The gbm Package

July 27, 2006

Version 1.5-7

Date 2006-4-15

Title Generalized Boosted Regression Models

Author Greg Ridgeway <gregr@rand.org>

Maintainer Greg Ridgeway <gregr@rand.org>

Depends R (>= 2.1.0), survival, lattice, mgcv

Description This package implements extensions to Freund and Schapire’s AdaBoost algorithm and J.
Friedman’s gradient boosting machine. Includes regression methods for least squares, absolute
loss, logistic, Poisson, Cox proportional hazards partial likelihood, and AdaBoost exponential
loss.

License GPL (version 2 or newer)

URL http://www.i-pensieri.com/gregr/gbm.shtml

R topics documented:

basehaz.gbm .......................................................... 2
calibrate.plot ....................................................... 3
gbm-package .......................................................... 4
gbm ................................................................. 5
gbm.object ........................................................... 10
gbm.perf ............................................................ 11
plot.gbm ............................................................ 12
predict.gbm ......................................................... 14
pretty.gbm.tree ................................................... 15
quantile.rug ........................................................ 16
relative.influence .................................................. 17
shrink.gbm .......................................................... 18
shrink.gbm.pred .................................................... 19
summary.gbm ....................................................... 20

Index 22
basehaz.gbm  

Baseline hazard function

Description
Computes the Breslow estimator of the baseline hazard function for a proportional hazard regression model.

Usage
```r
basehaz.gbm(t, delta, f.x, 
  t.eval = NULL, 
  smooth = FALSE, 
  cumulative = TRUE)
```

Arguments
- `t`: the survival times
- `delta`: the censoring indicator
- `f.x`: the predicted values of the regression model on the log hazard scale
- `t.eval`: values at which the baseline hazard will be evaluated
- `smooth`: if TRUE basehaz.gbm will smooth the estimated baseline hazard using Friedman's super smoother `supsmu`
- `cumulative`: if TRUE the cumulative survival function will be computed

Details
The proportional hazard model assumes $h(t|x) = \lambda(t) \exp(f(x))$. `gbm` can estimate the $f(x)$ component via partial likelihood. After estimating $f(x)$, `basehaz.gbm` can compute the a non-parametric estimate of $\lambda(t)$.

Value
a vector of length equal to the length of `t` (or of length `t.eval` if `t.eval` is not `NULL`) containing the baseline hazard evaluated at `t` (or at `t.eval` if `t.eval` is not `NULL`). If `cumulative` is set to `TRUE` then the returned vector evaluates the cumulative hazard function at those values.

Author(s)
Greg Ridgeway (gregr@rand.org)

References

See Also
- `survfit`, `gbm`
Description

An experimental diagnostic tool that plots the fitted values versus the actual average values. Currently developed for only distribution="bernoulli".

Usage

calibrate.plot(y,p,

distribution="bernoulli",
replace=TRUE,
line.par=list(col="black"),
shade.col="lightyellow",
shade.density=NULL,
rug.par=list(side=1),
xlab="Predicted value",
ylab="Observed average",
xlim=...,
ylim=...,
...)

Arguments

y the outcome 0-1 variable
p the predictions estimating E(y|x)
distribution the loss function used in creating p. bernoulli and poisson are currently the only special options. All others default to squared error assuming gaussian
replace determines whether this plot will replace or overlay the current plot. replace=FALSE is useful for comparing the calibration of several methods
line.par graphics parameters for the line
shade.col color for shading the 2 SE region. shade.col=NA implies no 2 SE region
shade.density the density parameter for polygon
rug.par graphics parameters passed to rug
xlab x-axis label corresponding to the predicted values
ylab y-axis label corresponding to the observed average
xlim,ylim x and y-axis limits. If not specified the function will select limits
... other graphics parameters passed on to the plot function

Details

Uses qam to estimate E(y|p). Well-calibrated predictions imply that E(y|p) = p. The plot also includes a pointwise 95 band.

Value

calibrate.plot returns no values.
gbm-package

Description

This package implements extensions to Freund and Schapire’s AdaBoost algorithm and J. Friedman’s gradient boosting machine. Includes regression methods for least squares, absolute loss, logistic, Poisson, Cox proportional hazards partial likelihood, and AdaBoost exponential loss.

Details

Package: gbm
Version: 1.5-6
Date: 2006-1-20
Depends: R (>= 2.1.0), survival, lattice, mgcv
License: GPL (version 2 or newer)
URL: http://www.i-pensieri.com/gregr/gbm.shtml
Built: R 2.2.1; i386-pc-mingw32; 2006-02-24 18:09:42; windows

Index:

basehaz.gbm Baseline hazard function
calibrate.plot Calibration plot
gbm Generalized Boosted Regression Modeling
gbm.object Generalized Boosted Regression Model Object
gbm.perf GBM performance
plot.gbm Marginal plots of fitted gbm objects
predict.gbm Predict method for GBM Model Fits
pretty.gbm.tree Print gbm tree components
quantile.rug Quantile rug plot
relative.influence Methods for estimating relative influence
shrink.gbm L1 shrinkage of the predictor variables in a GBM
shrink.gbm.pred Predictions from a shrunked GBM
summary.gbm Summary of a gbm object

Further information is available in the following vignettes:

gbm Generalized Boosted Models: A guide to the gbm package (source, pdf)

Author(s)

Greg Ridgeway (gregr@rand.org)

References

http://www.i-pensieri.com/gregr/gbm.shtml

Description

Fits generalized boosted regression models.

Usage

```r
gbm(formula = formula(data),
    distribution = "bernoulli",
    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)

gbm.fit(x,y,
offset = NULL,
misc = NULL,
distribution = "bernoulli",
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)

gbm.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 gbm then it is the user’s responsibility to resupply the offset to gbm.more.

distribution a description of the error distribution to be used in the model. Currently available options are "gaussian" (squared error), "laplace" (absolute loss), "bernoulli" (logistic regression for 0-1 outcomes), "adaboost" (the AdaBoost exponential loss for 0-1 outcomes), "poisson" (count outcomes), and "coxph" (censored observations). The current version’s Laplace distribution does 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 gbm is called. If keep.data=TRUE in the initial call to gbm then gbm stores a copy with the object. If keep.data=FALSE then subsequent calls to gbm.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 gbm then it is the user’s responsibility to resupply the weights to gbm.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.
**gbm**

**cv.folds** | Number of cross-validation folds to perform. If `cv.folds>1` then gbm, 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. gbm uses the R random number generator so set.seed can ensure that the model can be reconstructed. Preferably, the user can save the returned gbm.object using save.

**train.fraction** | The first `train.fraction * nrow(data)` observations are used to fit the gbm 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 gbm.more faster at the cost of storing an extra copy of the dataset.

**object** | a gbm object created from an initial call to gbm.

**n.new.trees** | the number of additional trees to add to object.

**verbose** | If TRUE, gbm will print out progress and performance indicators. If this option is left unspecified for gbm.more then it uses verbose from object.

**x, y** | For gbm.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 gbm.fit: misc is an R object that is simply passed on to the gbm engine. It can be used for additional data for the specific distribution. Currently it is only used for passing the censoring indicator for the Cox proportional hazards model.

**w** | For gbm.fit: w is a vector of weights of the same length as the y.

**var.names** | For gbm.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 gbm.fit: A character string label for the response variable.

**Details**

See vignette("gbm") for technical details of the package. Also available at ../doc/gbm.pdf (if you are using HTML help).

This package implements the generalized boosted modeling framework. 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, \texttt{gbm} offers additional features including the out-of-bag estimator for the optimal number of iterations, the ability to store and manipulate the resulting \texttt{gbm} object, and a variety of other loss functions that had not previously had associated boosting algorithms, including the Cox partial likelihood for censored data, the poisson likelihood for count outcomes, and a gradient boosting implementation to minimize the AdaBoost exponential loss function.

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

**Value**

\texttt{gbm}, \texttt{gbm.fit}, and \texttt{gbm.more} return a \texttt{gbm.object}.

**Author(s)**

Greg Ridgeway (gregr@rand.org)

**References**


http://www.i-pensieri.com/gregr/gbm.shtml


**See Also**

\texttt{gbm.object}, \texttt{gbm.perf}, \texttt{plot.gbm}, \texttt{predict.gbm}, \texttt{summary.gbm}, \texttt{pretty.gbm.tree}.

**Examples**

```r
# A least squares regression example
# create some data

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
gbm1 <- gbm(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="gaussian", # bernoulli, adaboost, gaussian,
# poisson, and coxph available
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 an out-of-bag estimator
# OOB underestimates the optimal number of iterations, performance is competitive
best.iter <- gbm.perf(gbm1,method="OOB")
print(best.iter)

# check performance using a 50
best.iter <- gbm.perf(gbm1,method="test")
print(best.iter)

# check performance using 5-fold cross-validation
best.iter <- gbm.perf(gbm1,method="cv")
print(best.iter)

# plot the performance
# plot variable influence
summary(gbm1,n.trees=1) # based on the first tree
summary(gbm1,n.trees=best.iter) # based on the estimated best number of trees

# compactly print the first and last trees for curiosity
print(pretty.gbm.tree(gbm1,1))
print(pretty.gbm.tree(gbm1,gbm1$n.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.gbm(gbm1,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.gbm(gbm1,1,best.iter)
plot.gbm(gbm1,2,best.iter)
plot.gbm(gbm1,3,best.iter)
par(mfrow=c(1,1))

# contour plot of variables 1 and 2 after "best" iterations
plot.gbm(gbm1,1:2,best.iter)

# lattice plot of variables 2 and 3
plot.gbm(gbm1,2:3,best.iter)

# lattice plot of variables 3 and 4
plot.gbm(gbm1,3:4,best.iter)

# 3-way plots
plot.gbm(gbm1,c(1,2,6),best.iter,cont=20)
plot.gbm(gbm1,1:3,best.iter)
plot.gbm(gbm1,2:4,best.iter)
plot.gbm(gbm1,3:5,best.iter)

# do another 100 iterations
gbm2 <- gbm.more(gbm1,100,
    verbose=FALSE) # stop printing detailed progress

---

**gbm.object**  
*Generalized Boosted Regression Model Object*

**Description**

These are objects representing fitted gbms.

**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 (e.g. log-odds scale for bernoulli, log scale for poisson)
- **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 gbm.perf
trees a list containing the tree structures. The components are best viewed using pretty.gbm.tree
c.splits a list of all the categorical splits in the collection of trees. If the trees[[i]] component of a gbm 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 gbm object.

Author(s)

Greg Ridgeway (gregr@rand.org)

See Also
gbm

gbm.perf GBM performance

Description

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

Usage

gbm.perf(object, plot.it = TRUE, oobag.curve = FALSE, overlay = TRUE, method)
Arguments

object: a `gbm.object` created from an initial call to `gbm`.

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 `gbm`.

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 `gbm` was called with `cv.folds>1`.

Value

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

Author(s)

Greg Ridgeway (gregr@rand.org)

References


See Also

`gbm`, `gbm.object`

Description

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

Usage

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

**Arguments**

- **x**: a *gbm.object* fitted using a call to *gbm*
- **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 *gbm* formula. If `length(i.var)` is between 1 and 3 then `plot.gbm` produces the plots. Otherwise, `plot.gbm` 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.gbm` 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.gbm` produces low dimensional projections of the *gbm.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.gbm` 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.gbm` produces no graphics and only returns the grid of evaluation points and their average predictions.

**Author(s)**

Greg Ridgeway (gregr@rand.org)

**References**


**See Also**

*gbm, gbm.object, plot*
Description

Predicted values based on a generalized boosted model object

Usage

```r
## S3 method for class 'gbm':
predict(object,
    newdata,
    n.trees,
    type="link",
    single.tree=FALSE,
    ...)  
```

Arguments

- `object`: Object of class inheriting from `gbm.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
- `type`: The scale on which gbm makes the predictions
- `single.tree`: If `single.tree=TRUE` then `predict.gbm` returns only the predictions from tree(s) `n.trees`
- `...`: further arguments passed to or from other methods

Details

`predict.gbm` produces predicted values for each observation in `newdata` using the first `n.trees` iterations of the boosting sequence. If `n.trees` is a vector then the result is a matrix with each column representing the predictions from `gbm` models with `n.trees[1]` iterations, `n.trees[2]` iterations, and so on.

The predictions from `gbm` 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 `gbm.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). For example, for the Bernoulli loss the returned value is on the log odds scale, poisson loss on the log scale, and coxph is on the log hazard scale.

If `type="response"` then `gbm` converts back to the same scale as the outcome. Currently the only effect this will have is returning probabilities for bernoulli and expected counts for poisson. For the other distributions "response" and "link" return the same.
**Description**

`gbm` stores the collection of trees used to construct the model in a compact matrix structure. This function extracts the information from a single tree and displays it in a slightly more readable form. This function is mostly for debugging purposes and to satisfy some users’ curiosity.

**Usage**

```r
pretty.gbm.tree(object, i.tree = 1)
```

**Arguments**

- `object` a `gbm.object` initially fit using `gbm`
- `i.tree` the index of the tree component to extract from `object` and display

**Value**

`pretty.gbm.tree` returns a data frame. Each row corresponds to a node in the tree. Columns indicate:

- **SplitVar**: index of which variable is used to split. -1 indicates a terminal node.
- **SplitCodePred**: if the split variable is continuous then this component is the split point. If the split variable is categorical then this component contains the index of `object$fc.split` that describes the categorical split. If the node is a terminal node then this is the prediction.
- **LeftNode**: the index of the row corresponding to the left node.
- **RightNode**: the index of the row corresponding to the right node.
- **ErrorReduction**: the reduction in the loss function as a result of splitting this node.
- **Weight**: the total weight of observations in the node. If weights are all equal to 1 then this is the number of observations in the node.

**Author(s)**

Greg Ridgeway (gregr@rand.org)

**See Also**

`gbm`, `gbm.object`
quantile.rug  Quantile rug plot

Description

Marks the quantiles on the axes of the current plot.

Usage

quantile.rug(x, prob=(1:10)/10,...)

Arguments

x  a numeric vector.
prob the quantiles of x to mark on the x-axis.
... additional graphics parameters currently ignored.

Details

Value

No return values

Author(s)

Greg Ridgeway (gregr@rand.org)

References

http://www.i-pensieri.com/gregr/gbm.shtml

See Also

plot, quantile.jitter.rug.

Examples

x <- rnorm(100)
y <- rnorm(100)
plot(x, y)
quantile.rug(x)
Methods for estimating relative influence

Description

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

Usage

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

Arguments

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

Value

Returns an unprocessed vector of estimated relative influences.

Author(s)

Greg Ridgeway (gregr@rand.org)

References


See Also

summary.gbm
shrink.gbm  L1 shrinkage of the predictor variables in a GBM

Description

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

Usage

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

Arguments

object A gbm.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)

Greg Ridgeway (gregr@rand.org)

References

shrink.gbm.pred

See Also

shrink.gbm.pred, gbm

Examples

shrink.gbm.pred  Predictions from a shrunken GBM

Description

Makes predictions from a shrunken GBM model.

Usage

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

Arguments

object  a gbm.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)

Greg Ridgeway (gregr@rand.org)

See Also

shrink.gbm, gbm

Examples
**summary.gbm**

**Summary of a gbm object**

**Description**

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

**Usage**

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

**Arguments**

- **object**
  a gbm object created from an initial call to `gbm`.

- **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.gbm`. 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 gbm currently computes using the entire training dataset (not the out-of-bag observations).

- **...**
  other arguments passed to the plot function.

**Details**

For `distribution="gaussian"` this returns exactly a set of Type III sum of squares for each variable normalized to sum to 100. For other loss functions this returns the reduction attributeable 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)

Greg Ridgeway (gregr@rand.org)

References


See Also

gbm
Index

*Topic aplot
  quantile.rug, 15

*Topic hplot
  calibrate.plot, 2
  plot.gbm, 12
  relative.influence, 16
  summary.gbm, 19

*Topic methods
  basehaz.gbm, 1
  gbm.object, 10
  shrink.gbm, 17
  shrink.gbm.pred, 18

*Topic models
  gbm, 5
  predict.gbm, 13

*Topic nonlinear
  gbm, 5
  gbm.perf, 11

*Topic nonparametric
  gbm, 5
  gbm.perf, 11

*Topic package
  gbm-package, 4

*Topic print
  pretty.gbm.tree, 14

*Topic regression
  predict.gbm, 13

*Topic survival
  basehaz.gbm, 1
  gbm, 5
  gbm.perf, 11

*Topic tree
  gbm, 5
  gbm.perf, 11

basehaz.gbm, 1

calibrate.plot, 2

gbm, 2, 5, 7, 11–16, 18–20
gbm-package, 4
gbm.fit, 14

gbm.loss(relative.influence), 16

gbm.more, 6, 7
gbm.object, 7, 8, 10, 11–15, 17, 18
gbm.perf, 8, 11, 11
jitter, 16
lattice, 13
model.frame, 7
permutation.test.gbm, 20
permutation.test.gbm(relative.influence), 16
plot, 13, 16
plot.gbm, 8, 12
polygon, 3
predict.gbm, 8, 13
pretty.gbm.tree, 8, 11, 14
quantile, 16
quantile.rug, 15
relative.influence, 16, 20
rug, 3, 16
save, 7
shrink.gbm, 17, 19
shrink.gbm.pred, 18, 18
summary.gbm, 8, 16, 17, 19
supsmu, 2
survfit, 2

22