text{ if and only if } y=1 \text{ and } f=-1\\ 0, & y=f\\ (y-G)^2=4, & y<f \text{ if and only if } y=-1 \text{ and } f=1\end{cases} \tag{22}$$

which is equivalent to the L_1 loss function

$$L_1(y-G)=\begin{cases}|y-G|=2, & y>f \text{ if and only if } y=1 \text{ and } f=-1\\ 0, & y=f\\ |y-G|=2, & y<f \text{ if and only if } y=-1 \text{ and } f=1\end{cases} \tag{23}$$

up to a constant scale of $\frac{1}{2}$, i.e., the L_2 loss is always twice the L_1 loss. So minimizing the L_2 loss is equivalent to minimizing the L_1 loss for this binary problem. □

Therefore the L_1 loss and the L_2 loss in the binary setup would yield the same result. Hence, AdaBoost can be seen as getting an additive model via least square or least absolute loss regression.

3. Component-wise Discrete Asymmetric AdaBoost

In this section, we explore the properties of the new asymmetric exponential loss function in detail. It solves the binary decision problem with asymmetric loss function. The key idea is to assign larger penalty on cases that may lead to higher losses and lower penalty where losses are limited. An algorithm for minimizing the asymmetric exponential loss is also provided in this section.

3.1 Asymmetric Least Squares

In Theorem 1 we established the equivalence between the symmetric exponential loss and least square loss. Now we explore on the relation between the asymmetric exponential loss and the asymmetric least square loss of Newey and Powell (1987) where the loss function is defined as

$$\rho_\tau (y-f) = \left|\tau - 1_{(y<f)}\right| \cdot (y-f)^2, \text{ for } \tau \in (0,1). \tag{24}$$

In the binary case where $f \in \{-1,1\}$ and $y \in \{-1,1\}$, this becomes

$$\rho_\tau (y-f) = \begin{cases} \tau (y-f)^2 & y>f \text{ (when } y=1 \text{ and } f=-1) \\ 0 & y=f \\ (1-\tau)(y-f)^2 & y<f \text{ (when } y=-1 \text{ and } f=1). \end{cases} \tag{25}$$

So without changing the behavior of $\rho_\tau(y-f)$, we can replace $\left|\tau - 1_{(y<f)}\right|$ with a function of y

$$t(y,x) = \begin{cases} \tau & y=1 \\ (1-\tau) & y=-1. \end{cases} \tag{26}$$

or $t(y,x) = \left|\tau - 1_{(y=-1)}\right|$.

3.2 Asymmetric Exponential Loss

Theorem 2. *If $F(x) \in \mathbb{R}$ is the solution for*

$$\min_{F(x)} E\left(t(y,x)\, e^{-yF(x)}|x\right) \tag{27}$$

and $G(x) \in \{1,-1\}$ is the solution for

$$\min_{G(x)} E\left(t(y,x)\, e^{-yG(x)}|x\right), \tag{28}$$

we have

$$\textit{sign}\,[F(x)] = G(x).$$

Proof. The first order condition for

$$\min_{F(x)} E\left(t(y,x)\, e^{-yF(x)}|x\right)$$

is

$$E\left(-ye^{-yF(x)}|x\right) = -P(y=1|x)\, t(1,x)\, e^{-F(x)} + P(y=-1|x)\, t(-1,x)\, e^{F(x)} = 0$$

Hence

$$F(x) = \frac{1}{2}\log\left(\frac{P(y=1|x)\,t(1,x)}{P(y=-1|x)\,t(-1,x)}\right)$$

and

$$\text{sign}[F(x)] = \begin{cases} 1 & P(y=1|x)\times t(1,x) > P(y=-1|x)\times t(-1,x) \\ -1 & otherwise. \end{cases}$$

Moreover,

$$E\left(t(y,x)\,e^{-yG(x)}|x\right) = \begin{cases} P(y=1|x)\,e^{-1}\times t(1,x) + P(y=-1|x)\,e\times t(-1,x) & G(x)=1 \\ P(y=1|x)\,e\times t(1,x) + P(y=-1|x)\,e^{-1}\times t(-1,x) & G(x)=-1, \end{cases}$$

Hence,

$$G(x) = \begin{cases} 1 & P(y=1|x)\times t(1,x) > P(y=-1|x)\times t(-1,x) \\ -1 & otherwise. \end{cases}$$

Thus, the proof is complete. □

3.3 Equivalence of Asymmetric Exponential Loss and Asymmetric Least Squares

Theorem 3. *Minimizing the exponential loss function $E\left[t(y,x)\,e^{-yG(x)}\right]$ is equivalent to minimizing the asymmetric squared errors loss function $E\left[t(y,x)\,(y-G(x))^2\right]$.*

Proof. Taking Taylor expansion of $E\left[t(y,x)\,e^{-yG(x)}\right]$ around $G(x)=0$, since $y^2=1$ and $G^2(x)=1$, we have that

$$\begin{aligned} E\left[t(y,x)\,e^{-yG(x)}\right] &= E\left[t(y,x)\left(1-yG(x)+\frac{y^2G^2(x)}{2}-\frac{y^3G^3(x)}{6}+\frac{y^4G^4(x)}{24}+\cdots\right)\right] \\ &= E\left[t(y,x)\left(1-yG(x)+\frac{1}{2}-\frac{yG(x)}{6}+\frac{1}{24}+\cdots\right)\right]. \end{aligned}$$

Note that the odd order terms in the Taylor expansion can be written as constant multiples of $-t(y,x)\,yG(x)$ and the even order terms are constant. Therefore, minimizing the exponential loss function $E\left[t(y,x)\,e^{-yG(x)}\right]$ is equivalent to

$$\min_{F(x)} E\left(-t(y,x)\,yG(x)\right).$$

Furthermore, we have

$$\begin{aligned} E\left[-t(y,x)\,yG(x)\right] &= \frac{1}{2}\times 2E\left\{-t(y,x)\,yG(x)+t(y,x)-t(y,x)\right\} \\ &= \frac{1}{2}\times E\left\{t(y,x)\left[-2yG(x)+2-2\right]\right\} \\ &= \frac{1}{2}\times E\left\{t(y,x)\left[y^2-2yG(x)+G^2(x)-2\right]\right\} \\ &= \frac{1}{2}\times E\left\{t(y,x)\left[y-G(x)\right]^2-2t(y,x)\right\} \\ &= \frac{1}{2}\times E\left\{t(y,x)\left[y-G(x)\right]^2\right\}-E\left[t(y,x)\right]. \end{aligned}$$

Hence, minimizing $E\left[t(y,x)\,e^{-yG(x)}\right]$ is equivalent to minimizing the asymmetric squared error loss function $E\left[t(y,x)\,(y-G(x))^2\right]$. □

Then minimizing (28) will be exactly the same as minimizing an asymmetric least squares loss function. It is easy to see that the symmetric AdaBoost in Section 2 is a special case of $t(y,x)$ where $\tau=0.5$.

3.4 Binary Expectile Regression and Binary Quantile Regression

Theorem 4. *For $y \in \{-1, 1\}$ and $G(x) \in \{-1, 1\}$, minimizing the asymmetric L_2 loss function is equivalent to minimizing the asymmetric L_1 loss function.*

Proof. we have that the L_2 loss function

$$L_2(y-G) = \begin{cases} \tau (y-G)^2 = 4\tau, & y > f \text{ if and only if } y = 1 \text{ and } f = -1 \\ 0, & y = f \\ (1-\tau)(y-G) = 4(1-\tau), & y < f \text{ if and only if } y = -1 \text{ and } f = 1 \end{cases} \tag{29}$$

which is equivalent to the L_1 loss function

$$L_1(y-G) = \begin{cases} \tau |y-G| = 2\tau, & y > f \text{ if and only if } y = 1 \text{ and } f = -1 \\ 0, & y = f \\ (1-\tau)|y-G| = 2(1-\tau), & y < f \text{ if and only if } y = -1 \text{ and } f = 1 \end{cases} \tag{30}$$

up to a constant scale of $\frac{1}{2}$, i.e., the L_2 loss is always twice the L_1 loss. So minimizing the L_2 loss is equivalent to minimizing the L_1 loss for this binary problem. □

Hence, the expectile from minimizing the Newey and Powell (1987)'s asymmetric L_2 loss function is the same as the binary quantile regression from minimizing the asymmetric L_1 loss function of Koenker and Bassett (1978) and Koenker (2005). Therefore the Asymmetric AdaBoost gives the same solution to the binary quantile regression as well as to the binary expectile regression.

3.5 Introducing Asymmetric AdaBoost Algorithm

Next, we propose an algorithm to minimize our new asymmetric exponential loss function. We take similar steps as in the symmetric case.

Algorithm 2 Component-wise Discrete Asymmetric AdaBoost

```
Start with weights $w_i = \frac{t(y_i,x_i)}{\sum_{s=1}^{n} t(y_s,x_s)}$, $i = 1, \ldots, n$.
  1. For $m = 1$ to $M$
      (a) For $j = 1$ to $k$ (for each variable)
          i. Fit the classifier $f_{mj}(x_{ij}) \in \{-1, 1\}$ using weights $w_i$.
          ii. Compute $err_{mj} = \sum_{i=1}^{n} w_i 1_{(y_i \neq f_{mj}(x_{ji}))}$.
      (b) Find $\hat{j}_m = \arg\min_j err_{mj}$.
      (c) Compute $c_m = \log\left(\frac{1-err_{m,\hat{j}_m}}{err_{m,\hat{j}_m}}\right)$.
      (d) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_{m,\hat{j}_m}(x_{\hat{j}_m,i}))}]$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^{n} w_i = 1$.
  2. Output the classifier, sign$\left[\sum_{m=1}^{M} c_m f_{m,\hat{j}_m}(x_{\hat{j}_m})\right]$.
```

Note that the error rate $E_w\left(1_{(y\neq f_{mj}(x_j))}\right) := E\left(w_i \times 1_{(y\neq f_{mj}(x_j))}\right)$ is estimated by $err_{mj} = \sum_{i=1}^{n} w_i \times 1_{(y_i \neq f_{mj}(x_{ji}))}$. The same classifiers can be used for the weak learner as in the symmetric AdaBoost, e.g. the stump learner in (1) and the logistic classifier in (2).

3.6 Understanding Algorithm 2

To better understand the new algorithm, we take the similar steps as in the symmetric AdaBoost. We start with the asymmetric exponential loss function

$$J\left(F\left(x\right)\right) = E\left(t\left(y,x\right)e^{-yF(x)}\right). \tag{31}$$

We have derived the optimal weak learner f and the optimal learning rate c. Only a summary is provided below. Proofs and more detailed discussion will be provided in the full version of the paper to be published elsewhere, which includes the asymptotic theory, Monte Carlo simulation, applications, variants of the new algorithm, and more extensive remarks and discussions on various properties of the Asymmetric AdaBoost algorithm.

3.6.1 Optimal weak learner $f_{m+1}(x)$

For the purpose of showing how the asymmetric function $t\left(y,x\right)$ changes the results of the algorithm, it can be shown that the solution for the optimal weak learner $f_{m+1}\left(x\right)$ is

$$f_{m+1}\left(x\right) = \begin{cases} 1, & E_w\left(t\left(y,x\right)y|x\right) > 0 \\ -1, & \text{otherwise,} \end{cases} \tag{32}$$

where

$$E_w\left(t\left(y,x\right)y|x\right) = P_w\left(y=1|x\right)t\left(1,x\right) - P_w\left(y=-1|x\right)t\left(-1,x\right). \tag{33}$$

3.6.2 Optimal learning rate c_{m+1}

After solving $f_{m+1}\left(x\right)$, we minimize the loss function w.r.t. c_{m+1}. We show that the first order condition from taking the derivative w.r.t. c_{m+1} gives the optimal c_{m+1}

$$c_{m+1} = \frac{1}{2}\log\left(\frac{1-err_{m+1}}{err_{m+1}}\right), \tag{34}$$

where

$$err_{m+1} = \text{FN} \times t\left(1,x\right) + \text{FP} \times t\left(-1,x\right) = E_w\left(t\left(y,x\right) \times 1_{(y\neq f_{m+1}(x))}\right), \tag{35}$$

where $P_w\left(y=1, f_{m+1}\left(x\right)=-1\right)$ is the rate of false negative (FN) and $P_w\left(y=-1, f_{m+1}\left(x\right)=1\right)$ is the rate of false positive (FP).

3.6.3 Updating the weight w_m and the strong learner F_m

In the next iteration, we have

$$F_{m+1}\left(x\right) = F_m\left(x\right) + c_{m+1}f_{m+1}\left(x\right).$$

Hence

$$\begin{aligned} w_{m+1} &= e^{-yF_{m+1}(x)} \\ &= e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))} \\ &= w_m \times e^{-c_{m+1}yf_{m+1}(x)} \end{aligned}$$

So our algorithm tends to give larger weights to the observations when $y \neq f_{m+1}\left(x\right)$ as long as a weak learner is not too weak with $err_{m+1} < 0.5$ and $c_{m+1} > 0$.

4. Conclusions

In this paper, we introduce a new Asymmetric AdaBoost algorithm which produces an additive logistic regression model from maximizing a new loss function, namely the asymmetric exponential loss function. The Asymmetric AdaBoost algorithm is based on the asymmetric exponential loss function. In the binary classification, we show that the asymmetric exponential loss function is equivalent to the asymmetric L_2 loss function of Newey and Powell (1987) and the asymmetric L_1 loss function of Koenker and Bassett (1978) and Koenker (2005). Therefore the Component-wise Discrete Asymmetric AdaBoost algorithm gives a solution to the high-dimensional binary quantile regression and the binary expectile regression.

REFERENCES

Fan, W., S.J. Stolfo, J. Zhang, and P.K. Chan (1999), "AdaCost: misclassification cost-sensitive boosting." *ICML* 99: 97-105.

Freund, Y. and Schapire, R. E. (1996), "Experiments with a new boosting algorithm." *In Machine Learning: Proceedings of the Thirteenth International Conference*: 148–156. Morgan Kaufman, San Francisco.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2000), "Additive logistic regression: a statistical view of boosting (with discussion)." *The Annals of Statistics* 28(2): 337-407.

Koenker, R., Bassett, G. (1978), "Regression quantiles". *Econometrica* 46, 33-50.

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

Masnadi-Shirazi, Hamed and Nuno Vasconcelos (2007), "Asymmetric boosting." *Proceedings of the 24th International Conference on Machine Learning*: 609-619.

Newey, Whitney K. and James L. Powell (1987), "Asymmetric least squares estimation and testing." *Econometrica* 55(4): 819-847.

Sun, Yanmin, Andrew K.C. Wong, and Yang Wang (2005), "Parameter inference of cost-sensitive boosting algorithms." *Machine Learning and Data Mining in Pattern Recognition*: 21-30.

Ting, Kai Ming (2000), "A comparative study of cost-sensitive boosting algorithms." *In Proceedings of the 17th International Conference on Machine Learning*.

Viola, Paul and Michael Jones (2002), "Fast and robust classification using asymmetric adaboost and a detector cascade." *Advances in Neural Information Processing System 14*: 1311-1318.