Unified Robust Boosting

Zhu Wang
UT Health San Antonio

Abstract

Boosting is a popular machine learning algorithm in regression and classification problems. Boosting can combine a sequence of regression trees to obtain accurate prediction. In the presence of outliers, traditional boosting, based on optimizing convex loss functions, may show inferior results. In this article, a unified robust boosting is proposed for more resistant estimation. The method utilizes a recently developed concave-convex family for robust estimation, composite optimization by conjugation operator, and functional decent boosting. As a result, an iteratively reweighted boosting algorithm can be conveniently constructed with existing software. Applications in robust regression, classification and Poisson regression are demonstrated in the R package **ccboost**.

Keywords: Robust method, CC-family, CC-estimation, CC-boosting, boosting, COCO.

1. Introduction

Boosting is a powerful supervised machine learning algorithm. As an ensemble method, boosting combines many weak learners to generate a strong prediction. As a functional decent method, boosting has a wide applications in regression and classification problems. Friedman (2001); Friedman, Hastie, and Tibshirani (2000); Bühlmann and Hothorn (2007) discussed boosting for a variety of convex loss functions. To deal with outliers, robust estimation has been brought into boosting in order to provide more accurate estimation. Wang (2018a,b) proposed robust functional gradient boosting for nonconvex loss functions. These methods applied majorization-minimization (MM) scheme, an extension of the popular expectation-maximization (EM) algorithm in statistics.

Recently, Wang (2020) innovatively proposed a unified robust loss function family, the concave convex (CC) family, and introduced the composite optimization by conjugation operator (COCO) to obtain the CC-estimation. The CC-family includes traditional robust loss functions such as the Huber loss, robust hinge loss for support vector machine, and robust exponential family for generalized linear models. The COCO algorithm is an iteratively reweighted estimation and can be conveniently implemented from the existing methods and software. In this article, we integrate the COCO and boosting to obtain CC-estimation in the context of function estimation, which is more broad than the linear predictor function in Wang (2020). For instance, the CC-boosting algorithm permits function space derived from the regression trees. We illustrate the proposed algorithm through the **ccboost** R package with applications to robust exponential family, including regression, binary classification and Poisson regression.

Concave	$g(z)$
hcave	$\begin{cases} z & \text{if } z \le \sigma^2/2 \\ \sigma(2z)^{\frac{1}{2}} - \frac{\sigma^2}{2} & \text{if } z > \sigma^2/2 \end{cases}$
acave	$\begin{cases} \sigma^2(1-\cos(\frac{(2z)^{\frac{1}{2}}}{\sigma})) & \text{if } z \le \sigma^2\pi^2/2 \\ 2\sigma^2 & \text{if } z > \sigma^2\pi^2/2 \end{cases}$
bcave	$\frac{\sigma^2}{6}\left(1-(1-\frac{2z}{\sigma^2})^3 I(z \le \sigma^2/2)\right)$
ccave	$\sigma^2\left(1-\exp(\frac{-z}{\sigma^2})\right)$
dcave	$\frac{1}{1-\exp(-\sigma)}\log(\frac{1+z}{1+z\exp(-\sigma)})$
ecave	$\begin{cases} \frac{2\exp(-\frac{\delta}{\sigma})}{\sqrt{\pi\sigma\delta}} z & \text{if } z \le \delta \\ \text{erf}(\sqrt{\frac{z}{\sigma}}) - \text{erf}(\sqrt{\frac{\delta}{\sigma}}) + \frac{2\exp(-\frac{\delta}{\sigma})}{\sqrt{\pi\sigma\delta}}\delta & \text{if } z > \delta \end{cases}$
gcave	$\begin{cases} \frac{\delta^{\sigma-1}}{(1+\delta)^{\sigma+1}} z & \text{if } z \le \delta \\ \frac{1}{\sigma}(\frac{z}{1+z})^\sigma - \frac{1}{\sigma}(\frac{\delta}{1+\delta})^\sigma + \frac{\delta^\sigma}{(1+\delta)^{\sigma+1}} & \text{if } z > \delta \end{cases}$ where $\delta = \begin{cases} \to 0+ & \text{if } 0 < \sigma < 1 \\ \frac{\sigma-1}{2} & \text{if } \sigma \ge 1 \end{cases}$
tcave	$\min(\sigma, z), \quad \sigma \ge 0$

Table 1: Concave component with $\sigma > 0$ except for tcave with $\sigma \ge 0$.

2. Robust CC-boosting

2.1. CC-function estimation

To unify robust estimation, Wang (2020) proposed the concave convex (CC) family with functions Γ satisfying the following conditions:

i. $\Gamma = g \circ s$

ii. g is a nondecreasing closed concave function on range s

iii. $\partial(-g(z))\ \forall z \in$ range s is nonempty and bounded

iv. s is convex on $\mathbb{R}$.

Table 1 lists some concave component. The convex component includes common loss functions in regression and classification such as least squares and logistic function. More broadly, the convex component contains the negative log-likelihood function in the exponential family adopted by the generalized linear models. Since the convex component can be non-differentiable, subgradient and subdifferential are useful tools. For instance, the `tcave` is not differentiable at $z = \sigma$, but is quite useful to truncate loss functions. Given a set of observations $(\mathbf{x}_i, y_i), i = 1, ..., n$ where $y_i \in \mathbb{R}$ and $\mathbf{x}_i = (x_{i1}, ..., x_{ip})^\intercal \in \mathbb{R}^p$, denote Ω the linear span of a set H of base learners including regression trees and linear predictor functions. Denote

$\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_n))^\intercal, \mathbf{f} \in \Omega$. We aim to minimize an empirical loss function

$$\sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)). \tag{1}$$

Here ℓ is a member of the CC-family, $\ell = g \circ s = g(s(u)) = g(s(y, f))$. To simplify notations, f and $f(\mathbf{x})$ are interchanged sometimes. For $i = 1, ..., n$, u_i is defined below:

$$u_i = \begin{cases} y_i - f_i, & \text{for regression,} \\ y_i f_i, & \text{for classification with } y_i \in [-1, 1], \\ f_i, & \text{for exponential family.} \end{cases} \tag{2}$$

Robust function estimation can be accomplished by the following algorithm.

Algorithm 1 CC-Function Estimation Algorithm

1: **Input:** training samples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, concave component g with parameter σ, convex component s, starting point $\mathbf{f}^{(0)}$ and iteration count K.
2: **for** $k = 1$ to K **do**
3: Compute $z_i = s(y_i, f_i^{(k-1)}), i = 1, ..., n$
4: Compute subgradient $v_i^{(k)}$ via $v_i^{(k)} \in \partial(-g(z_i))$ or $z_i \in \partial\varphi(v_i^{(k)}), i = 1, ..., n$
5: Compute $\mathbf{f}^{(k)} = \text{argmin}_{\mathbf{f}\in\Omega} \sum_{i=1}^{n} s(y_i, f_i)(-v_i^{(k)})$
6: **end for**
7: **Output:** $v_i^{(K)}$ and $\mathbf{f}^{(K)}$.

We have the convergence results for the COCO algorithm.

Theorem 1. *Suppose that g is a concave component in the CC-family, and g is bounded below. The loss function values $\rho(\mathbf{f}^{(k)}) \triangleq \sum_{i=1}^{n} \ell(y_i, f_i^{(k)})$ generated by Algorithm 1 are nonincreasing and converge.*

This result is a generalization of Theorem 4 in Wang (2020) in which the function is restricted to a linear predictor function. Here we study more broadly defined function spaces. On the other hand, if H is a space of linear models, Theorem 1 indeed coincides with the results in Wang (2020). The proof follows the same argument of Theorem 4 in Wang (2020), hence only a sketch is outlined. Define the surrogate loss function:

$$Q(\mathbf{f}|\mathbf{f}^{(k)}) = \sum_{i=1}^{n} s(y_i, f_i)(-v_i^{(k+1)}) + \varphi(v_i^{(k+1)}).$$

Apply the well-known results on the conjugation operator, we then have

$$\rho(\mathbf{f}^{(k+1)}) \leq Q(\mathbf{f}^{(k+1)}|\mathbf{f}^{(k)}) \leq Q(\mathbf{f}^{(k)}|\mathbf{f}^{(k)}) = \rho(\mathbf{f}^{(k)}). \tag{3}$$

Step 5 in the algorithm is equivalent to minimizing $Q(\mathbf{f}|\mathbf{f}^{(k)})$ since $\varphi(v_i^{(k+1)})$ is a constant with respect to $\mathbf{f}$. The conclusion of the theorem follows.

2.2. Boosting algorithm for function estimation

An important question is how to compute step 5 in Algorithm 1. Here we adopt weighted boosting algorithm. Boosting as a method for function estimation has been well studied by

Friedman *et al.* (2000); Friedman (2001). Boosting can be utilized to fit a variety of models with different base learners, including linear least squares, smoothing splines and regression trees (Bühlmann and Hothorn 2007; Wang 2018b). For ease of notation, we first consider unweighted estimation:

$$\underset{\mathbf{f}\in\Omega}{\operatorname{argmin}} \sum_{i=1}^{n} s(y_i, f_i). \tag{4}$$

In a boosting algorithm, the solution is an additive model given by

$$\hat{f}_i = F_M(\mathbf{x}_i) = \sum_{i=1}^{M} t_m(\mathbf{x}_i), \tag{5}$$

where $F_M(\mathbf{x}_i)$ is stagewisely constructed by sequentially adding an update $t_m(\mathbf{x}_i)$ to the current estimate $F_{m-1}(\mathbf{x}_i)$:

$$F_m(\mathbf{x}_i) = F_{m-1}(\mathbf{x}_i) + t_m(\mathbf{x}_i), m = 1, ..., M. \tag{6}$$

There are different ways to compute $\mathbf{t}_m(\mathbf{x}) = (t_m(\mathbf{x}_1), ..., t_m(\mathbf{x}_n))^\intercal$: gradient and Newton-type update are the most popular (Sigrist 2020). When the second derivative of loss function exists, the Newton-type update is preferred over gradient update to achieve fast convergence:

$$\mathbf{t}_m(\mathbf{x}) = \underset{\mathbf{f}\in H}{\operatorname{argmin}} \sum_{i=1}^{n} h_{m,i}(-\frac{d_{m,i}}{h_{m,i}} - f(x_i))^2, \tag{7}$$

where the first and second derivatives of the loss function s for observations i are given by:

$$d_{m,i} = \frac{\partial}{\partial f} s(y_i, f)|_{f=F_{m-1}(x_i)}, \tag{8}$$

$$h_{m,i} = \frac{\partial^2}{\partial f^2} s(y_i, f)|_{f=F_{m-1}(x_i)}. \tag{9}$$

For quadratic loss $s(y_i, f) = \frac{(y_i-f)^2}{2}$, we obtain $h_{m,i} = 1$. In this case, the Newton-update becomes the gradient update. Furthermore, the weighted minimization problem in step 5 of Algorithm 1 can be solved with the weighted boosting algorithm.

2.3. Penalized estimation

To avoid overfitting, we can add the objective function with a regularization term:

$$\sum_{i=1}^{n} \ell(y_i, \hat{f}_i) + \sum_{m=1}^{M} \Lambda(t_m), \tag{10}$$

where Λ penalizes the model complexity. If H is the space of linear regression with a p-dimensional predictor, i.e., $t_m(\mathbf{x}_i) = \mathbf{x}_i^\intercal \boldsymbol{\beta}_m, \boldsymbol{\beta}_m = (\beta_{1m}, ..., \beta_{pm})^\intercal$, denote

$$\Lambda(t_m) = \frac{1}{2}\lambda \sum_{j=1}^{p} \beta_{jm}^2 + \alpha \sum_{j=1}^{p} |\beta_{jm}|, \tag{11}$$

where $\lambda \geq 0, \alpha \geq 0$. Note that $\Lambda(t_m)$ provides shrinkage estimators and can conduct variable selection. Suppose that H is the space of regression trees. Each regression tree splits the whole predictor space into disjoint hyper-rectangles with sides parallel to the coordinate axes (Wang 2018b). Specifically, denote the hyper-rectangles in the m-th boosting iteration $R_{jm}, j = 1, ..., J$. Let $t_m(\mathbf{x}_i) = \beta_{jm}, \mathbf{x}_i \in R_{jm}, i = 1, ..., n, j = 1, ..., J$. With $\gamma \geq 0$, the penalty can be defined as in Chen and Guestrin (2016):

$$\Lambda(t_m) = \gamma J + \frac{1}{2}\lambda \sum_{j=1}^{p} \beta_{jm}^2 + \alpha \sum_{j=1}^{p} |\beta_{jm}|. \tag{12}$$

A different penalized estimation is to implement a shrinkage parameter $0 < \nu \leq 1$ in the update (6):

$$F_m(\mathbf{x}_i) = F_{m-1}(\mathbf{x}_i) + \nu t_m(\mathbf{x}_i), m = 1, ..., M. \tag{13}$$

2.4. Implementation and tuning parameter selection

In summary, we use Algorithm 1 coupled with the boosting algorithm to minimize the following objective function:

$$\sum_{i=1}^{n} \ell(y_i, \hat{f}_i), \tag{14}$$

where $\hat{f}_i$ is given by (5). There are two layers of iterations: the outer layer is the CC iteration and the inner layer is the boosting iterations. An early stop of iterations in boosting doesn't guarantee convergence. On the other hand, the output $\mathbf{f}^{(K)}$ may overfit the data. In this case, we may consider a two stage process: In the first stage, apply Algorithm 1 to obtain optimal weights of observations. In the second stage, we can use a data-driven method such as cross-validation to select optimal boosting iteration M, penalty numbers γ for trees, λ and α. The same strategy can also be applied to the robust parameter σ. However, since this parameter is typically considered a hyperparameter, a more computationally convenient approach in the literature is to conduct estimation for different values of σ and compare the results. One can begin with a large value σ with less robust estimation, and move towards smaller value σ for more robust results.

The source version of the **ccboost** package is freely available from the Comprehensive R Archive Network (http://CRAN.R-project.org). The reader can install the package directly from the R prompt via

```
R> install.packages("ccboost")
```

All analyses presented below are contained in a package vignette. The rendered output of the analyses is available by the R-command

```
R> library("ccboost")
R> vignette("ccbst", package = "ccboost")
```

To reproduce the analyses, one can invoke the R code

```
R> edit(vignette("ccbst", package = "ccboost"))
```

3. Data examples

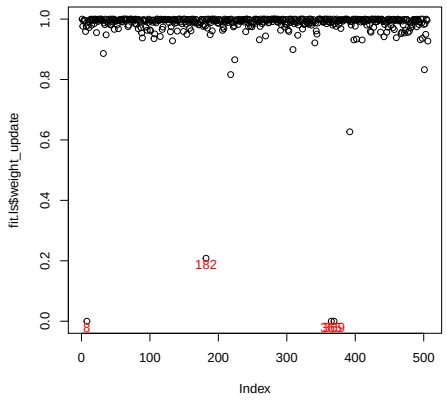
3.1. Robust boosting for regression

In this example, we predict median value of owner-occupied homes in suburbs of Boston, with data publicly available from the UCI machine learning data repository. There are 506 observations and 13 predictors. A different robust estimation can be found in Wang (2020).

```
R> urlname <- "https://archive.ics.uci.edu/ml/"
R> filename <- "machine-learning-databases/housing/housing.data"
R> dat <- read.table(paste0(urlname, filename), sep = "",
+      header = FALSE)
R> dat <- as.matrix(dat)
R> colnames(dat) <- c("CRIM", "ZN", "INDUS", "CHAS", "NOX",
+      "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B",
+      "LSTAT", "MEDV")
R> p <- dim(dat)[2]
```

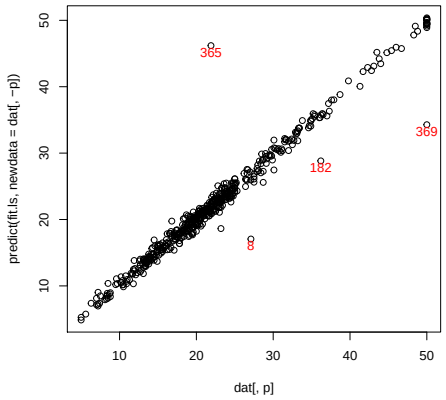
We fit a CC-boosting model with concave component `bcave` and convex component least squares. The observation weights are plotted. We also display the values of the 4 observations with the smallest weights. These observations are considered outliers. We can plot the original

```
R> library("ccboost")
R> fit.ls <- ccboost(dat[, -p], dat[, p], cfun = "bcave",
+      s = 10, dfun = "gaussian", verbose = 0, max.depth = 2,
+      nrounds = 50)
R> plot(fit.ls$weight_update)
R> id <- sort.list(fit.ls$weight_update)[1:4]
R> text(id, fit.ls$weight_update[id] - 0.02, id, col = "red")
```



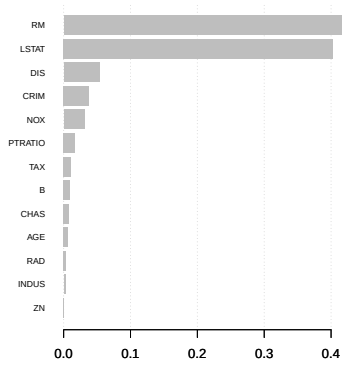
median housing price vs the predicted values. Not surprisingly, those 4 observations with the smallest weights have poor predictions. We can view feature importance/influence from the learned model. The figure shows that the top two factors to predict median housing price

```
R> plot(dat[, p], predict(fit.ls, newdata = dat[, -p]))
R> text(dat[id, p], predict(fit.ls, newdata = dat[id, -p]) -
+      1, id, col = "red")
```

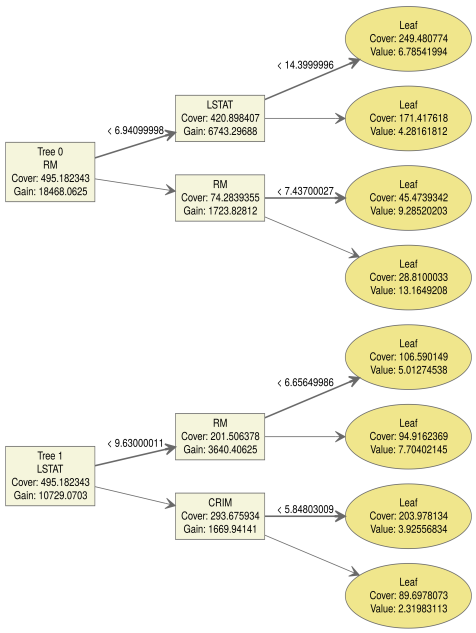


are average number of rooms per dwelling (RM) and percentage values of lower status of the population (LSTAT).

```
R> importance_matrix <- xgboost::xgb.importance(model = fit.ls)
R> xgboost::xgb.plot.importance(importance_matrix = importance_matrix)
```




```
R> gr <- xgboost::xgb.plot.tree(model = fit.ls, trees = 0:1)
```

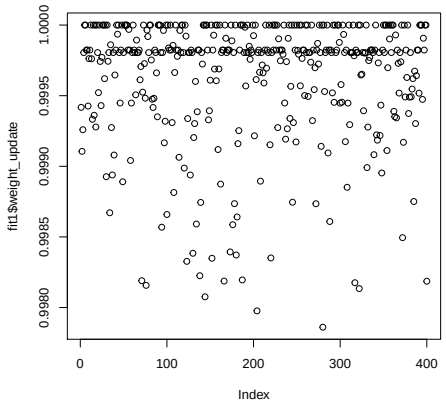

Tree 0
RM
Cover: 495.182343
Gain: 18468.0625
< 6.94099998
LSTAT
Cover: 420.898407
Gain: 6743.29688
< 14.3999996
Leaf
Cover: 249.480774
Value: 6.78541994
Leaf
Cover: 171.417618
Value: 4.28161812
RM
Cover: 74.2839355
Gain: 1723.82812
< 7.43700027
Leaf
Cover: 45.4739342
Value: 9.28520203
Leaf
Cover: 28.8100033
Value: 13.1649208
Tree 1
LSTAT
Cover: 495.182343
Gain: 10729.0703
< 9.63000011
RM
Cover: 201.506378
Gain: 3640.40625
< 6.65649986
Leaf
Cover: 106.590149
Value: 5.01274538
Leaf
Cover: 94.9162369
Value: 7.70402145
CRIM
Cover: 293.675934
Gain: 1669.94141
< 5.84803009
Leaf
Cover: 203.978134
Value: 3.92556834
Leaf
Cover: 89.6978073
Value: 2.31983113

3.2. Robust logistic boosting

A binary classification problem was proposed by Long and Servedio (2010). Response variable y is randomly chosen to be -1 or +1 with equal probability. We randomly generate symbols A, B and C with probability 0.25, 0.25 and 0.5, respectively. The predictor vector **x** with 21 elements is generated as follows. If A is obtained, $x_j = y, j = 1, ..., 21$. If B is generated, $x_j = y, i = 1, ..., 11, x_j = -y, j = 12, ..., 21$. If C is generated, $x_j = y$, where j is randomly chosen from 5 out of 1-11, and 6 out of 12-21. For the remaining $j \in (1, 21)$, $x_j = -y$. We generate training data $n = 400$ and test data $n = 200$.

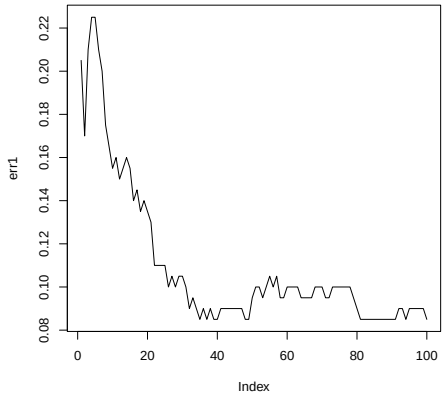
We fit a robust logistic boosting model with concave component `acave`, where the maximum depth of a tree is 5. Other concave components in Table 1 can be applied similarly. We can

```
R> set.seed(1947)
R> dat <- dataLS(ntr = 400, nte = 200, percon = 0)
R> fit1 <- ccboost(dat$xtr, dat$ytr, cfun = "acave", s = 3,
+      dfun = "binomial", verbose = 0, max.depth = 5, nrounds = 50)
R> plot(fit1$weight_update)
```

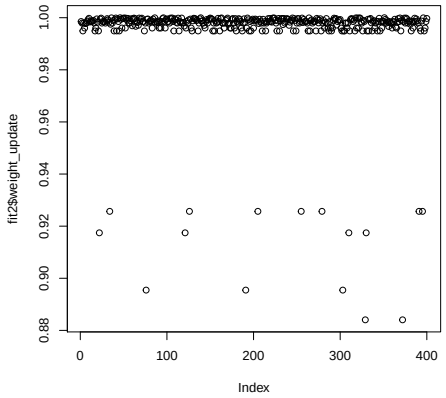


compute prediction error of test data at each boosting iteration. Furthermore, we simulate data with 10% contamination of response variables, and compute CC-boosting again. In the third robust logistic boosting, we reduce parameter value σ (`s` in the `ccboost` function) for more robust estimation. As a result, some observations would have decreased weights in the model.

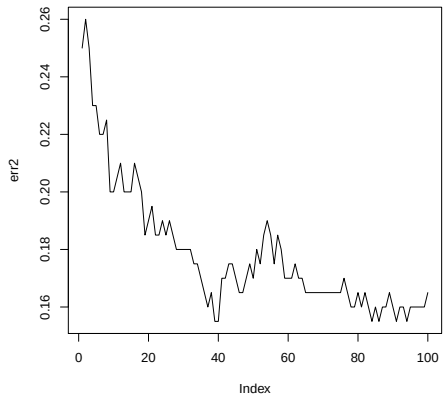
```
R> err1 <- rep(NA, 100)
R> for (i in 1:100) {
+      pred1 <- predict(fit1, newdata = dat$xte, ntreelimit = i)
+      err1[i] <- mean(sign(pred1) != dat$yte)
+  }
R> plot(err1, type = "l")
```



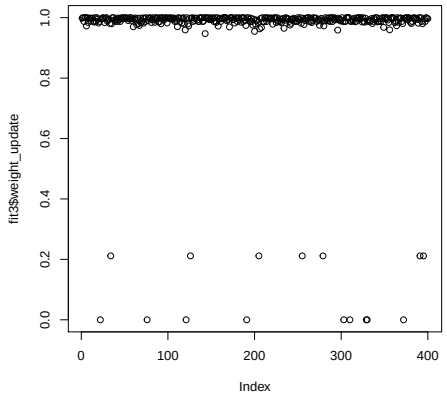

```
R> dat2 <- dataLS(ntr = 400, nte = 200, percon = 0.1)
R> fit2 <- ccboost(dat2$xtr, dat2$ytr, cfun = "acave", s = 3,
+      dfun = "binomial", verbose = 0, max.depth = 5, nrounds = 50)
R> plot(fit2$weight_update)
```



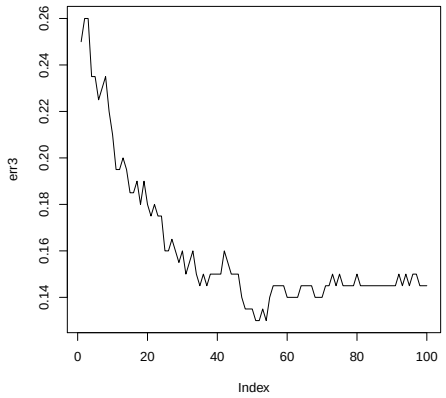
```
R> err2 <- rep(NA, 100)
R> for (i in 1:100) {
+      pred2 <- predict(fit2, newdata = dat2$xte, ntreelimit = i)
+      err2[i] <- mean(sign(pred2) != dat2$yte)
+  }
R> plot(err2, type = "l")
```




```
R> fit3 <- ccboost(dat2$xtr, dat2$ytr, cfun = "acave", s = 1,
+      dfun = "binomial", verbose = 0, max.depth = 5, nrounds = 50)
R> plot(fit3$weight_update)
```



```
R> err3 <- rep(NA, 100)
R> for (i in 1:100) {
+      pred3 <- predict(fit3, newdata = dat2$xte, ntreelimit = i)
+      err3[i] <- mean(sign(pred3) != dat2$yte)
+  }
R> plot(err3, type = "l")
```



3.3. Robust Poisson boosting

A survey collected from 3066 Americans was studied on health care utilization in lieu of doctor office visits (Heritier, Cantoni, Copt, and Victoria-Feser 2009). The data contained 24 risk factors. Robust Poisson regression was conducted in Wang (2020). Here robust Poisson boosting model is fitted with concave component `ccave`. The observation weights are estimated. The doctor office visits in two years are highlighted for the 8 smallest weights, ranging from 200 to 750. We can view feature importance/influence from the learned model. The

```
R> data(docvisits, package = "mpath")
R> x <- model.matrix(~age + factor(gender) + factor(race) +
+       factor(hispan) + factor(marital) + factor(arthri) +
+       factor(cancer) + factor(hipress) + factor(diabet) +
+       factor(lung) + factor(hearth) + factor(stroke) +
+       factor(psych) + factor(iadla) + factor(adlwa) + edyears +
+       feduc + meduc + log(income + 1) + factor(insur) +
+       0, data = docvisits)
R> fit.pos <- ccboost(x, docvisits$visits, cfun = "ccave",
+       s = 20, dfun = "poisson", verbose = 0, max.depth = 1,
+       nrounds = 50)
R> plot(fit.pos$weight_update)
R> id <- sort.list(fit.pos$weight_update)[1:8]
R> text(id, fit.pos$weight_update[id] - 0.02, docvisits$visits[id],
+       col = "red")
```

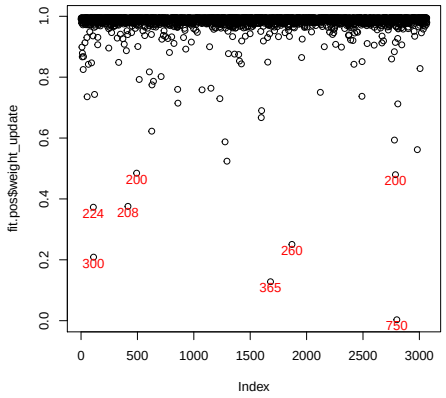
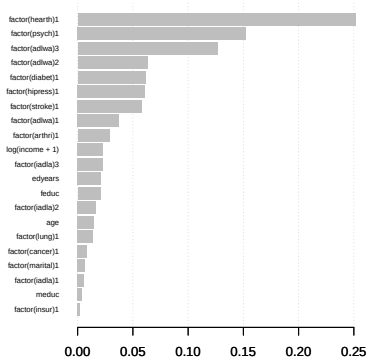


figure shows that the top two reasons of doctor office visits are heart disease and psychiatric problems.

4. Conclusion

In this article we propose CC-boosting as a unified robust boosting algorithm, and illustrate its applications in regression, classification and Poisson regression. The method can be used for outlier detection and can reduce the impact of outliers. Based on existing weighted boosting

```
R> importance_matrix <- xgboost::xgb.importance(model = fit.pos)
R> xgboost::xgb.plot.importance(importance_matrix = importance_matrix)
```



software, we can determine variable importance and explore the trees from the boosting algorithm. The R **ccboost** is a useful tool in the machine learning applications.

References

Bühlmann P, Hothorn T (2007). "Boosting algorithms: Regularization, prediction and model fitting (with discussion)." *Statistical Science*, **22**(4), 477–505.

Chen T, Guestrin C (2016). "Xgboost: A scalable tree boosting system." In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794.

Friedman J (2001). "Greedy function approximation: a gradient boosting machine." *Annals of Statistics*, **29**(5), 1189–1232.

Friedman J, Hastie T, Tibshirani R (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)." *Annals of Statistics*, **28**(2), 337–407.

Heritier S, Cantoni E, Copt S, Victoria-Feser MP (2009). *Robust Methods in Biostatistics*, volume 825. John Wiley & Sons.

Long PM, Servedio RA (2010). "Random classification noise defeats all convex potential boosters." *Machine learning*, **78**(3), 287–304.

Sigrist F (2020). "Gradient and Newton Boosting for Classification and Regression." *arXiv e-prints*. https://arxiv.org/abs/1808.03064, 1808.03064.

Wang Z (2018a). "Quadratic majorization for nonconvex loss with applications to the boosting algorithm." *Journal of Computational and Graphical Statistics*, **27**(3), 491–502.

Wang Z (2018b). "Robust boosting with truncated loss functions." *Electronic Journal of Statistics*, **12(1)**, 599–650. doi:10.1214/18-EJS1404. URL https://doi.org/10.1214/18-EJS1404.

Wang Z (2020). "Unified Robust Estimation via the COCO." *arXiv e-prints*, arXiv:2010.02848. https://arxiv.org/abs/2010.02848, 2010.02848.

Affiliation:

Zhu Wang
Department of Population Health Sciences
UT Health San Antonio
USA
E-mail: zhuwang@gmail.com