Stochastic Gradient Boosting with Squared Error Epsilon Insensitive Loss

Ken Lau
Statistics, UBC
ken.lau@stat.ubc.ca

Abstract

We often deal with outliers in air quality data. By fitting our model to all the data, we often put more emphasis on outliers. In this paper, we implement stochastic gradient boosting with a squared error epsilon insensitive loss function. The algorithm is carried out in R. We experimented on the Los Angeles Ozone pollution data [2] with multiple linear regression, boosting with least squares, and boosting with the new loss function. We found that both boosting methods to be more robust towards outliers than multiple linear regression. However, we cannot conclude that the new loss function is more robust than the traditional least squares loss.

1 Introduction

Although multiple linear regression is a quick and simple method for regression, it is not very robust to outliers (or extreme values) in data. The goal of this paper is to implement stochastic gradient boosting with a new robust loss function, in which we call squared error epsilon insensitive (SEEI) loss function. We also try to show its effectiveness towards fitting data with outliers.

In this paper, we deal with a regression problem with a continuous response variable and roughly 11 explanatory variables. Further description of the data can be found in Section 2 . Although it is difficult to visualize outliers with a feature space of 11 dimensions, we give two examples to try to indicate outliers in the data.

The stochastic gradient boosting algorithm is first developed by Friedman in 1998 [4] . It iteratively fits simple models and additively combines them to provide predictions. In this paper, we use regression trees as our simple models. We describe the algorithm in section 3.1 . The stochastic gradient boosting method uses a least squares loss function. The SEEI loss still uses least squares but any residuals falling within a pre-specified epsilon margin are not penalized. This allows the model to emphasize its fit towards the overall data than the extreme values. Section 4 provides the description of the loss function. We also show how SEEI can be implemented in R.

Finally we experimented on the Los Angeles Ozone Pollution data set [2] to compare multiple linear regression, boosted regression with least squares, and boosted regression with optimal values of epsilon.

2 Data

An example of a data set that contains outliers in the observations is the Los Angeles Ozone pollution data. There are 11 explanatory variables and 1 response variable. We have 330 complete observations, so essentially we have a 330 by 12 matrix of data we want to fit. See Table 1.

Explanatory Variables	Data Type
1. Month: 1 = January, ..., 12 = December	Discrete
2. Day of month	Discrete
3. Day of week: 1 = Monday, ..., 7 = Sunday	Discrete
4. 500 millibar pressure height (m) measured at Vandenberg AFB	Continuous
5. Wind speed (mph) at Los Angeles International Airport (LAX)	Continuous
6. Humidity (%) at LAX	Continuous
7. Temperature (degrees F) measured at Sandburg, CA	Continuous
8. Inversion base height (feet) at LAX	Continuous
9. Pressure gradient (mm Hg) from LAX to Daggett, CA	Continuous
10. Inversion base temperature (degrees F) at LAX	Continuous
11. Visibility (miles) measured at LAX	Continuous

Table 1: Explanatory Variables for Los Angeles Ozone Pollution [2]

The response variable of interest in our data set is daily maximum one-hour-average ozone reading (denoted as maximum O3). A kernel density plot of the maximum O3 response variable is shown in Figure 1 (right). From this plot, we could see the distribution of the response variable is skewed to the left. This could indicate that there are outliers on the upper quantile of maximum O3. We also fitted a multiple linear regression model to the full data. The predictions (or fitted values) are computed on each observation, and the observation versus prediction scatter plot is shown in Figure 1 (left). The plot shows that the linear regression fit seems to perform worse for observations in the upper quantiles. Therefore, we suspect that there are outliers in the upper quantiles of the data.

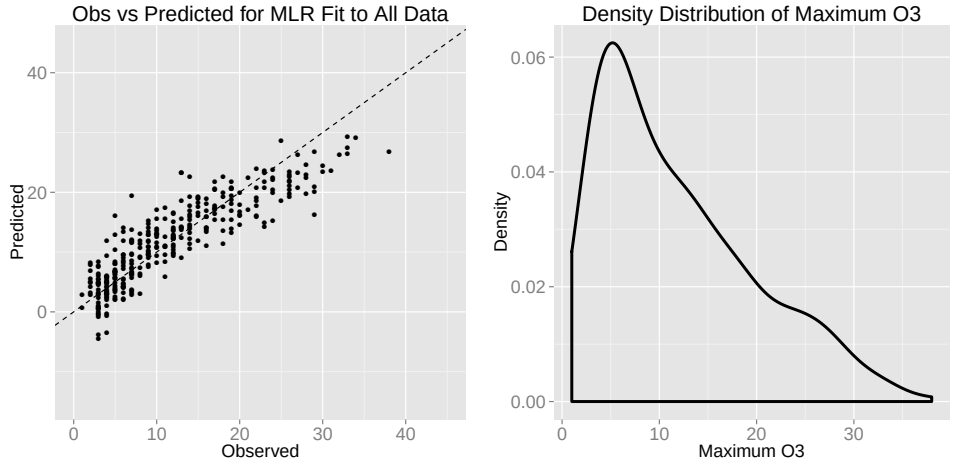


Figure 1: Left Plot: Observation versus Prediction Based on Fitting Multiple Linear Regression. Right Plot: Kernel Density Plot of Maximum O3 Response Variable

3 Introduction to Boosting

The technique of boosting improves the model performance by iteratively fitting simple models and combining them to provide predictions [3]. For example, we used regression trees for our simple models. At each iteration, the model gradually emphasizes the fit towards observations that are fit poorly to the current combination of trees. The fitted values are updated on each iteration based on the contribution of a newly fitted tree. We can take smaller steps at each iteration by updating only a fraction of the function estimates of the new tree. It usually improves the model fit, because it allows the model to correct itself on any additional tree which overfits the optimal estimation. Lastly, at each iteration, we should only fit a chunk (usually 50%) of the data to each tree. It has been shown

to reduce the variance of function estimates [4]. Therefore, predictions can be greatly improved by including this subsampling step for each iteration. Boosting is similar to Random Forest, except Random Forest uses a bagging type of method on regression trees. Its goal is to minimize variance of the predictions by averaging all the simple models. Whereas boosting updates predictions on each iteration. Both methods generally lead to better predictions upon fitting regression trees.

3.1 Stochastic Gradient Boosting Algorithm

We provide a description of the boosting algorithm as derived in [4] and implemented in package gbm which is written in R and C++ by Greg Ridgeway [5]. We first introduce a few input parameters. We denote T as the number of trees to fit in the overall model. Let K denote the fixed depth for each tree that is grown. We denote λ as the learning rate. We let Ψ be the loss function of interest. Finally, let the subsampling rate be p. The stochastic gradient boosting algorithm implemented in gbm is as follows.

Initialize current estimate, $\hat{f}(\mathbf{x})$ to be a constant, such that $\hat{f}(\mathbf{x}) = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} \Psi(y_i, \rho)$ where Ψ is a loss function of interest.

For each iteration, $t = 1, \ldots, T$,

1. Compute the negative gradient for each data point with the current estimate and denote by z.
$$z_i = \frac{-\partial}{\partial f(\mathbf{x_i})} \Psi(y_i, f(\mathbf{x_i})) \Big|_{f(\mathbf{x_i})=\hat{f}(\mathbf{x_i})}$$
2. Sample without replacement $p * N$ observations from the data, where N is the total number of observations and p is the subsampling rate between 0 and 1.
3. Using only the sampled data, fit a regression tree with a fixed depth K on the negative gradients calculated in step 1.
4. Compute the optimal terminal estimates at each node $(\rho_1, \ldots, \rho_{2^K})$ based on loss function Ψ.
$$\rho_k = \underset{\rho}{\operatorname{argmin}} \sum_{\mathbf{x_i} \in S_k} \Psi(y_i, \hat{f}(\mathbf{x_i}) + \rho)$$
where S_k is the set of $\mathbf{x}$ that define the terminal node k.
5. Update $\hat{f}(\mathbf{x})$ for all $\mathbf{x}$ by,
$$\hat{f}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda \rho_{k(\mathbf{x})}$$
where $k(\mathbf{x})$ is the index of the terminal node where $\mathbf{x}$ belongs to.

A similar description of this algorithm can be found in [5] .

3.2 Cross-Validation

In order to calculate the optimal number of trees in the boosting model, a cross validation method is implemented. The data is randomly partitioned into C slices. The gradient boosting algorithm is fit C times with each slice left out. The prediction mean squared error is calculated on the slice of data that is left out for each boosted model fit. The average cross-validation error is calculated on all slices, and the optimal number of trees is selected based on the lowest error.

4 Squared Error Epsilon Insensitive Loss Function (SEEI)

So far we haven't given a form for the loss function Ψ mentioned in the boosting algorithm. In most regression type problems, the least squares loss function is used. We implement a robust loss function called squared error epsilon insensitve abbreviated as SEEI. The method is more robust in that it tries to emphasize fits towards smaller residuals. It penalizes similarly to traditional least squares, except any squared residuals that are greater than ϵ are ignored in the penalty function. SEEI is defined as,

$$\Psi(x, y) = max\{0, (x - y)^2 - \epsilon\}$$

The loss function is shown in Figure 2

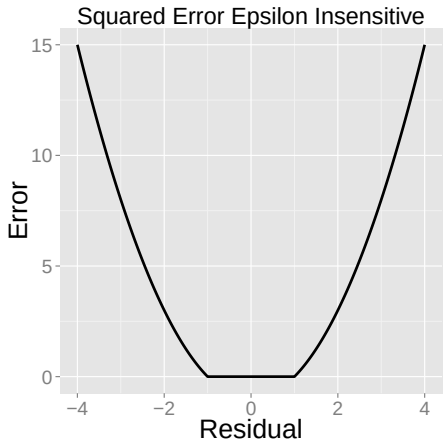


Figure 2: Squared Error Epsilon Insensitive Loss Function with $\epsilon = 1$

4.1 Convexity of Sum of Squared Errors Epsilon Insensitive

Similar to minimizing the residual sum of squared errors for most regression methods, our goal is to minimize the residual sum of squared errors with the epsilon insensitive loss. We can restrict our calculations to fit a constant linear model, because the terminal node estimates in the boosting algorithm only requires a constant fit.

Let $\mathbf{r} = (\mathbf{y} - \mathbf{1}\hat{\mu})$ and $\mathbf{D} = diag(\mathbb{1}((\mathbf{y} - \mathbf{1}\hat{\mu})^2 > \epsilon))$ where $\mathbf{r}$ is the vector of residuals, $\mathbf{y}$ is vector of response, and $\mathbf{1}$ is a vector of ones. The objective function, $\mathbf{S} = (\mathbf{r} \cdot \mathbf{D})^T(\mathbf{r} \cdot \mathbf{D})$ is what we seek to minimize in a regression model under the SEEI loss function. If we can show that $\mathbf{S}$ can be written in a quadratic form and the hessian is semi-positive definite, then we have convexity [6]. Since $\mathbf{D} \cdot \mathbf{D} = \mathbf{D}$, we minimize $\mathbf{S} = \mathbf{r}^T\mathbf{D}\mathbf{r}$. So, $\mathbf{S}$ is trivially a quadratic form. Now, if we expand $\mathbf{S}$, we get,

$$\mathbf{S} = \mathbf{y}^T\mathbf{D}\mathbf{y} - \mathbf{y}^T\mathbf{D}\mathbf{1}\mu - \mu\mathbf{1}^T\mathbf{D}\mathbf{y} + \mu^2\mathbf{1}^T\mathbf{D}\mathbf{1}$$

We now take the partial differentiation with respect to μ.

$$\frac{\partial}{\partial\mu}\mathbf{S}(\mu) = -2\mathbf{y}^T\mathbf{D}\mathbf{1} + 2\mu\mathbf{1}^T\mathbf{D}\mathbf{1}$$

Taking the second derivative of the objective function, we get,

$$\frac{\partial^2}{\partial\mu^2}\mathbf{S}(\mu) = 2\mathbf{1}^T\mathbf{D}\mathbf{1}$$

The hessian is therefore scalar and always greater or equal to 0. It is equal to 0 only if all squared residuals lie within the $\pm\epsilon$ margin. $\mathbf{S}$ is therefore almost always strictly convex, and any gradient descent type algorithm to solve for μ will eventually converge to a global minimum of $\mathbf{S}$.

4.2 Smoothness of Sum of Squared Error Epsilon Insensitive

Recall the squared error with epsilon insensitive (SEEI) loss function in Figure 2. The function appears to have a rough edge when the residual lies at ϵ. However, when we take the sum, we get a smoother function. Moreover, the boosting algorithm only requires a constant fit. An illustration of the smoothness is provided in Figure 3 . In the sum of absolute errors plot, there still remains a few rough edges especially near the optimal value. However, even with $\epsilon = 30$, the sum of squared residuals with epsilon insensitive remain very smooth compared to the original squared error loss.

The reason to mention the convexity and smoothness of the sum of SEEI loss is to show that the objective function, $\mathbf{S}$, can be solved by newton type methods, in particular, Newton-Raphson. This is relevant in the next section which describes how to implement the SEEI loss to the boosting algorithm.

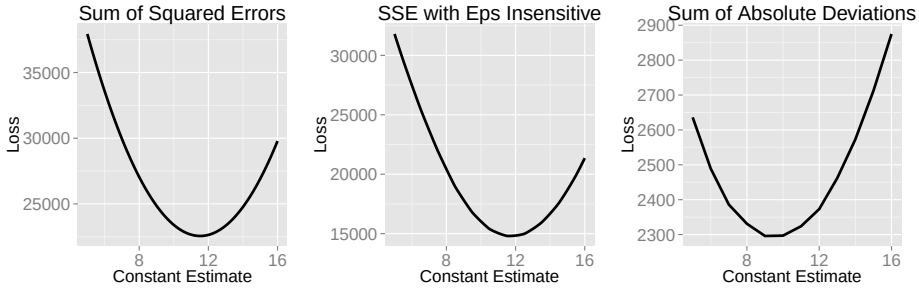


Figure 3: Comparison of the smoothness between loss functions on the residuals for Squared Error, Squared Error with Epsilon Insensitive ($\epsilon = 30$), and Absolute Error.

5 Implementing SEEI Loss into Boosting Alogirithm

5.1 Implementation

In part 4 of the gradient boosting algorithm, we need to solve the terminal node estimates for the fitted regression tree. For the traditional least squares loss function, the estimates are already solved by the regression tree. We need to modify the terminal node estimates based on the SEEI loss function. Note the terminal node estimates are simply constant fits.

At each new tree we fit, recall that we calculate,

$$\rho_k = \underset{\rho}{\operatorname{argmin}} \sum_{\mathbf{x_i} \in S_k} \Psi(y_i, \hat{f}(\mathbf{x_i}) + \rho)$$

for each terminal node of the tree where $\Psi(x, y) = max\{0, (x-y)^2 - \epsilon\}$, and $k = 1, \ldots, 2^K$ where K is the depth of the tree. Perform Newton-Raphson to find ρ_k for each terminal node:

For $k = 1, \ldots, 2^K$, iterate until ρ_k converges suffiently based on some tolerance level.

1. Since the fitted regression tree contains the mean estimate of each terminal node, set initial ρ_k to be the mean estimates of each respective terminal node.

2. Compute $\mathbf{D} = diag(\mathbb{1}((y - \rho_k)^2 > \epsilon))$

3. Compute $g = -2\mathbf{y}^T\mathbf{D1} + 2\mu\mathbf{1}^T\mathbf{D1}$ where g is the gradient of objective function $\sum \Psi$.

4. Compute $H = 2\mathbf{1}^T\mathbf{D1}$ where H is the hessian matrix of the objective function $\sum \Psi$.

5. Update $\rho_k \leftarrow \rho_k - H^{-1}g$

In our code, we set the tolerance level to be 10^{-2} on the absolute difference between ρ_k previous and ρ_k current. The next iteration of the boosting algorithm will continue to correct the error fitted by the past combinations of trees. So, it is possible to lower the tolerance to increase the speed of the algorithm. We can also set a maximum number of iterations.

5.2 Choosing Epsilon in SEEI by Cross-validation

Choosing the epsilon to use for the squared error epsilon insensitive loss function is similar to finding the optimal number of trees in the boosting algorithm. We can provide a short sequence of epsilon values, for example, 3, 6, 9, and 12. Each of these epsilon values are used in fitting boosting models using cross-validation to calculate prediction mean squared error based on least squares loss. The epsilon value yielding the smallest prediction mean squared error is chosen.

5.3 R Packages

We implemented the stochastic gradient boosting algorithm entirely in R. The code can be found from the following github link, https://github.com/kenlau177/cpsc546. The main file is called "main.R". We cleaned up the data by removing missing observations. Here we will acknowledge the use of a few packages in implementing SEEI loss in the boosting algorithm. Regression trees are fit by "rpart" [8] . Cross-validation is run in parallel using "snowfall" [7]. We used Hadley's vectorization and matrix reshaping packages, "plyr" [9] and "reshape2" [10] respectively. For plotting, we used "ggplot2" [11] .

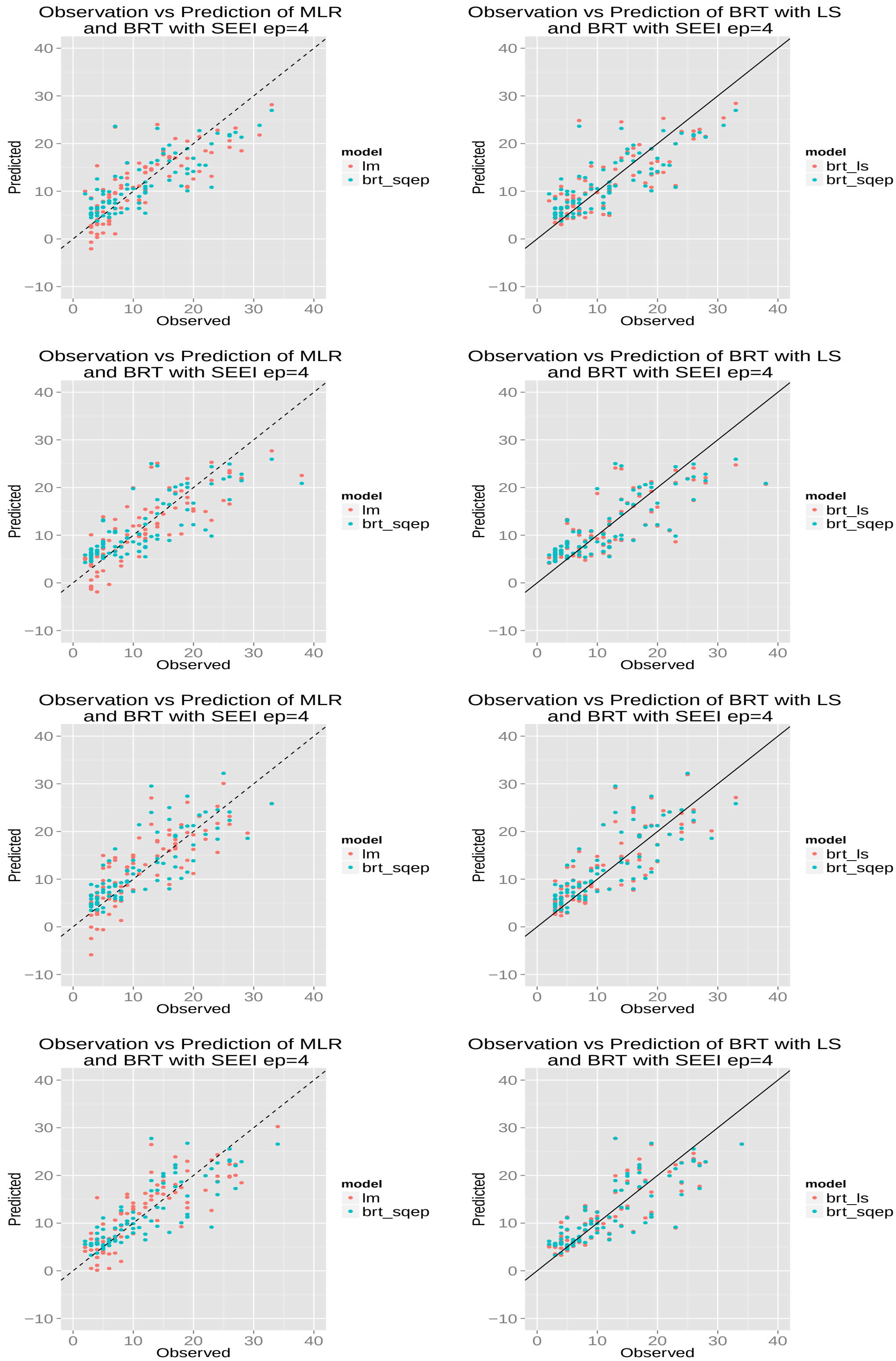
6 Experimentation and Results

6.1 Model Comparison

We used the Los Angeles Ozone Pollution data. The data is first cleaned up by removing missing observations. We then arbitrarily split the data 5 times into a training and testing set using 75% and 25% of the data respectively. Pseudo-random numbers were generated by R's set.seed function to retain the split for future use. We experimented on these 5 instances of the data. In addition to optimizing the epsilon in SEEI and number of trees, we also optimized for the fixed depth of the regression trees and learning rate λ. For each instance of the training data, cross-validation is performed and the combination of parameters of epsilon, number of trees, depth of tree, and learning rate giving the lowest cross-validation error based on traditional squared error is chosen. Using this optimal set of parameters, we fitted the full boosting model to the same training data. After the model is fitted, we used the testing data to make predictions. We compared the multiple linear regression, boosting with least squares, and boosting with the SEEI loss. Observation versus prediction scatter plots are provided in Figure 4 . We also calculated mean squared prediction error on certain portions of the data. We calculated the errors based on test observations greater than 75% quantile, test observations less than 60% quantile, and in between the 10% and 75% quantile of test observations. The results are provided in Table 2 . Recall from Section 2 on the data set, we expected the outliers are in the upper quantiles of the data. Therefore, we would be interested to find improvements from using SEEI loss for quantiles below 60%.

Combinations	**Model**	**Upper 75%**	**Lower 60%**	$> 10\%$ **and** $< 75\%$
Combination 1	MLR	671.22	843.68	949.36
Combination 1	BRT LS	593.55	713.87	889.3
Combination 1	BRT SEEI	616.36	755.47	890.652
Combination 2	MLR	547.07	627.58	862.77
Combination 2	BRT LS	751.13	483.12	866.95
Combination 2	BRT SEEI	664.01	507.93	905.94
Combination 3	MLR	450.62	968.05	1045.08
Combination 3	BRT LS	419.39	821.38	1076.72
Combination 3	BRT SEEI	486.48	889.36	1127.92
Combination 4	MLR	581.09	559.97	679.43
Combination 4	BRT LS	665.08	268.01	620.41
Combination 4	BRT SEEI	701.59	289.22	614.56
Combination 5	MLR	365.06	612.07	636.37
Combination 5	BRT LS	340.49	617.5	843.69
Combination 5	BRT SEEI	375.87	658.2	822.67

Table 2: Mean Squared Prediction Errors of Multiple Linear Regression (MLR), Boosted Regression with LS (BRT LS), and Boosted Regression with SEEI (BRT SEEI) on the upper 75%, lower 60%, and between 10% and 75% quantiles based on 5 combinations of the original data set split into training and testing sets.


Observation vs Prediction of MLR and BRT with SEEI ep=4
Predicted
Observed
model
lm
brt_sqep
Observation vs Prediction of BRT with LS and BRT with SEEI ep=4
Predicted
Observed
model
brt_ls
brt_sqep
Observation vs Prediction of MLR and BRT with SEEI ep=4
Observation vs Prediction of BRT with LS and BRT with SEEI ep=4
Observation vs Prediction of MLR and BRT with SEEI ep=4
Observation vs Prediction of BRT with LS and BRT with SEEI ep=4
Observation vs Prediction of MLR and BRT with SEEI ep=4
Observation vs Prediction of BRT with LS and BRT with SEEI ep=4

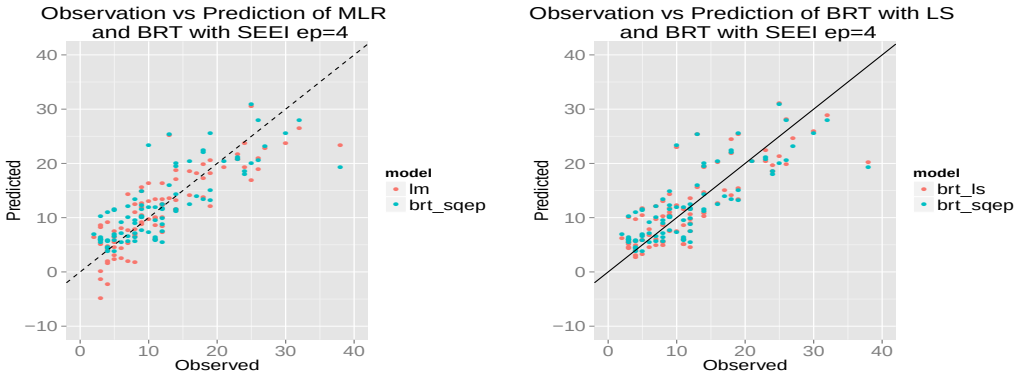


Figure 4: Observation versus Prediction Plots of model fits. The first row of plots correspond to the first randomization. The left plots are Multiple Linear Regression (lm) versus Boosting with SEEI (brt_sqep), and the right are Boosting with Least Squares (brt_ls) versus Boosting with SEEI (brt_sqep). The 45° line is where true observations match predictions

6.2 Results

The left plots of Figure 4 correspond to MLR against BRT with SEEI. For the upper quantiles on the test observations, the fits look similar for both the MLR and BRT with SEEI. However, for quantiles below 50%, the fits appear to have improved for BRT with SEEI over MLR. Most of the points for BRT with SEEI appear closer towards the 45° line which indicates a closer match between the true observations with predictions. From the plots, it appears the BRT with SEEI fitted better for lower quantiles whereas the BRT with LS fitted better for higher quantiles. However, the differences did not look significant based on the plots.

In Table 2, for all 5 combinations of the data and for the lower 60% quantile, both the BRT LS and BRT SEEI models had lower mean squared prediction errors (MSPE) than the MLR model. For example, in combination 1, we found the MSPE for MLR to be 843.58, and the MSPE for BRT with SEEI to be 755.47. This is 88.11 points improvement for BRT with SEEI. Whereas for quantiles greater than 75%, there does not appear to be any significant differences between the three models. So, the two BRT models were more robust to outliers and a better fit than the MLR model. However, We could not find any significant differences between BRT with LS and BRT with SEEI. For example, there are combinations of the data which gave lower MSPE for BRT with LS, and some combinations that gave lower MSPE for BRT with SEEI. So, we could not conclude that using SEEI for BRT was indeed a robust method over LS much less a better model than BRT with LS for all quantiles of the data. We could only conclude that both the BRT methods improved the fit for the test observations lower than the 60% quantile over multiple linear regression.

7 Conclusion

We described the gradient boosting regression method using the squared error epsilon insensitive loss function. We claimed that this method is more robust and could fit better towards data with outliers. We implemented the method in R, and we tested the method against multiple linear regression and boosting with least squares using the Los Angeles Ozone Pollution Data. We illustrated two examples to show that there appeared to be outliers in the data for upper quantiles of the ozone data. In our experiments, we were able to find both boosting methods improved the fits for quantiles of ozone data below 60%. However, no significant differences were found between the boosting method with least squares and squared error epsilon insensitive loss. We were able to show the boosting method is more robust than multiple linear regression, but we could not show that the new loss function is more robust than boosting with least squares.

References

[1] R Core Team (2014). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria. http://www.R-project.org/

[2] Leo Breiman (1985). *Estimating Optimal Transformations for Multiple Regression and Correlation.* Department of Statistics, UC Berkeley. JASA, 80, pp. 580-598.

[3] J. Elith, J.R. Leathwick and T. Hastie (2008). *A working guide to boosted regression trees.* Journal of Animal Ecology, 77, 802-813.

[4] J.H. Friedman (1998). *Stochastic Gradient Boosting.* Computational Statistics and Data Analysis, 38(4):367-378.

[5] Greg Ridgeway (2006). *Generalized Boosted Models: A guide to the gbm package.* R vignette.

[6] Stephen Boyd, Lieven Vandenberghe (2004). *Convex Optimization.* Department of Electrical Engineering, Stanford University. Cambridge University Press.

[7] Jochen Knaus (2013). *snowfall: Easier cluster computing (based on snow).* R package version 1.84-6. http://CRAN.R-project.org/package=snowfall.

[8] Terry Therneau, Beth Atkinson and Brian Ripley (2014). *rpart: Recursive Partitioning and Regression Trees.* R package version 4.1-6. http://CRAN.R-project.org/package=rpart

[9] Hadley Wickham (2011). *The Split-Apply-Combine Strategy for Data Analysis.* Journal of Statistical Software, 40(1), 1-29. URL http://www.jstatsoft.org/v40/i01/.

[10] Hadley Wickham (2007). *Reshaping Data with the reshape Package.* Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.

[11] Hadley Wickham (2009). *ggplot2: elegant graphics for data analysis.* Springer New York, 2009.