R documentation

of 'erboost.Rd' etc.

October 11, 2013

---

erboost *ER-Boost Expectile Regression Modeling*

Description

Fits ER-Boost Expectile Regression models.

Usage

```
erboost(formula = formula(data),
    distribution = list(name="expectile",alpha=0.5),
    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)

erboost.fit(x,y,
        offset = NULL,
        misc = NULL,
        distribution = list(name="expectile",alpha=0.5),
        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)

erboost.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 erboost then it is the user's responsibility to resupply the offset to erboost.more.
distribution	a list with a component name specifying the distribution and any additional parameters needed. Expectile regression is available and distribution must a list of the form list(name="expectile",alpha=0.25) where alpha is the expectile to estimate. The current version's expectile 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 erboost is called. If keep.data=TRUE in the initial call to erboost then erboost stores a copy with the object. If keep.data=FALSE then subsequent calls to erboost.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 erboost then it is the user's responsibility to resupply the weights to erboost.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 erboost, 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 randomnesses into the model fit. If bag.fraction<1 then running the same model twice will result in similar but

	different fits. erboost uses the R random number generator so set.seed can ensure that the model can be reconstructed. Preferably, the user can save the returned erboost.object using save.
train.fraction	The first train.fraction * nrows(data) observations are used to fit the erboost 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 erboost.more faster at the cost of storing an extra copy of the dataset.
object	a erboost object created from an initial call to erboost.
n.new.trees	the number of additional trees to add to object.
verbose	If TRUE, erboost will print out progress and performance indicators. If this option is left unspecified for erboost.more then it uses verbose from object.
x, y	For erboost.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 erboost.fit: misc is an R object that is simply passed on to the erboost engine.
w	For erboost.fit: w is a vector of weights of the same length as the y.
var.names	For erboost.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 erboost.fit: A character string label for the response variable.

Details

Expectile regression (Newey & Powell 1987) is a nice tool for estimating the conditional expectiles of a response variable given a set of covariates. This package implements a regression tree based gradient boosting estimator for nonparametric multiple expectile regression. The code is a modified version of gbm library (http://cran.r-project.org/web/packages/gbm/) 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, erboost offers additional features including the out-of-bag estimator for the optimal number of iterations, the ability to store and manipulate the resulting erboost object.

erboost.fit provides the link between R and the C++ erboost engine. erboost is a front-end to erboost.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 erboost.fit. For general practice erboost is preferable.

Value

erboost, erboost.fit, and erboost.more return a erboost.object.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

G. Ridgeway (1999). "The state of boosting," *Computing Science and Statistics* 31:172-181.

http://cran.r-project.org/web/packages/gbm/

J.H. Friedman (2001). "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics* 29(5):1189-1232.

J.H. Friedman (2002). "Stochastic Gradient Boosting," *Computational Statistics and Data Analysis* 38(4):367-378.

See Also

erboost.object, erboost.perf, plot.erboost, predict.erboost, summary.erboost,

Examples

```
N <- 1000
X1 <- runif(N)
X2 <- 2*runif(N)
X3 <- ordered(sample(letters[1:4],N,replace=TRUE),levels=letters[4:1])
X4 <- factor(sample(letters[1:6],N,replace=TRUE))
X5 <- factor(sample(letters[1:3],N,replace=TRUE))
X6 <- 3*runif(N)
mu <- c(-1,0,1,2)[as.numeric(X3)]

SNR <- 10 # signal-to-noise ratio
Y <- X1**1.5 + 2 * (X2**.5) + mu
sigma <- sqrt(var(Y)/SNR)
Y <- Y + rnorm(N,0,sigma)

# introduce some missing values
X1[sample(1:N,size=500)] <- NA
X4[sample(1:N,size=300)] <- NA

data <- data.frame(Y=Y,X1=X1,X2=X2,X3=X3,X4=X4,X5=X5,X6=X6)

# fit initial model
erboost1 <- erboost(Y~X1+X2+X3+X4+X5+X6,         # formula
    data=data,                   # dataset
    var.monotone=c(0,0,0,0,0,0), # -1: monotone decrease,
                                 # +1: monotone increase,
                                 #  0: no monotone restrictions
    distribution=list(name="expectile",alpha=0.5),
                                 # expectile, quantile
    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


# check performance using a 50% heldout test set
best.iter <- erboost.perf(erboost1,method="test")
print(best.iter)

# check performance using 5-fold cross-validation
best.iter <- erboost.perf(erboost1,method="cv")
print(best.iter)

# plot the performance
# plot variable influence
summary(erboost1,n.trees=1)         # based on the first tree
summary(erboost1,n.trees=best.iter) # based on the estimated best number of trees

# make some new data
N <- 1000
X1 <- runif(N)
X2 <- 2*runif(N)
X3 <- ordered(sample(letters[1:4],N,replace=TRUE))
X4 <- factor(sample(letters[1:6],N,replace=TRUE))
X5 <- factor(sample(letters[1:3],N,replace=TRUE))
X6 <- 3*runif(N)
mu <- c(-1,0,1,2)[as.numeric(X3)]

Y <- X1**1.5 + 2 * (X2**.5) + mu + rnorm(N,0,sigma)

data2 <- data.frame(Y=Y,X1=X1,X2=X2,X3=X3,X4=X4,X5=X5,X6=X6)

# predict on the new data using "best" number of trees
# f.predict generally will be on the canonical scale (logit,log,etc.)
f.predict <- predict.erboost(erboost1,data2,best.iter)

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

# create marginal plots
# plot variable X1,X2,X3 after "best" iterations
par(mfrow=c(1,3))
plot.erboost(erboost1,1,best.iter)
plot.erboost(erboost1,2,best.iter)
plot.erboost(erboost1,3,best.iter)
par(mfrow=c(1,1))
# contour plot of variables 1 and 2 after "best" iterations
plot.erboost(erboost1,1:2,best.iter)
# lattice plot of variables 2 and 3
plot.erboost(erboost1,2:3,best.iter)
# lattice plot of variables 3 and 4
plot.erboost(erboost1,3:4,best.iter)

# 3-way plots
plot.erboost(erboost1,c(1,2,6),best.iter,cont=20)
plot.erboost(erboost1,1:3,best.iter)
```

```
plot.erboost(erboost1,2:4,best.iter)
plot.erboost(erboost1,3:5,best.iter)

# do another 100 iterations
erboost2 <- erboost.more(erboost1,100,
                verbose=FALSE) # stop printing detailed progress
```

`erboost.object` *ER-Boost Expectile Regression Model Object*

Description

These are objects representing fitted `erboost`s.

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 `erboost.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 `erboost` 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 `erboost` object.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

See Also

`erboost`

erboost.perf *erboost performance*

Description

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

Usage

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

Arguments

object	a erboost.object created from an initial call to erboost.
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 erboost.
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 erboost was called with cv.folds>1

Value

erboost.perf returns the estimated optimal number of iterations. The method of computation depends on the method argument.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

G. Ridgeway (2003). "A note on out-of-bag estimation for estimating the optimal number of boosting iterations," a working paper available at http://www.i-pensieri.com/gregr/erboost.shtml.

See Also

erboost, erboost.object

`interact.erboost` *Estimate the strength of interaction effects*

Description

Computes Friedman's H-statistic to assess the strength of variable interactions.

Usage

```
interact.erboost(x,
             data,
             i.var = 1,
             n.trees = x$n.trees)
```

Arguments

`x`	a `erboost.object` fitted using a call to `erboost`
`data`	the dataset used to construct `x`. If the original dataset is large, a random subsample may be used to accelerate the computation in `interact.erboost`
`i.var`	a vector of indices or the names of the variables for compute the interaction effect. If using indices, the variables are indexed in the same order that they appear in the initial `erboost` formula.
`n.trees`	the number of trees used to generate the plot. Only the first `n.trees` trees will be used

Details

`interact.erboost` computes Friedman's H-statistic to assess the relative strength of interaction effects in non-linear models. H is on the scale of [0-1] with higher values indicating larger interaction effects. To connect to a more familiar measure, if x_1 and x_2 are uncorrelated covariates with mean 0 and variance 1 and the model is of the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

then

$$H = \frac{\beta_3}{\sqrt{\beta_1^2 + \beta_2^2 + \beta_3^2}}$$

Value

Returns the value of H.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

J.H. Friedman and B.E. Popescu (2005). "Predictive Learning via Rule Ensembles." Section 8.1

See Also

erboost, erboost.object

`plot.erboost` *Marginal plots of fitted erboost objects*

Description

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

Usage

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

Arguments

`x`	a `erboost.object` fitted using a call to `erboost`
`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 `erboost` formula. If `length(i.var)` is between 1 and 3 then `plot.erboost` produces the plots. Otherwise, `plot.erboost` 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.erboost` 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
`...`	other arguments passed to the plot function

Details

plot.erboost produces low dimensional projections of the erboost.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.erboost 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.erboost produces no graphics and only returns the grid of evaluation points and their average predictions.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

J.H. Friedman (2001). "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics 29(4).

See Also

erboost, erboost.object, plot

---

predict.erboost *Predict method for erboost Model Fits*

Description

Predicted values based on an ER-Boost Expectile regression model object

Usage

```
## S3 method for class 'erboost'
predict(object,
        newdata,
        n.trees,
        single.tree=FALSE,
        ...)
```

Arguments

object	Object of class inheriting from (erboost.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.erboost returns only the predictions from tree(s) n.trees
...	further arguments passed to or from other methods

Details

predict.erboost 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 erboost models with n.trees[1] iterations, n.trees[2] iterations, and so on.

The predictions from erboost 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 erboost.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 variables) 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 <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

See Also

erboost, erboost.object

---

relative.influence *Methods for estimating relative influence*

Description

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

Usage

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

Arguments

object	a erboost object created from an initial call to erboost.
n.trees	the number of trees to use for computations.
y,f,w,offset,dist,baseline	For erboost.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.erboost. erboost.loss is a helper function for permutation.test.erboost.

Value

Returns an unprocessed vector of estimated relative influences.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

J.H. Friedman (2001). "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics 29(5):1189-1232.

L. Breiman (2001). "Random Forests," Available at ftp://ftp.stat.berkeley.edu/pub/users/breiman/randomforest2001.pdf.

See Also

summary.erboost

---

shrink.erboost *L1 shrinkage of the predictor variables in a erboost*

Description

Performs recursive shrinkage in each of the trees in a erboost fit using different shrinkage parameters for each variable.

Usage

```
shrink.erboost(object,
               n.trees,
               lambda = rep(10, length(object$var.names)),
               ...)
```

Arguments

object	A erboost.object
n.trees	the number of trees to use
lambda	a vector with length equal to the number of variables containing the shrinkage parameter for each variable
...	other parameters (ignored)

Details

This function is currently experimental. Used in conjunction with a gradient ascent search for inclusion of variables.

Value

predF	Predicted values from the shrunken tree
objective	The value of the loss function associated with the predicted values
gradient	A vector with length equal to the number of variables containing the derivative of the objective function with respect to beta, the logit transform of the shrinkage parameter for each variable

Warning

This function is experimental.

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

Hastie, T. J., and Pregibon, D. "Shrinking Trees." AT&T Bell Laboratories Technical Report (March 1990). http://www-stat.stanford.edu/~hastie/Papers/shrinktree.ps

See Also

shrink.erboost.pred, erboost

shrink.erboost.pred *Predictions from a shrunked erboost*

Description

Makes predictions from a shrunken erboost model.

Usage

```
shrink.erboost.pred(object,
                    newdata,
                    n.trees,
                    lambda = rep(1, length(object$var.names)),
                    ...)
```

Arguments

object	a erboost.object
newdata	dataset for predictions
n.trees	the number of trees to use
lambda	a vector with length equal to the number of variables containing the shrinkage parameter for each variable
...	other parameters (ignored)

Value

A vector with length equal to the number of observations in newdata containing the predictions

Warning

This function is experimental

Author(s)

Yi Yang <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

See Also

shrink.erboost, erboost

summary.erboost *Summary of a erboost object*

Description

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

Usage

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

Arguments

object	a erboost object created from an initial call to erboost.
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.erboost. 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 erboost currently computes using the entire training dataset (not the out-of-bag observations.
normalize	if FALSE then summary.erboost returns the unnormalized influence.
...	other arguments passed to the plot function.

Details

This returns the reduction attributeable to each varaible 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 <yiyang@umn.edu> and Hui Zou <hzou@stat.umn.edu>

References

Yang, Y. and Zou, H. (2013), "Nonparametric Multiple Expectile Regression via ER-Boost," *Journal of Statistical Computation and Simulation*. Accept with minor revisions.

BugReport: http://code.google.com/p/erboost/

J.H. Friedman (2001). "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics 29(5):1189-1232.

L. Breiman (2001). "Random Forests," Available at ftp://ftp.stat.berkeley.edu/pub/users/breiman/randomforest2001.pdf.

See Also

erboost

Index