RegBoost: a gradient boosted multivariate regression algorithm

Wen Li, Wei Wang and Wenjun Huo
College of Electronics and Information Engineering,
Tongji University, Shanghai, China

Abstract
Purpose – Inspired by the basic idea of gradient boosting, this study aims to design a novel multivariate regression ensemble algorithm RegBoost by using multivariate linear regression as a weak predictor.

Design/methodology/approach – To achieve nonlinearity after combining all linear regression predictors, the training data is divided into two branches according to the prediction results using the current weak predictor. The linear regression modeling is recursively executed in two branches. In the test phase, test data is distributed to a specific branch to continue with the next weak predictor. The final result is the sum of all weak predictors across the entire path.

Findings – Through comparison experiments, it is found that the algorithm RegBoost can achieve similar performance to the gradient boosted decision tree (GBDT). The algorithm is very effective compared to linear regression.

Originality/value – This paper attempts to design a novel regression algorithm RegBoost with reference to GBDT. To the best of the knowledge, for the first time, RegBoost uses linear regression as a weak predictor, and combine with gradient boosting to build an ensemble algorithm.

Keywords Linear regression, Ensemble learning, Gradient boosting, RMSE

Paper type Research paper

1. Introduction
With the popularity of big data and cloud computing, more and more people are aware of the potential value of data. Machine learning, an important branch of artificial intelligence, which can learn from historical data and predict unknown data, has attracted a number of researchers’ attention in recent years. There are many branches of machine learning. Here, we only discuss supervised learning algorithms.

Gradient boosting is a common machine learning technique for regression and classification problems, which produces a final prediction model in the form of an ensemble of weak predictors. The idea of gradient boosting originated in the observation by Breiman (1997) and later developed by Jerome H. Friedman (2001, 2002). Gradient boosting optimizes a cost function over function space by iteratively choosing a function that points in the negative gradient direction. A common weak predictor for gradient boosting is the decision tree. Gradient boosted decision tree (GBDT) is widely used in Kaggle competitions and works incredibly well for many real-world problems. There are quite a lot of efficient and
effective implementations such as XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017) and CatBoost (Dorogush et al., 2017). GBDT iteratively constructs an ensemble of weak decision tree learners through boosting. The final prediction result of GBDT is obtained by adding the prediction results of all the trees.

The combination of gradient boosting and decision tree has been a great success. We are wondering whether it is feasible to use other algorithms as weak predictors. This paper considers the basic regression problem, which is to predict Y value given some relevant factors. Assuming we have a mass of historical data, the simplest regression model we can build linear regression. As we know, the final model of the gradient boosting technique is equal to the sum of all weak predictors. However, if we add multiple linear regression predictors directly, we will end up with a linear regression model. The algorithm proposed in this paper, RegBoost, divides the training data into two branches according to the prediction results using the current weak predictor. The linear regression modeling is recursively executed in two branches. In the test phase, test data is distributed to a specific branch to continue with the next weak predictor. The final result is the sum of all weak predictors across the entire path. Considering that the data may contain some features that are either redundant or irrelevant and can thus be removed without incurring much loss of information, RegBoost uses stepwise regression (Hocking, 1976) to select the most important factors when constructing each weak predictor. Our main contributions are:

- Inspired by the idea of gradient boosting, we design a novel regression algorithm RegBoost;
- We use stepwise regression to select features, which makes the algorithm more efficient and accurate; and
- We verify the effectiveness of our algorithm via comparative experiments on three public UCI data sets.

The rest of the paper is organized as follows: some related works are discussed in Section 2; then, Section 3 explains the main idea of our proposed algorithm RegBoost, and uses the pseudo code to formally describe the training and testing phases; Section 4 introduces comparative experiment on the public data set; and finally, we conclude the paper in Section 5.

2. Related work

In this section, we will discuss gradient boosted algorithms and stepwise regression. They are helpful for us to better understand our algorithm RegBoost.

2.1 Gradient boosted algorithms

Compared to the traditional machine learning algorithms, the principal difference of the boosting methods is that optimization is undertaken in the function space. Gradient boosting optimizes a cost function over function space by iteratively choosing a function that points in the negative gradient direction. As described in Friedman (2002), given training samples (x, y) of known values, our goal is to find a function \( F^*(x) \) that maps x to y, so that the target loss function is minimized.

\[
F^*(x) = \arg \min_{F(x)} E_{y|x} \varphi(y, F(x))
\]  

(1)

Stage-wise additive expansions of the weak predictor are used to search for optimal models. Boosting approximates \( F^*(x) \) by an “additive” expansion of the form.
\[ F(x) = \sum_{m=0}^{M} \beta_m h(x; a_m) \]  

(2)

where \( h \) (weak predictor) is usually a function of \( x \) with parameter \( a \). When constructing a weak predictor, the expansion coefficient \( \beta \) and the parameter can be calculated according to the following equation (3):

\[
(\beta_m, a_m) = \arg \min_{\beta, a} \sum_{i=1}^{N} \varphi \left( y_i, F_{m-1}(x_i) + \beta h(x_i; a) \right)
\]

(3)

We can adopt specific loss criteria for optimizing application objectives. Popular loss criteria are least-squares, least-absolute-deviation, Huber and logistic binomial log-likelihood. Our algorithm uses least-squares as loss function.

Gradient boosting is a strategy of combining weak predictors into a strong predictor. The algorithm designer can select the base learner according to specific applications. Many researchers have tried to combine gradient boosting with common machine learning algorithms to solve their problems. There is a great deal of gradient boosted algorithms in the literature (Liu et al., 2017), used MCNN as the base learner to estimate the density map of objects from single image with unknown perspective map (Hu et al., 2006), applied gradient boosting to HMMs based on the H-criterion (Bin et al., 2006), designed the continuous estimation of distribution algorithm based on boosting estimation of Gaussian mixture model (Zhang et al., 2016), proposed a gradient boosting random convolutional network framework for scene classification. In Dubossarsky et al. (2016), a new machine learning tool named wavelet-based gradient boosting was proposed and tested (Kenji and Kurita, 2005). Explored boosting soft-margin SVM with feature selection for pedestrian detection. Lots of researchers have proved the effectiveness of gradient boosting. However, the base learners mentioned above are all nonlinear models. They can be combined naturally by gradient boosting. The base learner of our algorithm RegBoost is linear regression, which is a linear model. We achieve nonlinearity by dividing data. It is a new method of constructing nonlinearity.

2.2 Stepwise regression

In machine learning and statistics, feature selection techniques are often used to avoid the curse of dimensionality and enhanced generalization by reducing overfitting. Stepwise regression (Hocking, 1976) is a method of fitting regression models in which the selection of independent variables is executed automatically. In each step, a variable is added or removed from the set of predictive variables based on the predefined criterion. This usually takes the form of a sequence of \( F \)-tests or \( t \)-tests, but other techniques are possible. Stepwise regression has three main approaches as follows: forward selection, backward elimination and bidirectional elimination.

- **Forward selection**: First, there is only one predictive variable in the model that explains most of the dependent variables. Then, choose the variable whose inclusion gives the most statistically significant improvement of the fit. Repeating the process until none can be added;

- **Backward elimination**: First, it starts with all independent variables. Delete the variable (if any) whose loss gives the most statistically insignificant deterioration of the model fit. Repeating the process until none can be deleted; and
Bidirectional elimination: If we adopt forward selection, each time a new predictive variable is added, the independent variable already existing in the model may be less explained to the dependent variable. When its contribution is not significant, we can remove it from the model. Bidirectional elimination does not always add the independent variable. It may delete variables too. In the end, we can get an optimal combination of variables.

3. Algorithm and methodology
3.1 Main idea
Inspired by the idea of gradient boosted algorithms, we try to design a regression model with linear regression as a weak predictor. To achieve nonlinearity after combining all linear regression predictors, the training data is divided into two branches according to the prediction results using the current weak predictor. The linear regression modeling is recursively executed in two branches. In the test phase, test data is distributed to a specific branch to continue with the next weak predictor. The final result is the sum of all weak predictors across the entire path.

First, we use all training data to obtain a basic linear regression predictor. For all training data, the predictor is used to obtain a predicted value. Then, we subtract the predicted value from the real value, classify all the training data with the result into two classes. If the residual is greater than or equal to 0 then we distribute it into “positive” class, otherwise, it is distributed to “negative” class. Then, we recursively executed the basic linear regression until the preset number of divisions is reached or the number of training instances is too small to continue the linear regression process.

In Figure 1, the letters A, B, C, D, E and F represent weak predictors. Using all the training data, we get weak predictor A. Then, we use weak predictor A to predict training data. With the predicted result, we can easily calculate the residual of every training instance.

Training data are divided into “positive” class and “negative” class according to the residual value. Here we assume that B (left) is “positive” class and E (right) is “negative” class. Weak predictor B is constructed via the training data that were previously divided into “positive” class. Similarly, predictor E is constructed via the training data that were previously divided into “negative” class. After the construction of weak predictors B and E, we need to continue dividing training data, respectively. E has no left node because few training instances from weak predictor E are classified as “positive” class. In this process, the training data is divided layer by layer.

We should note that when using the least-squares as the loss function the Y value of the subsequent weak predictor is the residual of previous weak predictors. To scatter the impact of all weak predictors in this process, we introduce the concept of the learning rate. Here is an example. Assume that the true value is Y, the prediction result of the first weak predictor

![Figure 1. A, B, C, D, E and F represent weak predictors](image-url)
is Ya and the learning rate is lr, then the target Y' value of the second weak predictor should be Y - Ya \times lr.

In GBDT, it is a natural process for test data to directly use the next decision tree as there is only one choice. However, in RegBoost the subsequent weak predictor is not unique. We need to choose, which branch to go according to the characteristics of the test data.

In the test phase, we need first use weak predictor A to obtain the predicted value Ya. Then, we calculate the K points closest to the current test data in the entire training data set, and count how many of the K nearest points are in the “positive” class (B) and how many are in the “negative” class (E). If most of the K points are in the “positive” class (B), we will distribute current test data into the “positive” class, and vice versa. Assuming that the current test data enter A, B and D in sequence, and the prediction results of the three weak predictors are Ya, Yb and Yd, the final prediction result of the test data is (Ya + Yb) \times lr + Yd. Note that the results obtained by the last weak predictor D should not be multiplied by the learning rate.

The core idea of RegBoost is described above. To avoid the curse of dimensionality and enhanced generalization by reducing overfitting, we also use stepwise regression to select the most important features first before constructing a weak predictor.

3.2 Formal description
We explained the main idea of RegBoost in the previous Section 3.1. Now, we present a formal pseudo code description of the training process and the testing process. Algorithm 1 is the training process in which Line 1 calls Algorithm 2. “max_layer” means maximal data division times, which can lead to early stopping. Algorithm 2 implements the construction of the entire model by recursively calling itself. Algorithm 3 is the test process that predicts unknown data based on the model obtained during the training process.

**Algorithm 1**: draining
Input: training_data (X, Y), max_layer
Output: model
1: model = get_jpredictor(trainX, train Y, max_layer)
2: return model

**Algorithm 2**: get_predictor
Input: training_data (X, Y), layer
Output: predictor
1: predictor = Null
2: if (layer <= 0) return predictor
3: features = stepwise_regression(candidate_features)
4: regr = linear_regression(X, Y, features)
5: Ypred = regr.predict(X, features)
6: (Xp, Xn, Yp, Yn) = divide_data(Y, Ypred)
7: (Yp’, Yn’) = modify_y(Yp, Yn) # Y = Y - learning_rate * Ypred
8: predictor_p = get_predictor(Xp, Yp’, layer-1)
9: predictor_n = get_predictor(Xn, Yn’, layer-1)
10: predictor = (regr, features, Xp, Xn, predictor_p, predictor_n)
11: return predictor

In Algorithm 2, Line 3 uses a stepwise regression method to select most important feature sets. Line 4 performs a linear regression on the training data based on the selected features. Line 5 uses the linear regression model just created to predict the training data. Line 6
divides the data into two parts based on the predicted value and the true value. Line 7 modifies the Y value because the target value of the next predictor should be different from the target value of the current predictor. Lines 8 and 9 call get_predictor recursively to build a complete model.

Algorithm 3 shows the complete test phase. We predict the unknown data based on the trained model. In Line 7, the “select_next” function counts how many of the K nearest points are in the “positive” class and how many are in the “negative” class. If more points are in the “positive” class, then “model_next” is equal to “model_p,” otherwise it is equal to “model_n.” The final prediction result of the model “adds” all selected predictors. Line 9 tells us how the returned result is calculated. Note that the last item of “preds” is not multiplied by the learning rate.

Algorithm 3: predict
Input: x, model
Output: result
1: preds = []
2: current_predictor = model
3: while (current_predictor is not null):
4: (regr, features, Xp, Xn, model_p, model_n) = current_predictor
5: ypred = regr.predict(x, features)
6: preds.add(ypred)
7: model_next = select_next(knn, features, Xp, Xn)
8: current_predictor = modelnext
9: result = sum(preds[:−1]) * learning_rate + preds[−1]
10: return result

3.3 Hyperparameter tuning
Now, we have a clear understanding of RegBoost. It is time to discuss some of the important hyperparameters in RegBoost and how to tune these hyperparameters to get the desired results. First, the two most important hyperparameters are the learning rate and the maximal number of layers (times of data division). When “max_layer” is small, we should set the learning rate to be larger, so that all weak predictors have a similar influence on the final result. We also need to pay attention to the fact that, as we have other termination conditions, that is, when the amount of data on a branch is less than a certain number, the division process is terminated in advance, the actual times of division may not equal “max_layer.” In addition, the number of selected features is also an important hyperparameter, which requires fine-tuning.

4. Experiments
4.1 Public data sets and selection of two comparative algorithms
We select three UCI machine learning data sets, which are CASP (archive.ics, 2019a), CCPP (archive.ics, 2019b) and SuperConduct (archive.ics, 2019c). These three data sets come from different fields and can be used for the regression algorithm. None of the three data sets have missing values:

(1) **CASP:** this data set contains a total of 45,739 data instances, each with 9 features. The learning goal of this data set is to predict the RMSD value of the three-dimensional structure of the protein by using the total surface area of the protein, the exposed area of the non-polar residue, the spatial distribution constraints, etc.
(2) **CCPP**: this data set contains 9,568 data points collected over 6 years from a combined cycle power plant, each containing 4 features. The learning goal of this data set is to predict the net energy output per hour based on ambient characteristics such as ambient temperature, pressure and relative humidity (Tufekci, 2014). Applied REPTree to this data set and achieved good results. In Kaya et al. (2012), the relationship between the four features and the target value of the data set is discussed in detail, and the experimental results of various machine learning algorithms and their combinations are compared and analyzed.

(3) **SuperConduct**: this data set extracts 81 features from 21,263 superconductors and is designed to use these features to predict the critical temperature of a superconductor.

To verify the effectiveness of RegBoost, we choose two regression algorithms to do comparative experiments. Multiple linear regression is used as a benchmark because it is the base learner of our algorithm RegBoost. If the performance of RegBoost is better than multiple linear regression, we may believe gradient boosting really helps. The other comparison algorithm is GBDT, as it is the most popular and widely used gradient boosted algorithm. Besides, in regression problems, both RegBoost and GBDT uses least-squares as loss criteria and learn subsequent base predictor via residuals. LightGBM (Ke et al., 2017) is an efficient implementation of GBDT. So we choose it to do experiments.

### 4.2 Evaluation

Root mean square error (RMSE) (Pontius et al., 2008; Willmott and Matsuura, 2006), also well-known as the standard error, is the square root of the average of squared errors. RMSE is frequently used to measure the deviation between the predicted value and the true value. The calculation formula is shown in equation (4).

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}} \tag{4}
\]

Mean absolute error (MAE) (Willmott and Matsuura, 2005) is the average of the absolute error. MAE has a clear interpretation as the average absolute difference between two continuous variables. Its formula is shown in equation (5). Both RMSE and MAE are widely used in the performance evaluation of regression algorithms.

\[
MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{5}
\]

### 4.3 Results

As mentioned above, we choose multiple linear regression, LightGBM to do comparative experiments. Below are the RMSE (Table I) and MAE (Table II) results of the three UCI data sets using multiple linear regression, LightGBM and RegBoost.

| Algorithms      | CASP  | CCPP  | SuperConduct |
|-----------------|-------|-------|--------------|
| Linear regression | 5.22  | 4.49  | 17.61        |
| LightGBM        | 4.71  | 3.78  | 12.26        |
| RegBoost        | 4.49  | 3.88  | 13.07        |
Based on Tables I and II, we plot histograms Figures 2 and 3. Figure 2 shows the RMSE values of the three algorithms on different data sets. From the histograms, we can see clearly that RegBoost performs best in the CASP data set. In the CCPP and SuperConduct data sets, although LightGBM is better than RegBoost, the difference is slight. Figure 3 shows the MAE values of the three algorithms on different data sets. It can be seen that in the CASP data set, RegBoost performs much better than LightGBM. In the CCPP and SuperConduct data sets, Linear regression performs best.

| Algorithms      | CASP | CCPP | SuperConduct |
|-----------------|-----|------|--------------|
| Linear regression | 4.38 | 3.73 | 13.55        |
| LightGBM        | 3.31 | 2.94 | 7.61         |
| RegBoost        | 2.53 | 2.75 | 7.56         |

Figure 2. RMSE values

Figure 3. MAE values

Table II. MAE of three algorithms on different data sets
data sets, the MAE of RegBoost is also slightly lower than that of LightGBM. In general, RegBoost and LightGBM behave similarly, both of which are far superior to linear regression.

Compared with multivariate linear regression, RegBoost reduced RMSE value by 13.98 per cent (CASP), 13.58 per cent (CCPP) and 25.78 per cent (SuperConduct), respectively. As for MAE values, RegBoost performs best among these three algorithms. RegBoost reduced MAE values by 23.56 per cent (CASP), 6.46 per cent (CCPP) and 0.65 per cent (SuperConduct) compared to LightGBM.

To better compare the actual performance of the three algorithms, we take the CASP data set as an example and randomly select 739 data instances as test data. We use three models to predict the test data, and then calculate the residuals based on the actual values and predicted values. Figure 4 plots the residuals obtained by linear regression, where the $x$-axis
is the real Y value and the y-axis is the residual. Besides, we plot to scatter plots Figures 5, 6 and 7. In each of the three figures, each point is a test data, the x-axis is the real Y value, and the y-axis is square of the residual. From Figures 5, 6 and 7, compared with Figure 5 (linear regression), most of the points in Figure 6 (LightGBM) and Figure 7 (RegBoost) are closer to the x-axis, which indicates the squared error of LightGBM and RegBoost is smaller.

The number of weak predictors in RegBoost has a crucial impact on performance. Taking CASP as an example, we record the RMSE and MAE values of RegBoost under a different number of weak predictors, and we obtain Figures 8 and 9.

It can be clearly seen from Figure 8 that as the number of weak predictors increases, the RMSE continues to decrease. When the number of weak predictors is less than 6, the RMSE value decreases sharply with the increasement of “max_layer.” Similarly, Figure 9 shows that the MAE value also decreases as the number of weak predictors continues to increase.

Figure 6. Square of residual (LightGBM)

Figure 7. Square of residual (RegBoost)
The training data is continuously divided so that the training data available to the subsequent predictors decrease exponentially. The number of weak predictors can never be set to a very large value due to limited training data.

5. Conclusion and future explorations
Inspired by the idea of gradient boosting, this paper attempts to design a novel regression algorithm RegBoost with reference to GBDT. To the best of our knowledge, for the first time, RegBoost uses linear regression as a weak predictor, and combine with gradient boosting to build an ensemble algorithm. Through comparison experiments, it is found that our algorithm RegBoost can achieve similar performance to GBDT. However, RegBoost divides the training data recursively during the sequential construction of the weak
predictors, resulting in an exponential decrease in the training data of the subsequent weak predictors. Therefore, RegBoost is currently not suitable for applications with too few data. If we can find other ways to make the number of weak predictors unrestricted to training data in the future, we may get better algorithms. Gradient boosting is a great way to build ensemble model. It deserves further study. In general, our algorithm is very effective compared to linear regression.

References

archive.ics (2019a), available at: https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure

archive.ics (2019b), available at: https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant

archive.ics (2019c), available at: https://archive.ics.uci.edu/ml/datasets/Superconductivity+Data

Bin, L., X-J., Wang, R-T. and Zhong Z., (2006), “Continuous optimization based-on boosting Gaussian mixture model”, 18th International Conference on Pattern Recognition (ICPR’06), Hong Kong, 2006, pp. 1192-1195.

Breiman, L.A. (1997), The Edge. Technical Report 486, Statistics Department, University of CA, Berkeley.

Chen, T. and Guestrin, C. (2016), “XGBoost: a scalable tree boosting system”, in Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 785-794.

Dorogush, A.V., Vasily, E. and Andrey, G. (2017), CatBoost: gradient Boosting with Categorical Features Support, Workshop on ML Systems at NIPS.

Dubossarsky, E., Friedman, J.H., Ormerod, J.T. and Wand, M.P. (2016), “Wavelet-based gradient boosting”, Statistics and Computing, Vol. 26 Nos 1/2, pp. 93-105.

Friedman, J.H. (2001), “Greedy function approximation: a gradient boosting machine”, The Annals of Statistics, Vol. 29 No. 5, pp. 1189-1232.

Friedman, J.H. (2002), “Stochastic gradient boosting”, Computational Statistics and Data Analysis, Vol. 38 No. 4, pp. 367-378.

Hocking, R.R. (1976), “The analysis and selection of variables in linear regression”, Biometrics, p. 32.

Hu, R., Li, X. and Zhao, Y. (2006), “Gradient boosting learning of hidden Markov models”, 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, p. 1.

Kaya, H., Tufekci, P. and Sadik, F.G. (2012), “Local and global learning methods for predicting power of a combined gas and steam turbine”, Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE, (March. 2012, Dubai), pp. 13-18

Ke, G., Meng, Q., Finley, T., Wang T., Chen, W., Ma, W. Ye Q. and Liu, T.-Y. (2017), “LightGBM: a highly efficient gradient boosting decision tree”, 31st Conference on Neural Information Processing Systems, Long Beach, CA.

Kenji, N. and Kurita, T. (2005), “Boosting Soft-Margin SVM with feature selection for pedestrian detection”, International Workshop on Multiple Classifier Systems Springer Berlin Heidelberg.

Liu, X., Campbell, D. and Guo, Z. (2017), “Single image density map estimation based on multi-column CNN and boosting”, IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, pp. 1393-1396.

Pontius, R., Thonette, O. and Chen, H. (2008), “Components of information for multiple resolution comparison between maps that share a real variable”, Environmental and Ecological Statistics, Vol. 15 No. 2, pp. 111-142, doi: 10.1007/s10651-007-0043-y.
Tufekci, P. (2014), “Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods”, *International Journal of Electrical Power and Energy Systems*, Vol. 60, pp. 126-140, ISSN 0142-0615.

Willmott, C.J. and Matsuura, K. (2005), “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance”, *Climate Research*, Vol. 30, pp. 79-82.

Willmott, C. and Matsuura, K. (2006), “On the use of dimensioned measures of error to evaluate the performance of spatial interpolators”, *International Journal of Geographical Information Science*, Vol. 20 No. 1, pp. 89-102.

Zhang, F., Du, B. and Zhang, L. (2016), “Scene classification via a gradient boosting random convolutional network framework”, in *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 54 No. 3, pp. 1793-1802.

**Corresponding author**

Wen Li can be contacted at: liwen00812@qq.com

For instructions on how to order reprints of this article, please visit our website: [www.emeraldpublishing.com/licensing/reprints.htm](http://www.emeraldpublishing.com/licensing/reprints.htm)

Or contact us for further details: permissions@emeraldinsight.com