Package ‘glmgen’

January 8, 2015

Type Package

Title Fast generalized lasso solver

Version 0.0.2

Date 2014-09-03

Author Taylor Arnold, Veeranjaneyulu Sadhanala, Ryan Tibshirani

Maintainer Taylor Arnold <taylor.arnold@acm.org>

Description Efficient algorithms for generalized lasso problems.
Specialized implementations are provided to deal with special
problem classes, such as trend filtering.

License GPL-2 | GPL-3

Depends methods

R topics documented:

`glmgen-package` *Fast generalized lasso algorithms*

Description

Efficient algorithms for generalized lasso problems. Specialized implementations are provided to deal with special problem classes, such as trend filtering.

Details

Package:	glmgen
Type:	Package
Version:	0.0.1
Date:	2014-09-03
License:	GPL-2 | GPL-3
Depends:	methods

Author(s)

Taylor Arnold, Veeranjaneyulu Sadhanala, Ryan Tibshirani

Maintainer: Taylor Arnold <taylor.arnold@acm.org>

Examples

```
set.seed(0)
n = 100
x = runif(n, min=-2*pi, max=2*pi)
y = 1.5*sin(x) + sin(2*x) + rnorm(n, sd=0.2)
out = trendfilter(y, x, k=2)

xx = seq(min(x),max(x),length=500)
lambda = out@lambda[25]
yy = predict(out,x.new=xx,lambda=lambda)
plot(x,y)
lines(xx,yy,col=2)
```

coef-methods *Coefficients – S4 Methods*

Description

The "glmgen" package provides methods for calculating generalized lasso coefficients fit via the generic function coef.

Methods

signature(object = "ANY")

signature(object = "glmgen")

glmgen-class *Class* "glmgen"

Description

Parent class for all estimated generalized lasso fits in the **glmgen** package.

Objects from the Class

Objects will generally be constructed as a result of a call to estimation algorithms such as `trendfilter`.

Slots

- `y`: the input observed numeric values
- `lambda`: vector sequency of penalty lambdas, sorted in decreasing order
- `beta`: matrix of beta coefficents, with one column for each value of `lambda`
- `family`: character value giving the glm family used for estimation. Currently supports 'gaussian', 'binomial', and 'poisson'.
- `method`: method used to fit the generalized lasso
- `n`: an integer giving the length of `y`
- `p`: an integer giving the number of columns in the penalty matrix `D`
- `m`: an integer giving the number of rows in the penalty matrix `D`
- `obj`: a, possibly missing, matrix giving the objective values of the fit, with one column for each value of lambda
- `call`: the function `call` used for fitting the generalized lasso model

Methods

- **coef** `signature(object = "glmgen")`: estimates the coefficents of beta for a vector of (possibly new) `lambda`'s. New values can only be estimated within the range of the previously fitted `lambda` values.
- **show** `signature(object = "glmgen")`: prints a summary of the `glmgen` object.

Author(s)

Taylor Arnold, Veeranjaneyulu Sadhanala, Ryan Tibshirani

See Also

`trendfilter`,

predict-methods *Make function predictions given a trendfilter object*

Description

Once a trendfilter object has been fit, predictions can be made for a new set of x points and/or a new set of lambda values (provided that the new lambda values lie within the range of the original tuning parameter values).

Usage

```
## S4 method for signature 'trendfilter'
predict(object, type = c("link", "response"),
        lambda = NULL, x.new = NULL, zero_tol=1e-6)
```

Arguments

object	an R object inheriting from class trendfilter.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative response is on the scale of the response variable.
lambda	option vector of lambda values at which to predict. If NULL, the set of lambdas used for fitting are used for prediction.
x.new	optional vector of locations to use for prediction. If NULL, the set of locations used for fitting are used for prediction.
zero_tol	tolerance for the prediction algorithm; a small number slightly greater than zero greatly speeds up the computations with negligable degradation to the predictions.

Examples

```
set.seed(0)
n = 100
x = runif(n, min=-2*pi, max=2*pi)
y = 1.5*sin(x) + sin(2*x) + rnorm(n, sd=0.2)
out = trendfilter(y, x, k=2)

xx = seq(min(x),max(x),length=500)
lambda = out@lambda[25]
yy = predict(out,x.new=xx,lambda=lambda)
plot(x,y)
lines(xx,yy,col=2)

# Will throw a warning
predict(out,x.new=-10,lambda=lambda)
```

trendfilter *Fit a trend filtering model*

Description

Find the trend filtering solution of some degree k for an arbitrary set of penalty values lambda. Can handle link functions of Gaussian, binomial, and Poisson penalized loss functions.

Usage

```
trendfilter(y, x, weights, k = 2L, family = c("gaussian",
            "logistic", "poisson"), method = c("admm"), lambda,
            nlambda = 50L, lambda.min.ratio = 1e-05,
            thinning = NULL, verbose = FALSE,
            control = trendfilter.control.list())
```

Arguments

y vector of observed data points.

x optional vector of observed data locations. If missing, will be assumed to be the integers 1 through the length of y.

weights optional vector of sample weights. If missing, the weights will be assumed to be constant (unity) across all samples.

k the polynomial order of the trendfilter fit; a nonnegative integer (orders larger than 3 are not recommended). For instance, constant trend filtering (i.e., the fused lasso) uses k equal to 0, linear trend filtering uses k equal to 1, quadratic trend filtering uses k equal to 2, etc.

family the family for the link function in the trend filtering estimator. Can be either "gaussian", "logistic", or "poisson".

lambda a sequence of lambda values at which to produce a fit. Can be left blank (highly recommended for general use), at which point the algorithm will determine appropriate lambda values.

nlambda if lambda is missing, this determines the number of lambda values dynamically constructed by the algorithm.

lambda.min.ratio
if lambda is missing, this determines the ratio between the largest and smallest lambda values. The values are evenly spaced on a log scale, so this ratio should typically be set fairly small.

thinning logical. If true, then the data are preprocessed so that a smaller, better conditioned data set is used for fitting. When set to NULL, the default, function will auto detect whether thinning should be applied (i.e., cases in which the numerical fitting algorithm will struggle to converge).

method the method used to calculate the fit. Currently only 'admm' is supported.

verbose logical. Should the function print out intermediate results as it is running.

control an optional named list of control parameters to pass to the underlying algorithm; see Details for more information. Names not matching any valid parameters will be silently ignored.

Details

Further algorithmic parameters can be passed by using `trendfilter.control.list`.

Value

an object of class `trendfilter`

Author(s)

Taylor Arnold, Veeranjaneyulu Sadhanala, Ryan Tibshirani

References

Tibshirani, R. J. (2014), "Adaptive piecewise polynomial estimation via trend filtering", Annals of Statistics 42 (1): 285–323.

Ramdas, A. and Tibshirani R. J. (2014), "Fast and flexible ADMMM algorithms for trend filtering", arXiv: 1406.2082.

See Also

`trendfilter`, `trendfilter-method`

Examples

```
set.seed(0)
n = 100
x = runif(n, min=-2*pi, max=2*pi)
y = 1.5*sin(x) + sin(2*x) + rnorm(n, sd=0.2)
out = trendfilter(y, x, k=2)

xx = seq(min(x),max(x),length=500)
lambda = out@lambda[25]
yy = predict(out,x.new=xx,lambda=lambda)
plot(x,y)
lines(xx,yy,col=2)
```

`trendfilter-class` *Class* `"trendfilter"`

Description

Class for estimated trendfilter fits in the **glmgen** package.

Objects from the Class

Objects will generally be constructed as a result of a call to the function `trendfilter`.

Slots

`y`**:** the input observed numeric values

`x`**:** the locations of the observed values `y`

`w`**:** sample weights

`k`**:** integer, order of the trendfilter fit

`lambda`**:** vector sequency of penalty lambdas, sorted in decreasing order

`beta`**:** matrix of beta coefficents, with one column for each value of `lambda`

`family`**:** character value giving the GLM family used for estimatio; currently supports 'gaussian', 'binomial', and 'poisson'

`method`**:** method used to fit the generalized lasso

`n`**:** an integer giving the length of `y`

`m`**:** an integer giving the number of rows in the penalty matrix `D`

`p`**:** an integer giving the number of columns in the penalty matrix `D`

`obj`**:** a matrix giving the objective values of the fit, with one column for each value of lambda

`status`**:** a vector of integer codes indicating success or failure of the optimization routines, with one component for each value of lambda

`iter`**:** a vector of integers giving the number of iterations used, with one component for each value of lambda

`call`**:** the function `call` used for fitting the generalized lasso model

Extends

Class `"glmgen"`, directly.

Methods

predict `signature(object = "trendfilter")`: efficent calculation of predictions from the fit for a new set of `lambda` and/or a new set of `x` values. The parameter `lambda` within the range of the original tuning parameter values.

summary `signature(object = "trendfilter")`: produces a summary of the trendfilter fit.

Author(s)

Taylor Arnold, Veeranjaneyulu Sadhanala, Ryan Tibshirani

References

Tibshirani, R. J. (2014), "Adaptive piecewise polynomial estimation via trend filtering", Annals of Statistics 42 (1): 285–323.

Ramdas, A. and Tibshirani R. J. (2014), "Fast and flexible ADMMM algorithms for trend filtering", arXiv: 1406.2082.

See Also

`glmgen`,

Examples

```
set.seed(0)
n = 100
x = runif(n, min=-2*pi, max=2*pi)
y = 1.5*sin(x) + sin(2*x) + rnorm(n, sd=0.2)
out = trendfilter(y, x, k=2)

xx = seq(min(x),max(x),length=500)
lambda = out@lambda[25]
yy = predict(out,x.new=xx,lambda=lambda)
plot(x,y)
lines(xx,yy,col=2)
```

trendfilter.control.list

Control list for tuning trend filtering algorithm

Description

Constructs the control parameters for the trend filtering algorithm. Allows the user to customize as many or as little as desired.

Usage

```
trendfilter.control.list(rho=1, obj_tol=1e-6, max_iter=200L,
                         max_iter_newton=50L, x_cond=1e11,
                         alpha_ls=0.5, gamma_ls=0.8, max_iter_ls=20L)
```

Arguments

rho — this is a scaling factor for the augmented Lagrangian parameter in the ADMM algorithm. To solve a given trend filtering problem with locations x at a tuning parameter value lambda, the augmented Lagrangian parameter is set to be rho * lambda * ((max(x)-min(x))/n)^k.

obj_tol — the tolerance used in the stopping criterion; when the relative change in objective values is less than this value, the algorithm terminates.

max_iter — number of ADMM iterations used; ignored for k=0.

max_iter_newton — for non-Gaussian GLM losses, the number of outer iterations used in Newton's method.

x_cond — condition number to control the degree of thinning, when applicable. Lower numbers enforce more thinning.

alpha_ls — tuning parameter for the line search used in the proximal Newton procedure for non-Gaussian GLM losses.

gamma_ls — tuning parameter for the line search used in the proximal Newton for non-Gaussian GLM losses.

max_iter_ls — tuning parameter for the number of line search iterations in the proximal Newton procedure for non-Gaussian GLM losses.

Value

an object of class `trendfilter`

Author(s)

Taylor Arnold

See Also

`trendfilter`

Examples

```
set.seed(0)
n = 100
x = runif(n, min=-2*pi, max=2*pi)
y = 1.5*sin(x) + sin(2*x) + rnorm(n, sd=0.2)
out = trendfilter(y, x, k=2, control=trendfilter.control.list(rho=3))
```

Index