

www.statsoft.com

- Products
- Solutions
- Buy
- Trials
- Support

Textbook Boosting Trees Regression Classification

What can we help you find?

Search



- Elementary Concepts
- Statistics Glossary
- Basic Statistics
- ANOVA / MANOVA
- Association Rules
- Boosting Trees
- Canonical Analysis
- CHAID Analysis
- C & R Trees
- Classification Trees
- Cluster Analysis
- Correspondence Analysis
- Data Mining Techniques
- Discriminant Analysis
- Distribution Fitting
- Experimental Design
- Factor Analysis
- General Discrim. Analysis
- General Linear Models
- Generalized Additive Mod.
- Generalized Linear Mod.
- General Regression Mod.
- Graphical Techniques
- Ind. Components Analysis
- Linear Regression
- Log-Linear Analysis
- MARSplines
- Machine Learning
- Multidimensional Scaling
- Neural Networks
- Nonlinear Estimation
- Nonparametric Statistics
- Partial Least Squares
- Power Analysis
- Process Analysis
- Quality Control Charts
- Reliability / Item Analysis
- SEPATH (Structural eq.)
- Survival Analysis
- Text Mining
- Time Series / Forecasting

## Introduction to Boosting Trees for Regression and Classification









Data Mining, Boosted Trees - Session 19



- [Boosting Trees for Regression and Classification Introductory Overview](#)
- [Gradient Boosting Trees](#)
- [The Problem of Overfitting; Stochastic Gradient Boosting](#)
- [Stochastic Gradient Boosting Trees and Classification](#)
- [Large Numbers of Categories](#)

### Boosting Trees for Regression and Classification Introductory Overview

The general computational approach of stochastic gradient boosting is also known by the names TreeNet (TM Salford Systems, Inc.) and MART (TM Jerill, Inc.). Over the past few years, this technique has emerged as one of the most powerful methods for [predictive data mining](#). Some implementations of these powerful algorithms allow them to be used for regression as well as classification problems, with continuous and/or categorical predictors. Detailed technical

-  Variance Components
-  Statistical Advisor
-  Distribution Tables
-  References Cited
-  Send Comments
-  Business Solutions
-  Free Resources
-  About Textbook

descriptions of these methods can be found in Friedman (1999a, b) as well as Hastie, Tibshirani, & Friedman (2001).

## Gradient Boosting Trees

The algorithm for Boosting Trees evolved from the application of [boosting](#) methods to regression trees. The general idea is to compute a sequence of (very) simple trees, where each successive tree is built for the prediction residuals of the preceding tree. As described in the [General Classification and Regression Trees Introductory Overview](#), this method will build binary trees, i.e., partition the data into two samples at each split node. Now suppose that you were to limit the complexities of the trees to 3 nodes only: a root node and two child nodes, i.e., a single split. Thus, at each step of the boosting (boosting trees algorithm), a simple (best) partitioning of the data is determined, and the deviations of the observed values from the respective means (residuals for each partition) are computed. The next 3-node tree will then be fitted to those residuals, to find another partition that will further reduce the residual (error) variance for the data, given the preceding sequence of trees.

It can be shown that such "additive weighted expansions" of trees can eventually produce an excellent fit of the predicted values to the observed values, even if the specific nature of the relationships between the predictor variables and the dependent variable of interest is very complex (nonlinear in nature). Hence, the method of gradient boosting - fitting a weighted additive expansion of simple trees - represents a very general and powerful [machine learning](#) algorithm.

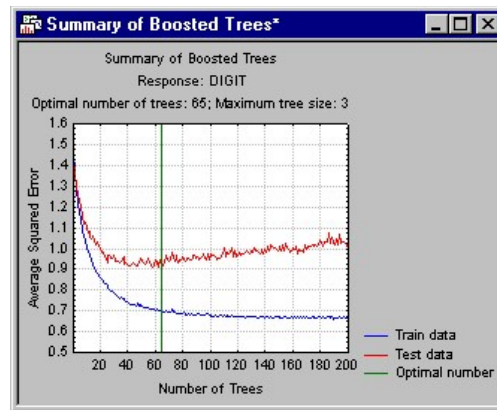
[To index](#)

## The Problem of Overfitting; Stochastic Gradient Boosting

One of the major problems of all machine learning algorithms is to "know when to stop," i.e., how to prevent the learning algorithm to fit esoteric aspects of the training data that are not likely to improve the predictive validity of the respective model. This issue is also known as the problem of [overfitting](#). To reiterate, this is a general problem applicable to most machine learning algorithms used in [predictive data mining](#). A general solution to this problem is to evaluate the quality of the fitted model by predicting observations in a test-sample of data that have not been used before to estimate the respective model(s). In this manner, one hopes to gage the predictive accuracy of the solution, and to detect when overfitting has occurred (or is starting to occur).

A similar approach is for each consecutive simple tree to be built for only a randomly selected subsample of the full data set. In other words, each consecutive tree is built for the prediction residuals (from all preceding trees) of an independently drawn random sample. The introduction of a certain degree of randomness into the analysis in this manner can serve as a powerful safeguard against overfitting (since each consecutive tree is built for a different sample of observations), and yield models (additive weighted expansions of simple trees) that generalize well to new observations, i.e., exhibit good predictive validity. This technique, i.e., performing consecutive boosting computations on independently drawn samples of observations, is known as stochastic gradient boosting.

Below is a plot of the prediction error function for the training data over successive trees and also an independently sampled testing data set at each stage.



With this graph, you can identify very quickly the point where the model (consisting of a certain number of successive trees) begins to overfit the data. Notice how the prediction error for the training data steadily decreases as more and more additive terms (trees) are added to the model. However, somewhere past 35 trees, the performance for independently sampled testing data actually begins to deteriorate, clearly indicating the point where the model begins to overfit the data.

[To index](#)

## Stochastic Gradient Boosting Trees and Classification

So far, the discussion of boosting trees has exclusively focused on regression problems, i.e., on the prediction of a continuous dependent variable. The technique can easily be expanded to handle classification problems as well (this is described in detail in Friedman, 1999a, section 4.6; in particular, see Algorithm 6).

First, different boosting trees are built for (fitted to) each category or class of the categorical dependent variable, after creating a coded variable (vector) of values for each class with the values 1 or 0 to indicate whether or not an observation does or does not belong to the respective class. In successive boosting steps, the algorithm will apply the [logistic transformation](#) (see also [Nonlinear Estimation](#)) to compute the residuals for subsequent boosting steps. To compute the final classification probabilities, the logistic transformation is again applied to the predictions for each 0/1 coded vector (class). This algorithm is described in detail in Friedman (1999a; see also Hastie, Tibshirani, and Freedman, 2001, for a description of this general procedure).

## Large Numbers of Categories

Note that the procedure for applying this method to classification problems requires that separate sequences of (boosted) trees be built for each category or class. Hence, the computational effort generally becomes larger by a multiple of what it takes to solve a simple regression prediction problem (for a single continuous dependent variable). Therefore, it is not prudent to analyze categorical dependent variables (class variables) with more than, approximately, 100 or so classes; past that point, the computations performed may require an unreasonable amount of effort and time. (For example, a problem with 200 boosting steps and 100 categories or classes for the dependent variable would yield  $200 * 100 = 20,000$  individual trees.)

[To index](#)