Ensemble Learning&
Homework Chanllenge:linear boosting regression

Presented by
Jingfang Liu (15320171151900)
&
Xingyue Yu (15320171151888)

Department of Economics, School of Economics,
Xiamen University

Keywords
Ensemble learning/Decision tree/Bagging/Boosting/Random Forests

Contents

◊ Introduction

In machine learning, ensemble learning combines many individual estimators to improve the robustness and generalization ability of the overall model.

There are two main categories of integration methods:

1. **Averaging:** the construction of multiple independent individual models, with the Averaging of their predicted values as the final prediction of the integrated model. Variance will be reduced.
   Example: Bagging, Random Forests.

2. **Boosting:** order to establish a series of weak model, each weak model to reduce the bias, resulting in a strong model.
   Example: AdaBoost, Gradient Boosting.

◊ Averaging

Motivation

The decision tree can automatically detect nonlinear relationships and model the interaction, though we admit it has highly ability of interpretation, the poor predictive performance can not be ignored.

For decision tree, once a split is made, all the splits under it are changed as well. So a small change in the observed data often results in a completely different tree. Due to this hierarchical nature, the trees generally suffer from **high variance** and are **unstable**.

To improve the method, one of the ideas is aim to degree the variance. **Averaging** is a general-purpose procedure for reducing the variance of a statistical learning method.

Two learning methods

1. Bagging

(1) Conception

Given a set of N independent random variables. i.e. $\{x_1, \dots, x_N\}$ are i.i.d.
Let b=1, . . . ,B index the bootstrap samples drawn from the training data. We fit our model on each bootstrap sample.

When making a prediction, each learner makes a prediction, and the overall result is the average of each prediction.

For regression problems, let $\hat{f}^{(b)}(x)$ be the model's prediction trained on sample b, We then average all the predictions to obtain

$$\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{(b)}(x)$$

For classification problems, let $\widehat{p_J}^{(b)}(x)$ be the class probabilities estimated on bootstrap sample b. We have:

$$\hat{f}(x) = argmax_{j\in\{1,\dots,J\}}\widehat{p_J}(x)$$

Where

$$\widehat{p_J}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{p_J}^{(b)}(x)$$

Bagging is also called Bootstrap aggregating. In this method, the individual learner is treated as a black box without further modification, so its individual learner can be any machine learning algorithm. It randomly sampled the training data set and used the sampled data subset to train each individual learner.

(2) Tree v.s bagging

In machine learning, error can be broken down into bias, variance and noise.

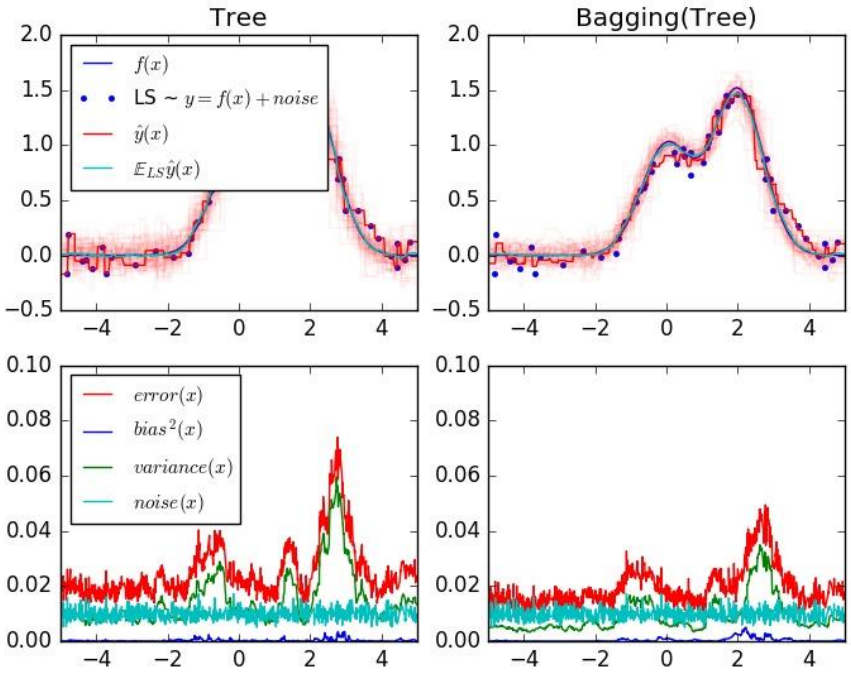

Figure 1

As shown in the figure 1 above, under the situation of same noise, the bias of both single decision tree and bagging (decision tree) is very low, but bagging can significantly reduce variance of the model.

(3) Code Implementation

1) sklearn.ensemble.BaggingClassifier:
   https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
2) sklearn.ensemble.BaggingRegression:
   https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html

(4) Pros and cons

1) Pros
   Bagging can dramatically reduce the variance and stabilize the predictions of unstable procedures.
   Bagging will reduce variance in the model, so it will work against over fitting
2) Cons
   Loss of interpretability. Bagging improves prediction accuracy at the expense of interpretability.

Bagging (bootstrap aggregating) regression trees is a technique that can turn a single tree model with high variance and poor predictive power into a fairly accurate prediction function. Unfortunately, bagging regression trees typically suffers from tree correlation, which reduces the overall performance of the model.

2. Random forests

Random forests are a modification of bagging that builds a large collection of de-correlated trees and have become a very popular "out-of-the-box" learning

algorithm that enjoys good predictive performance.

(1) Basic idea

As in bagging, random forests build a number of trees on bootstrapped samples. In the growth process of each random tree, when nodes split, the features used are **no longer** the best among **all** features, but the best in a subset of features. This kind of randomness reduces the amount of calculation and slightly increases bias, greatly increasing the diversity of the decision tree, and will greatly reduce variance after taking the average.

On the whole, random forest is a very powerful algorithm.

(2) The random forest had several variants, including **the extremely randomized trees** (Extra Trees).On the basis of stochastic forests, **Extra Trees** puts some randomness into every threshold for node splitting. For any single feature, Extra Trees randomly selects some threshold and selects the best threshold as the classification point. Compared with random forests, variance in Extra Trees is lower and bias higher.

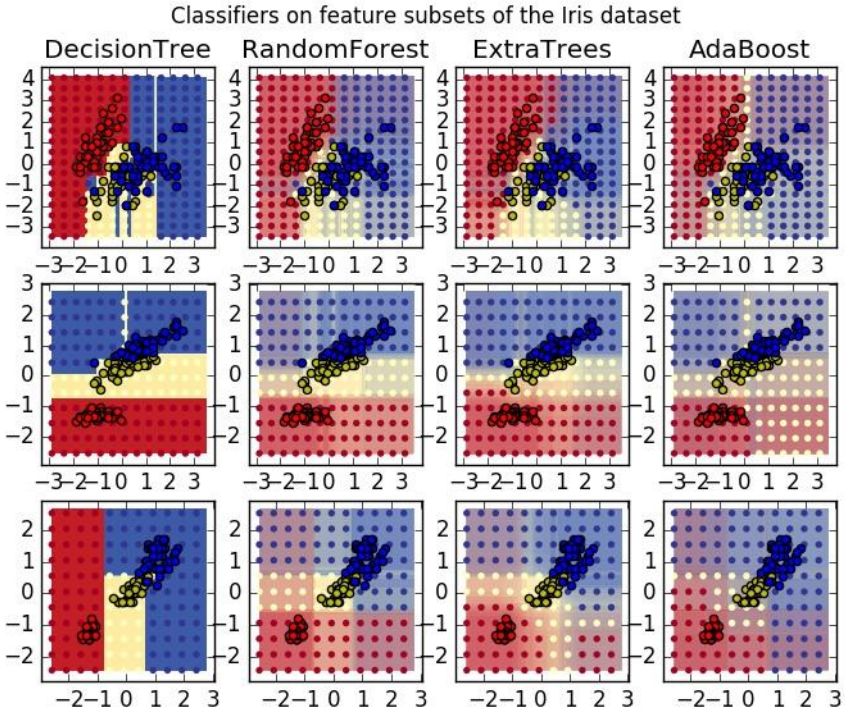

Figure 2

It can be seen from figure 2 that the confidence of single decision tree prediction is the highest, but it is easy to over fit; **Extra Trees** predicts slightly lower confidence than random forests, but it also has smoother decision boundaries.

(3) Code Implementation

1) sklearn.ensemble.RandomForestClassifier: https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
2) sklearn.ensemble.RandomForestRegressor: https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

(4) Pros and cons

1) Pros

Improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much.

2) Cons

Loss of interpretability.

Improve the independence at the expense of increasing the variance on each learner.

◊ Boosting

1. The introduction of boost

Boosted regression is a recent data mining technique that has shown considerable success in predictive accuracy.

(1) Boosting vesus traditional linear model:

In economics, a model built from a theory and data used to estimate parameters. Boosting is a highly flexible regression method. It allows the researcher to specify the x-variables without specifying the functional relationship to the response. Because of the flexibility, a boosted model will tend to fit better than a linear model and therefore inferences made based on the model may have more credibility.

(2) Boosting was invented by computational learning theorists and later reinterpreted and generalized by statisticians and machine learning researchers. Computer scientists tend to think of boosting as an "ensemble" method (a weighted average of predictions of individual classifiers), whereas statisticians tend to think of boosting as a sequential regression method.

2. Common methods

(1) The "AdaBoost" algorithm

Adaboost works only in the case where the response variable takes only one of two values: (-1/1,0/1).

Let *C*1 be a binary classifier (e.g. logistic regression) that predicts whether an observation belongs to the class "0" or "1".The classifier is fit to the data as usual and the misclassification rate is computed. This first classifier *C*1 receives a classifier weight that is a monotone function of the error rate it attains. Classifier weights there are also observation weights.

For the first classifier, all observations were weighted equally. The second classifier, C2 (the second logistic regression), is fit to the same data, however with changed observation weights. Observation weights corresponding to observations misclassified by the previous classifier are increased. Again, observations are reweighted, a third classifier C3 (a third logistic regression) is fit and so forth.

Altogether *iter* classifiers are fit where *iter* is some predetermined constant. Finally, using the classifier weights the classifications of the individual classifiers are combined by taking a weighted majority vote.

The procedure:

Initialize weights to be equal $wi = 1/n$, for $m = 1$ to *iter* classifiers *Cm*:

(a) Fit classifier Cm to the weighted data

(b) Compute the (weighted) misclassification rate rm

(c) Let the classifier weight $\alpha m = \log ((1 – rm)/rm)$

(d) Recalculate weights wi=wi exp(αm I(yi ≠ Cm))

In this method, observations that are repeatedly misclassified are given successively larger weights and the final "boosted" classifier consists of a majority-weighted vote of all previous classifiers.

(2) Regression trees

The most commonly used simple classifier is a regression tree. A regression tree partitions the space of input variables into rectangles and then fits a constant to each rectangle. Each observation descends the set of if-then conditions until a leaf is reached. If the if-then condition is true then the observation goes to the right branch otherwise to the left branch. A regression tree with only two terminal nodes (a tree stump) working surprisingly well in boosting.

Boosting with stumps fits an additive model but harbors a greater danger of overfitting.

(3) Friedman's gradient boosting algorithm

Then Friedman et al. (2000) were able to reinterpret this algorithm in a likelihood framework, enabling the authors to form a boosted logistic regression algorithm. Once the connection to the likelihood existed, boosting could be extended to generalized linear models.

The interpretation of boosting in terms of regression for a continuous, normally distributed response variable is as follows.

The average y-value is used as a first guess for predicting all observations. The residuals from the model are computed. A regression tree is fit to the residuals. For each terminal node, the average y-value of the residuals that the node contains is computed. The regression tree is used to predict the residuals.

The boosting regression model - consisting of the sum of all previous regression trees-is updated to reflect the current regression tree. The residuals are updated to reflect the changes in the boosting regression model; a tree is fit to the new residuals.

The procedure:

1) Initialization: Set initial guess to y

2) For all regressions trees m=1 to M:

(a) Compute the residuals based on the current model: $r_{mi} = y_i - f_{m-1}(x_i)$,where i indexes observations. Note that f_{m-1} refers to the sum of all previous regression trees.

(b) Fit a regression tree (with a fixed number of nodes) to the residuals

(c) For each terminal node of the tree, compute the average residual. The average value is the estimate for residuals that fall in the corresponding node.

(d) Add the regression tree of the residuals to the current best fit

$f_m = f_{m-1}$ + last regression tree of residuals

3. An example to compare boosted Gaussion regression and linear regression

(1) linear model set

$$y = 5 + (x_1 - 0.5)^2 + 2x_2^{-0.5} + x_3 + \varepsilon$$

Where $\varepsilon \sim uniform(0, 1)$

(2) the number of interactions

The actual number of iterations that maximizes the likelihood, *bestiter*, varies.

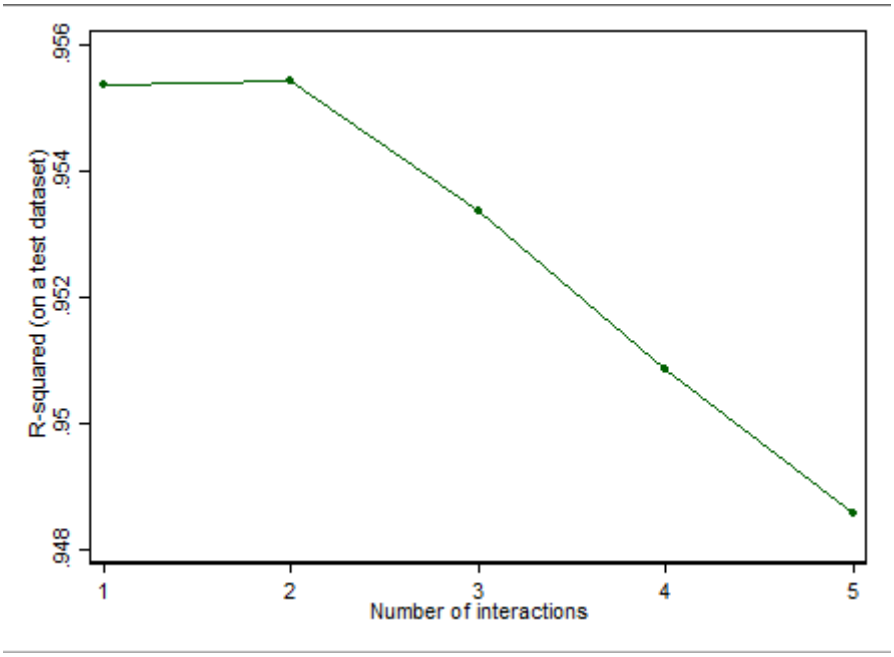

Figure 3

The test R2 is roughly the same regardless of the number of interactions, from the above graph,we set the interaction = 1.

(3) compasion of linear regressions and boosting

The boosted regression R2 refers is computed on a test data set, whereas the linear regression R2 is computed on the training data set. Thus we lead to pseudo R2:

$$preudoR^2 = 1 - L1 / L0$$

where L1 and L0 are the log likelihoods of the model under consideration and an intercept-only model, respectively.

In the case of Gaussian (normal) regression, the pseudo R2 turns into the familiar R2 that can be interpreted as "fraction of variance explained by the model".

For Gaussian regression it is sometimes convenient to compute R2 as

$$R2 = \frac{Var(y) - MSE(\mathrm{y}, y)}{Var(y)}$$

where Var(.) and MSE() refer to variance and mean squared error respectively

The training R2 and test R2 are computed on training and test data, respectively.

The training R2 is always between 0 and 1. Usually, the test R2 is also between 0 and 1.However, if the log likelihood based on the intercept-only model is greater than the one for the model under consideration the test R2 is negative. A negative test R2 is a sign that the model is strongly over fit.

Using preudoR2,it is possible to compute an R2 on a test data set for the linear regression.

The test R2 will usually be lower than the R2 in the output that Stata displays – but because of variability it may be larger on occasion. the training R2 value is R2= 43.2%. The test R2 =32.1 %.

Substituting the boosting predictions for the linear regression predictions in the above set of Stata commands the test boosting R2 turns out to be R2= 95.5%.

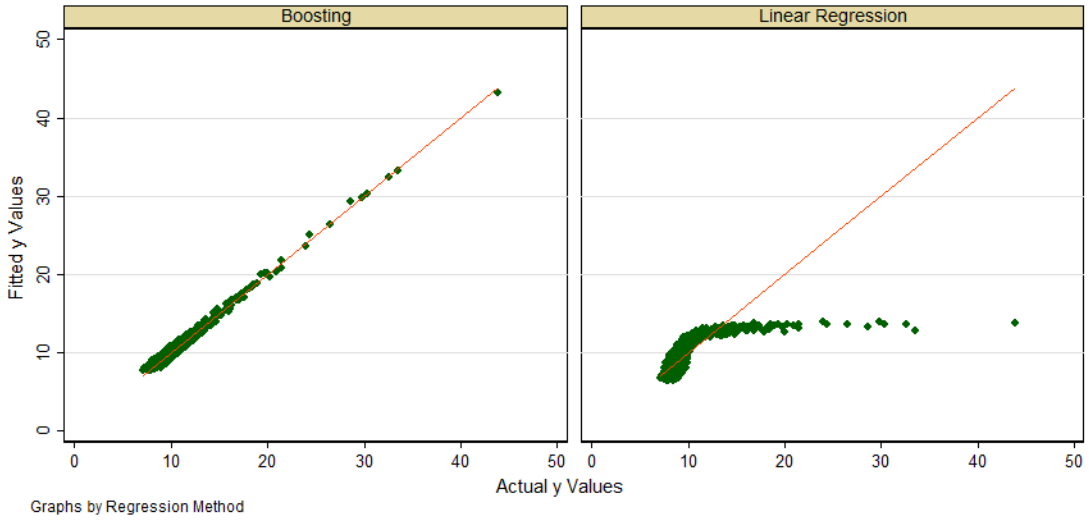

Figure 4

Figure 3 displays actual versus fitted y-values for both linear regression and boosting. The boosting model fits much better.

◊ Acknowlegement

Part of this summary is adapted from the following sources:

1.L. Breiman, “Pasting small votes for classification in large databases and on-line”, Machine Learning, 36(1), 85-103, 1999.

2.L. Breiman, “Bagging predictors”, Machine Learning, 24(2), 123-140, 1996.

T. Ho, “The random subspace method for constructing decision forests”, Pattern Analysis and Machine Intelligence, 20(8), 832-844, 1998.

3.G. Louppe and P. Geurts, “Ensembles on Random Patches”, Machine Learning and Knowledge Discovery in Databases, 346-361, 2012.

https://jiamingmao.github.io/data-analysis/Lectures/

4. Schonlau M. Boosted Regression (Boosting): An introductory tutorial and a Stata plugin. *The Stata Journal*, 5(3), 330-354.