Package ‘TDboost’

January 18, 2024

Title  A Boosted Tweedie Compound Poisson Model
Version  1.5
Date  2024-01-17
Depends  R (>= 2.12.0), lattice
Description  An implementation of a boosted Tweedie compound Poisson model proposed by Yang, Y., Qian, W. and Zou, H. (2018) <doi:10.1080/07350015.2016.1200981>. It is capable of fitting a flexible nonlinear Tweedie compound Poisson model (or a gamma model) and capturing high-order interactions among predictors. This package is based on the ‘gbm’ package originally developed by Greg Ridgeway.
LazyData  yes
License  GPL-3
NeedsCompilation  yes
Date/Publication  2024-01-18 10:00:14 UTC
Author  Yi Yang [aut, cre] (http://www.math.mcgill.ca/yyang/),
        Hui Zou [aut] (http://users.stat.umn.edu/~zouxx019/),
        Wei Qian [aut] (https://sites.google.com/a/udel.edu/weiqian/),
        Greg Ridgeway [ctb, cph]
Maintainer  Yi Yang <yi.yang6@mcgill.ca>
Repository  CRAN

R topics documented:

  FHT ................................................................. 2
  plot.TDboost .................................................. 2
  predict.TDboost ................................................ 4
  relative.influence ............................................ 5
  summary.TDboost .............................................. 6
  TDboost .......................................................... 7
  TDboost.object ................................................ 12
  TDboost.perf .................................................. 13

Index  15
Simulation data generated based from the FHT model used in Yang, Y., Qian, W. and Zou, H. (2013).

Description

There are two data sets, one for training and the other for testing. The training data set has \( n = 200 \) observations and \( p = 6 \) predictors. The testing data set has \( n = 20 \) observations and \( p = 6 \) predictors. See details in Friedman et al. (2010).

Usage

data(FHT)

Format

Two data frames both contain the following columns:

- \( X_1 - X_6 \) predictor columns
- \( Y \) response variable

References


Examples

data(FHT)

plot.TDboost Marginal plots of fitted TDboost objects

Description

Plots the marginal effect of the selected variables by "integrating" out the other variables.

Usage

```r
## S3 method for class 'TDboost'
plot(x,
i.var = 1,
n.trees = x$n.trees,
continuous.resolution = 100,
return.grid = FALSE,
...)```
Arguments

x a TDboost.object fitted using a call to TDboost

i.var a vector of indices or the names of the variables to plot. If using indices, the variables are indexed in the same order that they appear in the initial TDboost formula. If length(i.var) is between 1 and 3 then plot.TDboost produces the plots. Otherwise, plot.TDboost returns only the grid of evaluation points and their average predictions

n.trees the number of trees used to generate the plot. Only the first n.trees trees will be used

continuous.resolution The number of equally space points at which to evaluate continuous predictors

return.grid if TRUE then plot.TDboost produces no graphics and only returns the grid of evaluation points and their average predictions. This is useful for customizing the graphics for special variable types or for dimensions greater than 3

Details

plot.TDboost produces low dimensional projections of the TDboost.object by integrating out the variables not included in the i.var argument. The function selects a grid of points and uses the weighted tree traversal method described in Friedman (2001) to do the integration. Based on the variable types included in the projection, plot.TDboost selects an appropriate display choosing amongst line plots, contour plots, and lattice plots. If the default graphics are not sufficient the user may set return.grid=TRUE, store the result of the function, and develop another graphic display more appropriate to the particular example.

Value

Nothing unless return.grid is true then plot.TDboost produces no graphics and only returns the grid of evaluation points and their average predictions.

Author(s)

Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>

References


See Also

TDboost, TDboost.object, plot
predict.TDboost  Predict method for TDboost Model Fits

Description
Predicted values based on an TDboost Tweedie regression model object

Usage
## S3 method for class 'TDboost'
predict(object,
         newdata,
         n.trees,
         single.tree=FALSE,
         type=c("response","link"),
         ...)

Arguments
object  Object of class inheriting from (TDboost.object)
newdata  Data frame of observations for which to make predictions
n.trees  Number of trees used in the prediction. n.trees may be a vector in which case
         predictions are returned for each iteration specified
single.tree  If single.tree=TRUE then predict.TDboost returns only the predictions from
              tree(s) n.trees

Details
predict.TDboost produces predicted values for each observation in newdata using the the first
n.trees iterations of the boosting sequence. If n.trees is a vector than the result is a matrix
with each column representing the predictions from TDboost models with n.trees[1] iterations,
The predictions from TDboost do not include the offset term. The user may add the value of the
offset to the predicted value if desired.
If object was fit using TDboost.fit there will be no Terms component. Therefore, the user has
greater responsibility to make sure that newdata is of the same format (order and number of vari-
ables) as the one originally used to fit the model.
Value

Returns a vector of predictions. By default the predictions are on the scale of f(x).

Author(s)

Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>

See Also

TDboost, TDboost.object

-----------

relative.influence  Methods for estimating relative influence

Description

Helper functions for computing the relative influence of each variable in the TDboost object.

Usage

  relative.influence(object, n.trees)
  permutation.test.TDboost(object, n.trees)
  TDboost.loss(y,f,w,offset,dist,baseline)

Arguments

  object   a TDboost object created from an initial call to TDboost.
  n.trees  the number of trees to use for computations.
  y,f,w,offset,dist,baseline
            For TDboost.loss: These components are the outcome, predicted value, observation weight, offset, distribution, and comparison loss function, respectively.

Details

This is not intended for end-user use. These functions offer the different methods for computing the relative influence in summary.TDboost. TDboost.loss is a helper function for permutation.test.TDboost.

Value

Returns an unprocessed vector of estimated relative influences.

Author(s)

Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>
References


See Also

summary.TDboost

summary.TDboost Summary of a TDboost object

Description

Computes the relative influence of each variable in the TDboost object.

Usage

## S3 method for class 'TDboost'
summary(object,
cBars=length(object$var.names),
n.trees=object$n.trees,
plotit=TRUE,
order=TRUE,
method=relative.influence,
normalize=TRUE,
...)

Arguments

object a TDboost object created from an initial call to TDboost.
cBars the number of bars to plot. If order=TRUE the only the variables with the cBars largest relative influence will appear in the barplot. If order=FALSE then the first cBars variables will appear in the plot. In either case, the function will return the relative influence of all of the variables.
n.trees the number of trees used to generate the plot. Only the first n.trees trees will be used.
plotit an indicator as to whether the plot is generated.
order an indicator as to whether the plotted and/or returned relative influences are sorted.
method

The function used to compute the relative influence. `relative.influence` is the default and is the same as that described in Friedman (2001). The other current (and experimental) choice is `permutation.test.TDboost`. This method randomly permutes each predictor variable at a time and computes the associated reduction in predictive performance. This is similar to the variable importance measures Breiman uses for random forests, but TDboost currently computes using the entire training dataset (not the out-of-bag observations).

normalize

if FALSE then `summary.TDboost` returns the unnormalized influence.

other arguments passed to the plot function.

Details

This returns the reduction attributable to each variable in sum of squared error in predicting the gradient on each iteration. It describes the relative influence of each variable in reducing the loss function. See the references below for exact details on the computation.

Value

Returns a data frame where the first component is the variable name and the second is the computed relative influence, normalized to sum to 100.

Author(s)

Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>

References


See Also

TDboost

TDboost Tweedie Regression Modeling

Description

Fits TDboost Tweedie Regression models.
Usage

TDboost(formula = formula(data),
    distribution = list(name="EDM",alpha=1.5),
    data = list(),
    weights,
    var.monotone = NULL,
    n.trees = 100,
    interaction.depth = 1,
    n.minobsinnode = 10,
    shrinkage = 0.001,
    bag.fraction = 0.5,
    train.fraction = 1.0,
    cv.folds=0,
    keep.data = TRUE,
    verbose = TRUE)

TDboost.fit(x,y,
    offset = NULL,
    misc = NULL,
    distribution = list(name="EDM",alpha=1.5),
    w = NULL,
    var.monotone = NULL,
    n.trees = 100,
    interaction.depth = 1,
    n.minobsinnode = 10,
    shrinkage = 0.001,
    bag.fraction = 0.5,
    train.fraction = 1.0,
    keep.data = TRUE,
    verbose = TRUE,
    var.names = NULL,
    response.name = NULL)

TDboost.more(object,
    n.new.trees = 100,
    data = NULL,
    weights = NULL,
    offset = NULL,
    verbose = NULL)

Arguments

formula a symbolic description of the model to be fit. The formula may include an offset term (e.g. y~offset(n)+x). If keep.data=FALSE in the initial call to TDboost then it is the user's responsibility to resupply the offset to TDboost.more.

distribution a list with a component name specifying the distribution and any additional parameters needed. Tweedie regression is available and distribution must a list of the form list(name="EDM",alpha=1.5) where alpha is the index param-
The current version’s Tweedie regression methods do not handle non-constant weights and will stop.

**data**

an optional data frame containing the variables in the model. By default the variables are taken from `environment(formula)`, typically the environment from which `TDboost` is called. If `keep.data=TRUE` in the initial call to `TDboost` then `TDboost` stores a copy with the object. If `keep.data=FALSE` then subsequent calls to `TDboost.more` must resupply the same dataset. It becomes the user’s responsibility to resupply the same data at this point.

**weights**

an optional vector of weights to be used in the fitting process. Must be positive but do not need to be normalized. If `keep.data=FALSE` in the initial call to `TDboost` then it is the user’s responsibility to resupply the weights to `TDboost.more`.

**var.monotone**

an optional vector, the same length as the number of predictors, indicating which variables have a monotone increasing (+1), decreasing (-1), or arbitrary (0) relationship with the outcome.

**n.trees**

the total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion.

**cv.folds**

Number of cross-validation folds to perform. If `cv.folds>1` then `TDboost`, in addition to the usual fit, will perform a cross-validation, calculate an estimate of generalization error returned in `cv.error`.

**interaction.depth**

The maximum depth of variable interactions. 1 implies an additive model, 2 implies a model with up to 2-way interactions, etc.

**n.minobsinnode**

minimum number of observations in the trees terminal nodes. Note that this is the actual number of observations not the total weight.

**shrinkage**

a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction.

**bag.fraction**

the fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomness into the model fit. If `bag.fraction<1` then running the same model twice will result in similar but different fits. `TDboost` uses the R random number generator so `set.seed` can ensure that the model can be reconstructed. Preferably, the user can save the returned `TDboost.object` using `save`.

**train.fraction**

The first `train.fraction * nrow(data)` observations are used to fit the `TDboost` and the remainder are used for computing out-of-sample estimates of the loss function.

**keep.data**

a logical variable indicating whether to keep the data and an index of the data stored with the object. Keeping the data and index makes subsequent calls to `TDboost.more` faster at the cost of storing an extra copy of the dataset.

**object**

a `TDboost` object created from an initial call to `TDboost`.

**n.new.trees**

the number of additional trees to add to object.

**verbose**

If TRUE, `TDboost` will print out progress and performance indicators. If this option is left unspecified for `TDboost.more` then it uses `verbose` from `object`.
For `TDboost.fit`: `x` is a data frame or data matrix containing the predictor variables and `y` is the vector of outcomes. The number of rows in `x` must be the same as the length of `y`.

- **offset**: a vector of values for the offset
- **misc**: For `TDboost.fit`: `misc` is an R object that is simply passed on to the TDboost engine.
- **w**: For `TDboost.fit`: `w` is a vector of weights of the same length as the `y`.
- **var.names**: For `TDboost.fit`: A vector of strings of length equal to the number of columns of `x` containing the names of the predictor variables.
- **response.name**: For `TDboost.fit`: A character string label for the response variable.

**Details**

This package implements a regression tree based gradient boosting estimator for nonparametric multiple Tweedie regression. The code is a modified version of `gbm` library originally written by Greg Ridgeway.

Boosting is the process of iteratively adding basis functions in a greedy fashion so that each additional basis function further reduces the selected loss function. This implementation closely follows Friedman’s Gradient Boosting Machine (Friedman, 2001).

In addition to many of the features documented in the Gradient Boosting Machine, TDboost offers additional features including the out-of-bag estimator for the optimal number of iterations, the ability to store and manipulate the resulting TDboost object.

`TDboost.fit` provides the link between R and the C++ TDboost engine. TDboost is a front-end to `TDboost.fit` that uses the familiar R modeling formulas. However, `model.frame` is very slow if there are many predictor variables. For power-users with many variables use `TDboost.fit`. For general practice TDboost is preferable.

**Value**

`TDboost`, `TDboost.fit`, and `TDboost.more` return a `TDboost.object`.

**Author(s)**

Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>

**References**


See Also

TDboost.object, TDboost.perf, plot.TDboost, predict.TDboost, summary.TDboost,

Examples

data(FHT)

# training on data1
TDboost1 <- TDboost(Y~X1+X2+X3+X4+X5+X6, # formula
data=data1, # dataset
var.monotone=c(0,0,0,0,0,0), # -1: monotone decrease,
# +1: monotone increase,
# 0: no monotone restrictions
distribution=list(name="EDM",alpha=1.5), # specify Tweedie index parameter
n.trees=3000, # number of trees
shrinkage=0.005, # shrinkage or learning rate,
# 0.001 to 0.1 usually work
interaction.depth=3, # 1: additive model, 2: two-way interactions, etc.
bag.fraction = 0.5, # subsampling fraction, 0.5 is probably best
train.fraction = 0.5, # fraction of data for training,
# first train.fraction*N used for training
n.minobsinnode = 10, # minimum total weight needed in each node
cv.folds = 5, # do 5-fold cross-validation
keep.data=TRUE, # keep a copy of the dataset with the object
verbose=TRUE) # print out progress

# print out the optimal iteration number M
best.iter <- TDboost.perf(TDboost1,method="test")
print(best.iter)

# check performance using 5-fold cross-validation
best.iter <- TDboost.perf(TDboost1,method="cv")
print(best.iter)

# plot the performance
# plot variable influence
summary(TDboost1,n.trees=1) # based on the first tree
summary(TDboost1,n.trees=best.iter) # based on the estimated best number of trees

# making prediction on data2
f.predict <- predict.TDboost(TDboost1,data2,best.iter)

# least squares error
print(sum((data2$Y-f.predict)^2))

# create marginal plots
# plot variable X1 after "best" iterations
plot.TDboost(TDboost1,1,best.iter)
# contour plot of variables 1 and 3 after "best" iterations
plot.TDboost(TDboost1,c(1,3),best.iter)

# do another 20 iterations
TDboost2 <- TDboost更多的TDboost1,20,  
    verbose=FALSE) # stop printing detailed progress

# fit a gamma model (when alpha = 2.0)
data2 <- data[!is.na(data$Y),]
TDboost3 <- TDboost(Y~X1+X2+X3+X4+X5+X6,  
    data=data2,  
    # formula
    distribution=list(name="EDM", alpha=2.0),  
    n.trees=3000,  
    # dataset
    train.fraction = 0.5,  
    # number of trees
    verbose=TRUE)  
    # print out progress
best.iter2 <- TDboost.perf(TDboost3, method="test")

TDboost.object

TDboost Tweedie Regression Model Object

Description
These are objects representing fitted TDboosts.

Value
- **initF** the "intercept" term, the initial predicted value to which trees make adjustments
- **fit** a vector containing the fitted values on the scale of regression function
- **train.error** a vector of length equal to the number of fitted trees containing the value of the loss function for each boosting iteration evaluated on the training data
- **valid.error** a vector of length equal to the number of fitted trees containing the value of the loss function for each boosting iteration evaluated on the validation data
- **cv.error** if cv.folds<2 this component is NULL. Otherwise, this component is a vector of length equal to the number of fitted trees containing a cross-validated estimate of the loss function for each boosting iteration
- **oobag.improve** a vector of length equal to the number of fitted trees containing an out-of-bag estimate of the marginal reduction in the expected value of the loss function. The out-of-bag estimate uses only the training data and is useful for estimating the optimal number of boosting iterations. See **TDboost.perf**
- **trees** a list containing the tree structures.
- **c.splits** a list of all the categorical splits in the collection of trees. If the trees[[i]] component of a TDboost object describes a categorical split then the splitting value will refer to a component of c.splits. That component of c.splits will be a vector of length equal to the number of levels in the categorical split variable. -1 indicates left, +1 indicates right, and 0 indicates that the level was not present in the training data

Structure
The following components must be included in a legitimate TDboost object.
**TDboost.perf**

**Author(s)**
Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>

**See Also**
TDboost

**Description**
Estimates the optimal number of boosting iterations for a TDboost object and optionally plots various performance measures

**Usage**

```r
TDboost.perf(object, plot.it = TRUE, oobag.curve = FALSE, overlay = TRUE, method)
```

**Arguments**
- **object**: a TDboost.object created from an initial call to TDboost.
- **plot.it**: an indicator of whether or not to plot the performance measures. Setting plot.it=TRUE creates two plots. The first plot plots object$train.error (in black) and object$valid.error (in red) versus the iteration number. The scale of the error measurement, shown on the left vertical axis, depends on the distribution argument used in the initial call to TDboost.
- **oobag.curve**: indicates whether to plot the out-of-bag performance measures in a second plot.
- **overlay**: if TRUE and oobag.curve=TRUE then a right y-axis is added to the training and test error plot and the estimated cumulative improvement in the loss function is plotted versus the iteration number.
- **method**: indicate the method used to estimate the optimal number of boosting iterations. `method="OOB"` computes the out-of-bag estimate and `method="test"` uses the test (or validation) dataset to compute an out-of-sample estimate. `method="cv"` extracts the optimal number of iterations using cross-validation if TDboost was called with `cv.folds`>1

**Value**

TDboost.perf returns the estimated optimal number of iterations. The method of computation depends on the method argument.
Author(s)
Yi Yang <yi.yang6@mcgill.ca>, Wei Qian <wxqsma@rit.edu> and Hui Zou <hzou@stat.umn.edu>

References

See Also
TDboost, TDboost.object
Index

* datasets
  FHT, 2
* hplot
  plot.TDboost, 2
  relative.influence, 5
  summary.TDboost, 6
* methods
  TDboost.object, 12
* models
  predict.TDboost, 4
  TDboost, 7
* nonlinear
  TDboost, 7
  TDboost.perf, 13
* nonparametric
  TDboost, 7
  TDboost.perf, 13
* regression
  predict.TDboost, 4
* survival
  TDboost, 7
  TDboost.perf, 13
* tree
  TDboost, 7
  TDboost.perf, 13

data1 (FHT), 2
data2 (FHT), 2

FHT, 2

lattice, 3

model.frame, 10

permutation.test.TDboost, 7
permutation.test.TDboost
  (relative.influence), 5
plot, 3
plot.TDboost, 2, 11
predict.TDboost, 4, 11

relative.influence, 5, 7

save, 9
summary.TDboost, 5, 6, 11

TDboost, 3, 5–7, 7, 9, 13, 14
TDboost.fit, 4
TDboost.loss(relative.influence), 5
TDboost.more, 8, 9
TDboost.object, 3–5, 9–11, 12, 13, 14
TDboost.perf, 11, 12, 13

15