Package ‘l2boost’

October 13, 2022

Type Package
Title Exploring Friedman's Boosting Algorithm for Regularized Linear Regression
Version 1.0.3
Date 2022-02-10
Description Efficient implementation of Friedman's boosting algorithm with l2-loss function and coordinate direction (design matrix columns) basis functions.
License GPL (>= 2)
Depends R (>= 2.14.0), MASS, parallel
Author John Ehrlinger [aut, cre], Hemant Ishwaran [aut]
Maintainer John Ehrlinger <john.ehrlinger@gmail.com>
RoxygenNote 7.1.2
LazyData true
NeedsCompilation no
Repository CRAN
Date/Publication 2022-02-11 12:10:05 UTC

R topics documented:

l2boost-package .................................................. 2
coefl2boost .................................................... 3
cv.l2boost ....................................................... 4
diabetes .......................................................... 7
elasticNetSim .................................................... 8
error.bars ........................................................ 9
fitted.l2boost ................................................... 10
l2boost ......................................................... 11
mvnorm.l2boost ............................................... 15
plot.l2boost ................................................... 17
Efficient implementation of Friedman's boosting algorithm for linear regression using an l2-loss function and coordinate direction (design matrix columns) basis functions.

Description

The l2boost package implements a generic boosting method [Friedman (2001)] for linear regression settings using an l2-loss function. The basis functions are simply the column vectors of the design matrix. **l2boost** scales the design matrix such that the boosting coefficients correspond to the gradient direction for each covariate. Friedman's gradient descent boosting algorithm proceeds at each step along the covariate direction closest (in L2 distance) to the maximal gradient descent direction.

The **l2boost** function uses an arbitrary L1-regularization parameter (nu), and includes the elementary data augmentation of Ehrlinger and Ishwaran (2012), to add an L2-penalization (lambda) similar to the elastic net [Zou and Hastie (2005)]. The L2-regularization reverses repressibility, a condition where one variable acts as a boosting surrogate for other, possibly informative, variables. Along with the decorrelation effect, this elasticBoost regularization circumvents L2Boost deficiencies in correlated settings.

We include a series of S3 functions for working with **l2boost** objects:

- **print** (**print.l2boost**) prints a summary of the l2boost model fit.
- **coef** (**coef.l2boost**) returns the model regression coefficients at any point along the solution path indexed by step m.
- **fitted** (**fitted.l2boost**) returns the fitted response values from the training set at any point along the solution path.
- **residuals** (**residuals.l2boost**) returns the training set residuals at any point along the solution path.
- **plot** (**plot.l2boost**) for graphing either beta coefficients or gradient-correlation as a function of boosting steps.
- **predict** (**predict.l2boost**) for boosting prediction on possibly new observations at any point along the solution path.

A cross-validation method (**cv.l2boost**) is also included for L2boost and elasticBoost cross-validating regularization parameter optimizations.
Example Datasets We have repackaged the diabetes data set from Efron et. al. (2004) for demonstration purposes. We also include data simulation functions for reproducing the elastic net simulation (elasticNetSim) of Zou and Hastie (2005) and the example multivariate normal simulations (mvnorm.l2boost) of Ehrlinger and Ishwaran (2012).

References


 coef.l2boost

**Extract model coefficients from an l2boost model object at any point along the solution path indexed by step m. coef is a generic function which extracts model coefficients from objects returned by modeling functions.**

Description
By default, coef.l2boost returns the model (beta) coefficients from the last step, M of the l2boost model. For a cv.l2boost object, the default returns the coefficients from model at the cross-validation optimal step (m = opt.step return value).

Coefficients from alternative steps along the solution can be obtained using the m parameter.

Usage
```
## S3 method for class 'l2boost'
coef(object, m = NULL, ...)
```

Arguments
- `object` an l2boost fit object (l2boost or cv.l2boost)
- `m` the iteration number within the l2boost solution path. If m=NULL, the coefficients are obtained from the last iteration M.
- `...` other arguments passed to generic function.

Value
vector of coefficient estimates for l2boost objects. The estimates correspond to the given iteration number m, or the final step M.
See Also

coef and l2boost, cv.l2boost and predict.l2boost methods of l2boost.

Examples

# Example: Diabetes data
# See Efron B., Hastie T., Johnstone I., and Tibshirani R.
data(diabetes, package='l2boost')

object <- l2boost(diabetes$x, diabetes$y, M=1000, nu=.01)
coef(object)

# At the m=500 step
coef(object, m=500)

Description

Calculate the K-fold cross-validation prediction error for l2boost models. The prediction error is calculated using mean squared error (MSE). The optimal boosting step \( m=\text{opt.step} \) is obtained by selecting the step \( m \) resulting in the minimal MSE.

Usage

```r
cv.l2boost(
  x,
  y,
  K = 10,
  M = NULL,
  nu = 1e-04,
  lambda = NULL,
  trace = FALSE,
  type = c("discrete", "hybrid", "friedman", "lars"),
  cores = NULL,
  ...
)
```

Arguments

- **x**: the design matrix
- **y**: the response vector
cv.l2boost

K  number of cross-validation folds (default: 10)
M  the total number of iterations passed to `l2boost`.
nu  l1 shrinkage parameter (default: 1e-4)
lambda  l2 shrinkage parameter for elasticBoost (default: NULL = no l2-regularization)
trace  Show computation/debugging output? (default: FALSE)
type  Type of l2boost fit with (default: discrete) see `l2boost` for description.
cores  number of cores to parallel the cv analysis. If not specified, detects the number
       of cores. If more than 1 core, use n-1 for cross-validation. Implemented using
       multicore (mclapply), or clusterApply on Windows machines.

Details

The cross-validation method splits the test data set into K mutually exclusive subsets. An `l2boost`
model is built on K different training data sets, each created from a subsample of the full data set
by sequentially leaving out one of the K subsets. The prediction error estimate is calculated by
averaging the mean square error of each K test sets of the all of the K training datasets. The optimal
step \(m\) is obtained at the step with a minimal averaged mean square error.

The full `l2boost` model is run after the cross-validation models, on the full data set. This model is
run for the full number of iteration steps \(M\) and returned in the cv.l2boost$fit object.

cv.l2boost only optimizes along the iteration count \(m\) for a given value of \(nu\). This is equiv-
alent to an L1-regularization optimization. In order to optimize an elasticBoost model on the L2-
regularization parameter lambda, a manual two way cross-validation can be obtained by sequen-
tially optimizing over a range of lambda values, and selecting the lambda/opt.step pair resulting in
the minimal cross-validated mean square error. See the examples below.

cv.l2boost uses the parallel package internally to speed up the cross-validation process on multi-
core machines. Parallel is packaged with base R >= 2.14, for earlier releases the multicore package
provides the same functionality. By default, cv.l2boost will use all cores available except 1. Each
fold is run on it’s own core and results are combined automatically. The number of cores can be
overridden using the cores function argument.

Value

A list of cross-validation results:

call  the matched call.
type  Choice of l2boost algorithm from "discrete", "hybrid", "friedman","lars". see
       `l2boost`
names  design matrix column names used in the model
nu  The L1 boosting shrinkage parameter value
lambda  The L2 elasticBoost shrinkage parameter value
K  number of folds used for cross-validation
mse  Optimal cross-validation mean square error estimate
mse.list  list of K vectors of mean square errors at each step \(m\)
cv.l2boost

- **coef**: Beta coefficient estimates from the full model at opt.step
- **coef.stand**: Standardized beta coefficient estimates from full model at opt.step
- **opt.step**: Optimal step m calculated by minimizing cross-validation error among all K training sets
- **opt.norm**: L1 norm of beta coefficients at opt.step
- **fit**: L2boost fit of full model
- **yhat**: Estimate of response from full model at opt.step

**See Also**

- l2boost, plot.l2boost, predict.l2boost, and coef.l2boost

**Examples**

```r
## Not run:
# Example: ElasticBoost simulation
# Compare l2boost and elasticNetBoosting using 10-fold CV
#
# Elastic net simulation, see Zou H. and Hastie T. Regularization and
# variable selection via the elastic net. J. Royal Statist. Soc. B,
# 67(2):301-320, 2005
# set.seed(1025)
dta <- elasticNetSim(n=100)
# The default values set up the signal on 3 groups of 5 variables,
# Color the signal variables red, others are grey.
sig <- c(rep("red", 15), rep("grey", 40-15))

# Set the boosting parameters
Mtarget = 1000
nuTarget = 1.e-2
# For CRAN, only use 2 cores in the CV method
cvCores=2

cv.obj <- cv.l2boost(dta$x, dta$y, M=Mtarget, nu=nuTarget, cores=cvCores)
# Plot the results
par(mfrow=c(2,3))
plot(cv.obj)
abline(v=cv.obj$opt.step, lty=2, col="grey")
plot(cv.obj$fit, type="coef", ylab=expression(beta[i]))
abline(v=cv.obj$opt.step, lty=2, col="grey")
plot(coef(cv.obj$fit, m=cv.obj$opt.step), cex=.5,
     ylab=expression(beta[i]), xlab="Column Index", ylim=c(0,140), col=sig)

# elasticBoost l1-regularization parameter lambda=0.1
# 5 fold elasticNet CV
cv.eBoost <- cv.l2boost(dta$x, dta$y, M=Mtarget, K=5, nu=nuTarget, lambda=.1, cores=cvCores)
```

# plot the results
plot(cv.eBoost)
apline(v=cv.eBoost$opt.step, lty=2, col="grey")
plot(cv.eBoost$fit, type="coef", ylab=expression(beta[i]))
apline(v=cv.eBoost$opt.step, lty=2, col="grey")
plot(coef(cv.eBoost$fit, m=cv.obj$opt.step), cex=.5,
     ylab=expression(beta[i]), xlab="Column Index", ylim=c(0,140), col=sig)

## End(Not run)

diabetes  Blood and other measurements in diabetics [Hastie and Efron (2012)]

Description

A repackaged diabetes dataset [Hastie and Efron (2012)] is a list of two different design matrices and a response vector with 442 observations [Efron et. al. (2004)].

The x matrix has been standardized to have unit L2 norm in each column and zero mean. The matrix x2 consists of x plus 54 interaction terms.

Usage

diabetes

Format

A list of 3 data objects,

- x: A data frame with 10 variables and 442 observations
- y: a numeric response vector of 442 observations
- x2: a design matrix including interaction terms with 64 columns and 442 observations.

References


elasticNetSim  A blocked correlated data simulation.

Description

Creates a data simulation of \( n \) observations with \( \text{signal} \) groups of \( (p_0/\text{signal}) \) signal variables and \( (p-p_0) \) noise variables. Random noise is added to all columns. The default values, with \( n=100 \) create the simulation of Zou and Hastie (2005).

Usage

```r
elasticNetSim(  
n,  
p = 40,  
p0 = 15,  
signal = 3,  
sigma = sqrt(0.01),  
beta.true = NULL  
)
```

Arguments

- \( n \): number of observations
- \( p \): number of coordinate directions in the design matrix (default 40)
- \( p_0 \): number of signal coordinate directions in the design matrix (default 15)
- \( \text{signal} \): number of signal groups (default 3)
- \( \sigma \): within group correlation coefficient (default \( \text{sqrt}(0.01) \))
- \( \beta.true \): specify the true simulation parameters. (default NULL = generated from other arguments)

Value

list of

- \( x \) simulated design matrix
- \( y \) simulated response vector
- \( \beta.true \) true beta parameters used to create the simulation

References

Examples

# Example: Elastic net simulation
#
# For elastic net simulation data, see Zou, H. and Hastie, T. (2005)
# Regularization and variable selection via the elastic net J. R. Statist. Soc. B
# , 67, Part 2, pp. 301-320
# Set the RNG seed to create a reproducible simulation
# Takes an integer argument
#
set.seed(432) # Takes an integer argument
#
# Create a simulation with 100 observations.
dta <- elasticNetSim(n=100)

# The simulation contains a design matrix x, and response vector y
dim(dta$x)
length(dta$y)
print(dta$x[1:5,])

error.bars

Description

nice standard errors for plots

Usage

error.bars(x, upper, lower, width = 0.001, max.M = 100, ...)

Arguments

x Vector of error bar x value locations
upper Vector of upper error bar limits
lower Vector of limit error bar limits
width errorbar line width (default: 0.001)
max.M maximum number of bars to show in a plot (default: 100)
... Additional arguments passed to segment function

See Also

segments
fitted.l2boost  

Extract the fitted model estimates along the solution path for an l2boost model.

Description

fitted is a generic function which extracts fitted values from objects returned by modeling functions.

Usage

## S3 method for class 'l2boost'
fitted(object, m = NULL, ...)

Arguments

- object: an l2boost object
- m: the iteration number with the l2boost path. (default m=NULL)
- ...: other arguments

Details

fitted.l2boost returns the function estimates obtained from the training set observations of an l2boost model object at any point along the solution path. The estimate, \( F_m(x) \) is evaluated at iteration \( m \) using the training data set \( x \). By default, fitted.l2boost returns the estimate at the last iteration step \( M \), unless a specific iteration step \( m \) is specified.

Value

The vector of fitted response estimates at the given iteration \( m \). By default, the coefficients are obtained from the last iteration \( m=M \).

See Also

fitted and l2boost and predict.l2boost

Examples

# Example: Diabetes
#
# See Efron B., Hastie T., Johnstone I., and Tibshirani R.
data(diabetes, package="l2boost")

l2.object <- l2boost(diabetes$x, diabetes$y, M=1000, nu=.01)

# return the fitted values
fitted(l2.object)
fitted(l2.object, m=500)

#' # Create diagnostic plots
par(mfrow=c(2,2))
qqnorm(fitted(l2.object), ylim=c(0, 300))
qqline(fitted(l2.object), col=2)

qqnorm(fitted(l2.object, m=500), ylim=c(0, 300))
qqline(fitted(l2.object, m=500), col=2)

# Tukey-Anscombe's plot
plot(y=residuals(l2.object), x=fitted(l2.object), main="Tukey-Anscombe's plot",
ylim=c(-3e-13, 3e-13))
lines(smooth.spline(fitted(l2.object), residuals(l2.object), df=4), type="l",
1ty=2, col="red", lwd=2)
abline(h=0, lty=2, col = 'gray')

plot(y=residuals(l2.object, m=500), x=fitted(l2.object, m=500),
main="Tukey-Anscombe's plot", ylim=c(-3e-13, 3e-13))
lines(smooth.spline(fitted(l2.object,m=500), residuals(l2.object, m=500), df=4),
type="l", lty=2, col="red", lwd=2)
abline(h=0, lty=2, col = 'gray')

---

l2boost  
Generic gradient descent boosting method for linear regression.

Description

Efficient implementation of Friedman's boosting algorithm [Friedman (2001)] with L2-loss function and coordinate direction (design matrix columns) basis functions. This includes the elasticNet data augmentation of Ehrlinger and Ishwaran (2012), which adds an L2-penalization (lambda) similar to the elastic net [Zou and Hastie (2005)].

Usage

l2boost(x, ...)

## Default S3 method:
12boost(x, y, M, nu, lambda, trace, type , qr.tolerance, eps.tolerance, ...)

## S3 method for class 'formula'
12boost(formula, data, ...)

Arguments

x  
design matrix of dimension n x p

...  
other arguments (currently unused)
The \texttt{l2boost} function is an efficient implementation of a generic boosting method [Friedman (2001)] for linear regression using an L2-loss function. The basis functions are the column vectors of the design matrix. \texttt{l2boost} scales the design matrix such that the coordinate columns of the design correspond to the gradient directions for each covariate. The boosting coefficients are equivalent to the gradient-correlation of each covariate. Friedman's gradient descent boosting algorithm proceeds at each step along the covariate direction closest (in L2 distance) to the maximal gradient descent direction.

We include a series of algorithms to solve the boosting optimization. These are selected through the \texttt{type} argument

- \texttt{friedman} - The original, bare-bones \texttt{l2boost} (Friedman (2001)). This method takes a fixed step size of length \texttt{nu}.
- \texttt{lars} - The \texttt{l2boost}-\texttt{lars}-limit (See Efron et.al (2004)). This algorithm takes a single step of the optimal length to the critical point required for a new coordinate direction to become favorable. Although optimal in the number of steps required to reach the OLS solution, this method may be computationally expensive for large p problems, as the method requires a matrix inversion to calculate the step length.
- \texttt{discrete} - Optimized Friedman algorithm to reduce number of evaluations required [Ehrlinger and Ishwaran 2012]. The algorithm dynamically determines the number of steps of length \texttt{nu} to take along a descent direction. The discrete method allows the algorithm to take step sizes of multiples of \texttt{nu} at any evaluation.
- \texttt{hybrid} - Similar to discrete, however only allows combining steps along the first descent direction. \texttt{hybrid} Works best if \texttt{nu} is moderate, but not too small. In this case, Friedman's algorithm would take many steps along the first coordinate direction, and then cycle when multiple coordinates have similar gradient directions (by the L2 measure).
L2boost keeps track of all gradient-correlation coefficients (\( \rho \)) at each iteration in addition to the maximal descent direction taken by the method. Visualizing these coefficients can be informative of the inner workings of gradient boosting (see the examples in the `plot.l2boost` method).

The L2boost function uses an arbitrary L1-regularization parameter (nu), and includes the elementary data augmentation of Ehrlinger and Ishwaran (2012), to add an L2-penalization (lambda) similar to the elastic net [Zou and Hastie (2005)]. The L2-regularization reverses repressibility, a condition where one variable acts as a boosting surrogate for other, possibly informative, variables. Along with the decorrelation effect, this elasticBoost regularization circumvents L2Boost deficiencies in correlated settings.

We include a series of S3 functions for working with L2boost objects:

- `print (print.l2boost)` prints a summary of the L2boost fit.
- `coef (coef.l2boost)` returns the L2boost model regression coefficients at any point along the solution path.
- `fitted (fitted.l2boost)` returns the fitted L2boost response estimates (from the training dataset) along the solution path.
- `residuals (residuals.l2boost)` returns the training set L2boost residuals along the solution path.
- `plot (plot.l2boost)` for graphing model coefficients of an L2boost object.
- `predict (predict.l2boost)` for generating L2boost prediction estimates on possibly new test set observations.

A cross-validation method (`cv.l2boost`) is also included for L2boost and elasticBoost, for cross-validated error estimates and regularization parameter optimizations.

**Value**

A "l2boost" object is returned, for which print, plot, predict, and coef methods exist.

- `call` the matched call.
- `type` Choice of l2boost algorithm from "friedman", "discrete", "hybrid", "lars"
- `nu` The L1 boosting shrinkage parameter value
- `lambda` The L2 elasticNet shrinkage parameter value
- `x` The training dataset
- `x.na` Columns of original design matrix with values na, these have been removed from x
- `x.attr` scale attributes of design matrix
- `names` Column names of design matrix
- `y` training response vector associated with x, centered about the mean value ybar
- `ybar` mean value of training response vector
- `mjk` measure to favorability. This is a matrix of size p by m. Each coordinate j has a measure at each step m
- `stepSize` vector of step lengths taken (NULL unless type = "lars")
- `l.crit` vector of column index of critical direction
L\textsubscript{crit} \hspace{1cm} \text{number of steps along each l\textsubscript{crit} direction}

S\textsubscript{crit} \hspace{1cm} \text{The critical step value where a direction change occurs}

path.Fm \hspace{1cm} \text{estimates of response at each step m}

Fm \hspace{1cm} \text{estimate of response at final step M}

rhom.path \hspace{1cm} \text{boosting parameter estimate at each step m}

betam.path \hspace{1cm} \text{beta parameter estimates at each step m. List of m vectors of length p}

betam \hspace{1cm} \text{beta parameter estimate at final step M}

The notation for the return values is described in Ehrlinger and Ishwaran (2012).

References


See Also

\texttt{print.l2boost, plot.l2boost, predict.l2boost, coef.l2boost, residuals.l2boost, fitted.l2boost}

methods of l2boost and \texttt{cv.l2boost} for K fold cross-validation of the l2boost method.

Examples

# Example 1: Diabetes data

data(diabetes, package="l2boost")

l2.object <- l2boost(diabetes$x,diabetes$y, M=1000, nu=.01)

# Plot the boosting rho, and regression beta coefficients as a function of
# boosting steps m
#
# Note: The selected coordinate trajectories are colored in red after selection, and
# blue before. Unselected coordinates are colored grey.
#
par(mfrow=c(2,2))
plot(l2.object)
plot(l2.object, type="coef")

# increased shrinkage and number of iterations.

l2.shrink <- l2boost(diabetes$x,diabetes$y,M=5000, nu=1.e-3)
mvnorm.l2boost

multiivariate normal data simulations.

Description

Create simulated dataset from a multivariate normal. Used to recreate data simulations from Ehrlinger and Ishwaran (2012).

Usage

mvnorm.l2boost(n = 100, p = 100, beta = NULL, which.beta = NULL, rho = 0)
mvnorm.l2boost

Arguments

- **n**: number of observations
- **p**: number of coordinate directions in the design matrix
- **beta**: a "true" beta vector of length p (default=NULL) See details.
- **which.beta**: indicator vector for which beta coefficients to include as signal in simulation (default=NULL) see details
- **rho**: correlation coefficient between coordinate directions

Details

By default, mvnorm.l2boost creates a data set of n multivariate normal random observations of p covariates (see MASS:mvrnorm). The correlation matrix is constructed with 1 on the diagonals and the correlation coefficient \( \rho \) on the off diagonals.

The response is constructed as follows: If a true beta vector is not supplied, the first 10 beta coefficients carry the signal with a value of 5, and the remaining p-10 values are set to zero. Given a \( \text{beta.true} \) vector, all values are used as specified. The coefficient vector is truncated to have \( p \) signal terms if \( \text{length} (\text{beta.true}) > p \), and noise coordinates are added if \( \text{length} (\text{beta.true}) < p \).

It is possible to pass an indicator vector \( \text{which.beta} \) to select specific signal elements from the full vector \( \text{beta.true} \).

Value

- call Matched function call
- **x**: design matrix of size \( n \times p \)
- **y**: response vector of length \( n \)

References


Examples

```r
# Example: Multivariate normal data simulation

# Create a (reproducible) data set of size 100 x 100
set.seed(1024)
n<- 100
p<- 100

# Set 10 signal variables using a uniform beta=5, the remaining (p-10)=90 are
# set to zero indicating random noise.
beta <- c(rep(5,10), rep(0,p-10))

# Example with orthogonal design matrix columns (orthogonal + noise)
ortho.data <- mvnorm.l2boost(n, p, beta)
cbind(ortho.data$y[1:5],ortho.data$x[1:5])
```
# Example with correlation between design matrix columns
corr.data <- mvnorm.l2boost(n, p, beta, rho=0.65)
cbind(corr.data$y[1:5], corr.data$x[1:5,])

plot.l2boost

Plotting for l2boost objects.

Description

plotting methods for l2boost objects (l2boost and cv.l2boost).
By default, plotting an l2boost object produces a gradient-correlation vs iteration steps (m) plot. Plotting a cv.l2boost object produces a cross-validation error plot, and prints the minimal CV MSE value and optimal step opt.step to the R console.
Many generic arguments to plot are passed through the plot.l2boost function.

Usage

## S3 method for class 'l2boost'
plot(
  x,
  type = c("rho", "coef"),
  standardize = TRUE,
  active.set = NULL,
  xvar = c("step", "norm"),
  xlab = NULL,
  ylab = NULL,
  trim = TRUE,
  clip = NULL,
  col = NULL,
  ylim = NULL,
  xlim = NULL,
  ...
)

Arguments

x  l2boost or cv.l2boost object

type which type of plot. rho plots gradient-correlation, coef regression (beta) coefficients vs the step number m along the x-axis

standardize Should we plot standardized gradient-correlation (default: TRUE)

active.set Vector of indices of the coordinates for highlighting with color=col (default: NULL shows all active coordinates)

xvar what measure do we plot on the x-axis? step plots the step m, norm plots the normalized distance (1-\nu)^(m-1)
plot.l2boost

xlab  specific x-axis label (NULL results in default value depending on xvar)
ylab  specific y-axis label (NULL results in default value depending on type)
trim  (default: TRUE)
clip  Do we want to c
col   Color to highlight active.set coordinates (NULL indicates default all active set at step M in blue, changes to red after selection)
ylim  Control plotted y-values (default: NULL for auto range)
xlim  Control plotted x-values (default: NULL for auto domain )
...  other arguments passed to plot functions

Details

Gradient-correlation plots are created by tracing out the boosting coefficient (rho) for each candidate direction. The coefficient and gradient-correlation are equivalent under standard scaling (zero intercept with design matrix columns scaled to have mean=0 and variance=1).

Unless explicitly set using col argument, the plot function colors the gradient-correlation paths along each direction by the following criteria:

- Red: indicates the coordinate direction has been selected in the boosting path at some step <= m.
- Blue: indicates the coordinate will be selected within the specified number of steps M (and switch to red upon selection).
- Grey: indicates coordinates have not and will not be selected by the algorithm over all iterations.

The colors are set using the l.crit return value from the l2boost object.

Value

NULL

See Also

l2boost, print.l2boost, predict.l2boost methods of l2boost and cv.l2boost

Examples

# Example: Diabetes
#
# See Efron B., Hastie T., Johnstone I., and Tibshirani R.
data(diabetes, package = "l2boost")

l2.object <- l2boost(diabetes$x, diabetes$y, M=1000, nu=.01)

# Plot the gradient-correlation, and regression beta coefficients as a function of
# boosting steps m
par(mfrow=c(2,2))
plot(l2.object)
abline(v=500, lty=2, col="grey")
plot(l2.object, type="coef")
abline(v=500, lty=2, col="grey")

# limit the plot to only the first 500 steps of the algorithm
# (grey vertical line in previous plots).
plot(l2.object, xlim=c(0,500))
plot(l2.object, type="coef", xlim=c(0,500))

## Not run:
#--------------------------------------------------------------------------
# Example: Plotting cross-validation objects
dta <- elasticNetSim(n=100)
# Set the boosting parameters
Mtarget = 1000
nuTarget = 1.e-2
cv.l2 <- cv.l2boost(dta$x,dta$y, M=Mtarget, nu=nuTarget, lambda=NULL)

# Show the CV MSE plot, with a marker at the "optimal iteration"
plot(cv.l2)
abline(v=cv.l2$opt.step, lty=2, col="grey")

# Show the l2boost object plots.
plot(cv.l2$fit)
abline(v=cv.l2$opt.step, lty=2, col="grey")
plot(cv.l2$fit, type="coef")
abline(v=cv.l2$opt.step, lty=2, col="grey")

# Create a color vector of length p=40 (from elasticNetSim defaults)
clr <- rep("black", 40)
# Set coordinates in the boosting path to color red.
clr[unique(cv.l2$fit$l.crit)] = "red"

# Show the "optimal" coefficient values,
# red points are selected in boosting algorithm.
plot(coef(cv.l2$fit, m=cv.l2$opt.step), col=clr, ylab=expression(beta))

## End(Not run)

plot.lines is used by plot.l2boost to the path lines (each j, against each r-step)

Description
plots.lines is used by plot.l2boost to the path lines (each j, against each r-step)
Usage

## S3 method for class 'lines'
plot(xval = NULL, ind, path, l.crit, active = TRUE, col = NULL)

Arguments

xval vector of x-values corresponding to the path y-values (default: NULL index of path)
ind Coordinate of the path (for coloring individual paths)
path Plot the path values along the y-axis
l.crit change the color at the value of $m=l.crit$
active active set coloring (default: TRUE)
col vector of color values length $\geq 1$ (default: NULL use built in scheme)

predict.l2boost predict method for l2boost models.

Description

predict is a generic function for predictions from the results of various model fitting functions.

@details predict.l2boost takes the optional $xnew$ (equivalent $newdata$) data.frame and returns the model estimates from an l2boost object. If neither $xnew$ or $newdata$ are provided, predict returns estimates for the l2boost training data set.

By default, predict.l2boost returns the function estimates, unless type="coef" then the set of regression coefficients (beta) are returned from the l2boost object.

Usage

## S3 method for class 'l2boost'
predict(object, xnew = NULL, type = c("fit", "coef"), newdata = xnew, ...)

Arguments

object an l2boost object
xnew a new design matrix to fit with the l2boost object
type "fit" or "coef" determines the values returned. "fit" returns model estimates, "coef" returns the model coefficients
newdata a new design matrix to fit with the l2boost object
... other arguments (currently not used)
**predict.l2boost**

**Value**

- `yhat`: vector of `n` function estimates from the final step `M`
- `yhat.path`: list of `M` function estimates, one at each step `m`

or

- `coef`: vector of `p` beta coefficient estimates from final step `M`
- `coef.stand`: vector of `p` standardized beta coefficient estimates from final step `M`
- `coef.path`: list of vectors of `p` beta coefficient estimates, one for each step `m`
- `coef.stand.path`: list of vectors of `p` standardized beta coefficient estimates, one for each step `m`

**See Also**

`predict` and `l2boost`, `coef.l2boost`, `fitted.l2boost`, `residuals.l2boost` and `cv.l2boost`

**Examples**

```r
# Example 1: Diabetes
# See Efron B., Hastie T., Johnstone I., and Tibshirani R.
data(diabetes)

object <- l2boost(diabetes$x, diabetes$y, M=1000, nu=.01)

# With no arguments returns the estimates at the full `M` from the training data.
prd <- predict(object)
prd$yhat

# at step `m`=600
prd$yhat.path[[600]]

# Also can return coefficient estimates. This is equivalent to `\code{\link{coef.l2boost}}`
cf <- predict(object, type="coef")

# at step `m`=600
cf$coef.path[[600]]

# Or used to predict new data, in this case a subset of training data
cbind(diabetes$y[1:5], predict(object, xnew=diabetes$x[1:5,])$yhat)
```
print.12boost

print method for 12boost and cv.12boost objects.

Description

print is a generic function for displaying model summaries

print.12boost returns a model summary for 12boost and cv.12boost objects including the coefficient estimates at the specified step m. By default, print.12boost returns the summary for the object at the final iteration step M

Usage

## S3 method for class '12boost'
print(x, m = NULL, ...)

Arguments

x
  an 12boost object

m
  return the result from iteration m

... other arguments passed to helper functions

See Also

12boost, cv.12boost and coef.12boost

Examples

# Example 1: Diabetes
#
# See Efron B., Hastie T., Johnstone I., and Tibshirani R.
data(diabetes)

object <- 12boost(diabetes$x, diabetes$y, M=1000, nu=.01)

# A summary of the 12boost object at M=1000
print(object)

# Similar at m=100
print(object, m=100)
print.summary.l2boost

Unimplemented generic function These are placeholders right now.

Description

Unimplemented generic function These are placeholders right now.

Usage

```r
## S3 method for class 'summary.l2boost'
print(x, ...)
```

Arguments

- `x`: an l2boost object
- `...`: other arguments (not used)

residuals.l2boost

Model residuals for the training set of an l2boost model object

Description

`residuals` is a generic function which extracts model residuals from objects returned by modeling functions.

`residuals.l2boost` returns the training set residuals from an `l2boost` object. By default, the residuals are returned at the final iteration step `m=M`.

Usage

```r
## S3 method for class 'l2boost'
residuals(object, m = NULL, ...)
```

Arguments

- `object`: an l2boost object for the extraction of model coefficients.
- `m`: the iteration number with the l2boost path. If `m=NULL`, the coefficients are obtained from the last iteration M.
- `...`: arguments (unused)

Value

a vector of n residuals
See Also

residuals and l2boost and predict.l2boost

Examples

# Example: Diabetes

# For diabetes data set, see Efron B., Hastie T., Johnstone I., and Tibshirani R.
data(diabetes, package = "l2boost")

l2.object <- l2boost(diabetes$x, diabetes$y, M=1000, nu=.01)
rsd<-residuals(l2.object)
rss.mid <- residuals(l2.object, m=500)

# Create diagnostic plots
par(mfrow=c(2,2))
qqnorm(residuals(l2.object), ylim=c(-3e-13, 3e-13))
qqline(residuals(l2.object), col=2)

qqnorm(residuals(l2.object, m=500), ylim=c(-3e-13, 3e-13))
qqline(residuals(l2.object, m=500), col=2)

# Tukey-Anscombe's plot
plot(y=residuals(l2.object), x=fitted(l2.object), main="Tukey-Anscombe's plot",
     ylim=c(-3e-13, 3e-13))
lines(smooth.spline(fitted(l2.object), residuals(l2.object), df=4), type="l",
     lty=2, col="red", lwd=2)
abline(h=0, lty=2, col = 'gray')

plot(y=residuals(l2.object, m=500), x=fitted(l2.object, m=500), main="Tukey-Anscombe's plot",
     ylim=c(-3e-13, 3e-13))
lines(smooth.spline(fitted(l2.object,m=500), residuals(l2.object, m=500), df=4), type="l",
     lty=2, col="red", lwd=2)
abline(h=0, lty=2, col = 'gray')

summary.l2boost

Unimplemented generic function These are placeholders right now.

Description

Unimplemented generic function These are placeholders right now.

Usage

## S3 method for class 'l2boost'
summary(object, ...)

summary.l2boost Unimplemented generic function These are placeholders right now.
VAR

Arguments

  object          an l2boost object
    ...          other arguments (unused)

VAR

This is a hidden function of the l2boost package. VAR is a helper function that specifically returns NA if all values of the argument x are NA, otherwise, it returns a var object.

Description

This is a hidden function of the l2boost package. VAR is a helper function that specifically returns NA if all values of the argument x are NA, otherwise, it returns a var object.

Usage

VAR(x)

Arguments

  x          return variance of x matrix.
Index

* datasets
  diabetes, 7
  as.data.frame, 12
  coef, 2–4, 13
  coef.12boost, 2, 3, 6, 13, 14, 21, 22
  cv.12boost, 2–4, 4, 5, 13, 14, 17, 18, 21, 22
  data.frame, 20
  diabetes, 3, 7
  elasticNetSim, 3, 8
  error.bars, 9
  fitted, 2, 10, 13
  fitted.12boost, 2, 10, 10, 13, 14, 21
  formula, 12
  l2boost, 2–6, 10, 11, 12, 13, 17, 18, 20–24
  l2boost-package, 2
  mvnorm.12boost, 3, 15
  plot, 2, 13, 17
  plot.12boost, 2, 6, 13, 14, 17, 19
  plot.lines, 19
  predict, 2, 13, 20, 21
  predict.12boost, 2, 4, 6, 10, 13, 14, 18, 20, 20, 24
  print, 2, 13, 22
  print.12boost, 2, 13, 14, 18, 22, 22
  print.summary.12boost, 23
  qr.solve, 12
  residuals, 2, 13, 23, 24
  residuals.12boost, 2, 13, 14, 21, 23, 23
  segments, 9
  summary.12boost, 24

VAR, 25