Package 'autotune'

September 2, 2026

Type Package

Title Fast, Accurate and Automatic Tuning Parameter Selection for Lasso

Version 0.1.1

Maintainer Tathagata Sadhukhan <ts767@cornell.edu>

Description Fits Lasso paths for high-dimensional regression using coordinate descent with automatic, data-driven tuning of the regularization parameter. The implementation is 10 to 200 times faster than the standard 'glmnet' implementation of Lasso tuned via Cross Validation and over 100 times faster than scaled Lasso. It also provides a reliable estimate of the regression noise level and a new diagnostic for sparsity. For details of the method, see Sadhukhan, Wilms, Smeekes and Basu (2025) ``Autotune: fast, accurate, and automatic tuning parameter selection for Lasso'' <doi:10.48550/arXiv.2512.11139>.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 2.10)

Imports Rcpp (>= 1.0.13)

LinkingTo Rcpp

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, glmnet, AUC, ggplot2, ggExtra, dplyr, tidyr, Matrix

VignetteBuilder knitr

NeedsCompilation yes

Author Tathagata Sadhukhan [aut, cre], Ines Wilms [aut], Stephan Smeekes [aut], Sumanta Basu [aut]

Repository CRAN

Date/Publication 2026-09-02 16:20:02 UTC

Contents

`autotune-package` *Autotune: fast, accurate, and automatic tuning parameter selection for LASSO*

Description

This package fits lasso path for high-dimensional regression using coordinate descent while automatically tuning it's regularization parameter. The implementation is 10 to 200 times faster than the standard 'glmnet' implementation of Lasso tuned via Cross Validation and over 100 times faster than scaled Lasso. It also provides a reliable estimate of the regression noise level and a new diagnostic for sparsity.

Details

Package:	autotune
Type:	Package
Version:	1.0
Date:	2025-09-18

Simple to use. Accepts `x,y` data for linear regression model, and produces the regularization path which automatically estimates an optimal tuning parameter `lambda`.

Functions

Only 1 function, more autotune algorithms for logistic regression, graphical Lasso, group Lasso, VAR, random forests, coming soon!

- `autotune_lasso`

Author(s)

Tathagata Sadhukhan Ines Wilms Stephan Smeekes and Sumanta Basu
Maintainer: Tathagata Sadhukhan(ts767@cornell.edu)

Examples

```
set.seed(10)
n <- 80
p <- 500
s <- 5
type <- 1
snr <- 3
betatrue <- c(rep(1,s), rep(0, p - s))
x <- scale(matrix(rnorm(n * p), ncol = p))
error.sd <- sqrt((betatrue %*% betatrue)/snr)
err <- rnorm(n, sd = error.sd)
y <- x %*% betatrue + err
y <- y - mean(y)
ans <- autotune_lasso(x, y, trace_it = TRUE)
b <- betatrue
# The Predictors which are actually significant:
which(b != 0)
# The Predictors which had nonzero estmated coefficients:
which(ans$beta != 0)
# Top 10 predictors X_i's in the ranking of X_i's given by autotune:
ans$sorted_predictors[1:10] + 1
# No of significant predictors in each CD iteration when sigma_hat is allowed to vary:
ans$count_sig_beta
# Sigma estimates in each CD iteration:
ans$sigma2_seq
# Empirical noise variance:
var(err)
```

`autotune_lasso` *Data-adaptive automatic and fast tuning of LM with Lasso regularization*

Description

Fits a linear model via alternative optimization of penalized gaussian maximum likelihood which is a biconvex function of regression coefficients β and noise variance σ^2. The regularization path of `autotune_lasso` quickly picks out a good lambda for Lasso and then returns the corresponding linear fit along with various attributes related to the fit.

Usage

```
autotune_lasso(
  x,
  y,
  alpha = 0.01,
  standardize = TRUE,
  standardize_response = TRUE,
```

```
  intercept = TRUE,
  active = FALSE,
  trace_it = FALSE,
  tolerance = 1e-04,
  beta_tolerance = 0.001,
  iter_max = 30,
  beta_iter_max = 40,
  active_iter_max = 5,
  PR_norm_l2 = FALSE,
  ...
)
```

Arguments

x	Matrix of predictors, with one column for each predictor. Dimension will be nobs $\times$ nvars; so each row is a new observation and nvars > 1.
y	Vector of responses.
alpha	Default 0.01, significance level of sequential F-tests used for estimation of support set.
standardize	Logical flag for standardization of all variables in x, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default value is TRUE.
standardize_response	Logical flag for demeaning the reponse variable y. Default value is TRUE.
intercept	Should intercept(s) be fitted (default=TRUE) or set to zero (FALSE).
active	Should active set selection be used (TRUE) or avoided (default = FALSE). It is under experimentation phase, use it with care.
trace_it	Logical input, default FALSE; if TRUE prints out the iteration number while running, useful for big datasets that take a long time to fit.
tolerance	Numeric input for an additional stopping criteria on the coordinate descent when sigma is updating. Stops that coordinate descent when the relative change between successive iterates of coefficients' estimates is less than the tolerance. Default value is 10^{-4}.
beta_tolerance	Numeric input for the stopping criteria on the coordinate descent when sigma is not updating. Stops that coordinate descent when the relative change between successive iterates of coefficients' estimates is less than the beta_tolerance. Default value is 10^{-3}.
iter_max	Maximum number of iterations of coordinate descent allowed when sigma is updating. Default value is 30.
beta_iter_max	Maximum number of iterations of coordinate descent allowed when sigma is not updating. Default value is 40.
active_iter_max	If active = TRUE, maximum number of times the active set is updated as per the violations of KKT conditions. Default value is 5.
PR_norm_l2	Logical flag to whether use the l2 norm of partial residuals for ordering them instead of the default l1 norm.
...	Additional arguments passed to the internal fitting routine.

Value

A list with various intermediate and final outputs produced by autotune Lasso in its regularization path.

`beta` Final estimates of regression coefficients.

`a0` Intercept of the fit.

`lambda` Final thresholding value $\lambda = \lambda_0 \hat{\sigma}^2$ used in the coordinate descent after noise variance estimate $\hat{\sigma}^2$ has converged.

`sigma_sq` Final estimate of noise variance σ^2

`nobs` Number of observations.

`nvars` Number of variables.

`CD.path.details`

A list of additional details about the coordinate descent path taken by autotune Lasso:

`sorted_predictors` Decreasing ordering of predictors in terms of their contribution to predicting the response values.

`sigma_sq_seq` A `no_of_iterations`-length sequence of noise variance estimates σ^2.

`beta_matrix` A (`length(sigma_sq_seq) + 1`) $\times$ `nvars` matrix of estimated coefficients.

`no_of_iter_before_lambda_conv` Number of coordinate descent iterations performed before the noise variance estimate $\hat{\sigma}^2$ converged.

`no_of_iter_after_lambda_conv` After the noise variance estimate $\hat{\sigma}^2$ has converged, the number of coordinate descent iterations required for coefficients $\hat{\beta}$ to converge.

`no_of_iterations` Total number of coordinate descent iterations implemented by autotune Lasso.

`lambda0` Value of λ_0 used in autotune Lasso. Refer to the original paper for details.

`support_set` Final set of predictors included in the support set for noise variance estimation by autotune Lasso.

`count_sig_beta` A `no_of_iterations`-length vector containing the support set sizes across the coordinate descent iterations while the noise variance estimate is being updated.

`null_support` Boolean output indicating whether the final support set estimate is a null set. If so, `autotune_lasso` uses the support set estimate from the previous iteration to obtain the final noise variance estimate σ^2.

`active_iterations` If `active = TRUE`, the number of times the active set is updated according to violations of the KKT conditions.

`active_set_sizes` An `active_iterations`-length vector containing the active-set sizes used for coordinate descent across the active-set iterations.

Examples

```
library(autotune)
set.seed(10)
n <- 80
p <- 400
s <- 5
snr <- 4
betatrue <- c(rep(1,s), rep(0, p - s))
x <- matrix(rnorm(n * p), ncol = p)
error.sd <- sqrt((betatrue %*% betatrue)/snr)
err <- rnorm(n, sd = error.sd)
y <- x %*% betatrue + err
ans <- autotune_lasso(x, y, trace_it = TRUE)
b <- betatrue
# The Predictors which are actually significant:
which(b != 0)
# The Predictors which had nonzero estmated coefficients:
which(ans$beta != 0)
# Top 10 predictors X_i's in the ranking of X_i's given by autotune:
ans$CD.path.details$sorted_predictors[1:10]
# Cardinality of autotune's estimated support set in each CD iteration before lambda converged:
ans$CD.path.details$count_sig_beta
# Sigma estimates in each CD iteration:
ans$CD.path.details$sigma_sq_seq
# Empirical noise variance:
var(err)
```

---

coef.autotune_lasso *Extract coefficients for autotune_lasso objects*

Description

This function extracts the regression coefficients estimated by an autotune lasso model, using the stored "autotune_lasso" object

Usage

```
## S3 method for class 'autotune_lasso'
coef(object, intercept = TRUE, path = FALSE, sparse = TRUE, ...)

## S3 method for class 'autotune_lasso_path'
coef(object, intercept = TRUE, a0 = NULL, ...)
```

Arguments

object	An "autotune_lasso" object (for coef.autotune_lasso) or "autotune_lasso_path" object (for coef.autotune_lasso_path).

intercept	Logical; include the intercept term? If FALSE, the intercept row is omitted.
path	Logical; if TRUE, return the entire coefficient path across different lambdas.
sparse	Logical; controls the output format of coef.autotune_lasso() when path = FALSE. If TRUE, returns a sparse dgCMatrix (glmnet-style); if FALSE, returns a base numeric vector.
...	Other arguments to coef.
a0	Intercept of the fitted autotune regression. Used only when intercept = TRUE.

Value

- coef.autotune_lasso(path = FALSE, sparse = TRUE) returns a sparse dgCMatrix of dimension (nvars + 1) $\times 1$ (or nvars $\times 1$ if intercept = FALSE).
- coef.autotune_lasso(path = FALSE, sparse = FALSE) returns a named numeric vector.
- coef.autotune_lasso(path = TRUE) calls coef.autotune_lasso_path(), which returns a sparse dgCMatrix of dimension (nvars + 1) $\times$ (k + 1) (or nvars $\times$ (k + 1) if intercept = FALSE), with columns named lambda_1, ..., lambda_k, final_lambda.

Examples

```
set.seed(10)
n = 300
p = 500
s = 10
beta = c(rep(1, s), rep(0, p - s))
x = matrix(rnorm(n * p), ncol = p)
# Maunal sigma allocation
# y = x %*% beta + rnorm(n, sd = 1)
# Dynamic sigma allocation with snr specified
snr = 2
y = x %*% beta + rnorm(n, sd = sqrt(var(x%*%beta)/snr))
fit <- autotune_lasso(x, y)
coef(fit)                    # final solution
coef(fit, path = TRUE)        # coefficient path (sparse)
```

---

plot.autotune_lasso *Plots for autotune_lasso objects*

Description

Plots for autotune_lasso objects

Usage

```
## S3 method for class 'autotune_lasso'
plot(x, max_preds = NULL, cumulative = TRUE, ...)
```

Arguments

x	Fitted "autotune_lasso" object.
max_preds	Integer input stating maximum number of predictors to include in the plot. Default NULL (then max_preds is calculated internally). Plotting R-squared involves running max_preds many linear models, in order to reduce the computation, use it with the value ν when you want to check whether the dataset satisfies the sparsity assumption with $\leq \nu$ many predictors or not.
cumulative	Logical input. If TRUE, plots cumulative R-squared; if FALSE, plots adjusted R-squared.
...	Other graphical parameters to plot

Value

Invisibly returns a data frame with columns:

n_predictors Number of predictors included in the nested linear model.

predictor_index Index of the predictor added at each step.

has_nonzero_beta Logical value indicating whether the predictor added at each step has a non-zero coefficient.

r_squared Cumulative R-squared value.

adj_r_squared Adjusted R-squared value.

See Also

autotune_lasso

Examples

```
# Fit autotune lasso
set.seed(10)
n = 300
p = 500
s = 10
beta = c(rep(1, s), rep(0, p - s))
x = matrix(rnorm(n * p), ncol = p)
# Maunal sigma allocation
# y = x %*% beta + rnorm(n, sd = 1)
# Dynamic sigma allocation with snr specified
snr = 2
y = x %*% beta + rnorm(n, sd = sqrt(var(x%*%beta)/snr))
fit <- autotune_lasso(x, y)

# Basic diagnostic plot
plot(fit)

# Plot adjusted R-squared for first 15 predictors
plot(fit, max_preds = 15, cumulative = FALSE)
```

predict.autotune_lasso *Predict from a autotune_lasso fit*

Description

Predict from a autotune_lasso fit

Usage

```
## S3 method for class 'autotune_lasso'
predict(
  object,
  newx = NULL,
  type = c("response", "coefficients"),
  path = NULL,
  ...
)
```

Arguments

object	An object of class autotune_lasso.
newx	Numeric matrix of new x (rows = observations, columns = predictors) at which predictions are required. This argument is not used for type="coefficients").
type	Character; one of "response", "coefficients".
path	For type = "coefficients", NULL returns the final coefficients and TRUE returns the full coefficient path. This argument is ignored for type = "response".
...	Further arguments passed from the generic.

Value

If type = "response", returns the prediction at the x provided for prediction.

If type = "coefficients", returns the fitted coefficients.

Examples

```
set.seed(10)
n = 300
p = 500
s = 10
beta = c(rep(1, s), rep(0, p - s))
x = matrix(rnorm(n * p), ncol = p)
# Maunal sigma allocation
# y = x %*% beta + rnorm(n, sd = 1)
# Dynamic sigma allocation with snr specified
snr = 2
y = x %*% beta + rnorm(n, sd = sqrt(var(x%*%beta)/snr))
```

```
fit <- autotune_lasso(x, y)
predict(fit, newx = x[1:5, , drop = FALSE])
predict(fit, type = "coefficients")

predict(fit, type = "coefficients", path = TRUE)
```

---

sp500 *S&P 500 Stock Data*

Description

The sp500 data file contains a year's worth of close-of-day data for most of the Standard and Poor's 500 stocks. The data are in reverse chronological order, with the first row corresponding to December 31, 2008.

Usage

```
data("sp500")
```

Format

A list containing the following two matrices:

sp500.percent A 252 by 494 matrix of daily percentage changes. The first column is the DJIA index, the second is the S&P 500 index, and the remaining columns are labeled individual stocks.

sp500.2008 A 253 by 494 matrix of raw close-of-day data. The first column is the DJIA index, the second is the S&P 500 index, and the remaining columns are labeled individual stocks.

Source

Redistributed from version 1.0.1 of the GPL-2-licensed R package **scalreg**.

References

This database was used in the R package **plus**.

Examples

```
data("sp500")
names(sp500)
dim(sp500$sp500.percent)

attach(sp500)
head(sp500.percent[, 1:5])
detach(sp500)
```

Index