

# Package ‘grplasso’

April 17, 2009

**Type** Package

**Title** Fitting user specified models with Group Lasso penalty

**Version** 0.4-2

**Date** 2009-04-01

**Author** Lukas Meier

**Maintainer** Lukas Meier <lukas.meier@gmx.net>

**Description** Fits user specified (GLM-) models with Group Lasso penalty

**Depends** methods

**License** GPL

**Repository** CRAN

**Date/Publication** 2009-04-01 11:43:14

## R topics documented:

grplasso-package . . . . .	2
grpl.control . . . . .	3
grpl.control-class . . . . .	4
grpl.model . . . . .	5
grpl.model-class . . . . .	6
grplasso . . . . .	7
lambdamax . . . . .	9
plot.grplasso . . . . .	11
predict.grplasso . . . . .	12
splice . . . . .	13

<b>Index</b>	<b>14</b>
--------------	-----------

---

 grplasso-package    *grplasso*


---

## Description

Fits generalized linear models or user specified models with group lasso penalty.

## Details

```

Package:  grplasso
Type:    Package
Version:  0.1-2
Date:    2006-08-11
Depends:  methods
License:  GPL
Built:    R 2.3.1; ; 2006-06-01 09:38:41; unix
  
```

Please note that this is an *\*early test release\**.

The best entry point for the package are the examples in the help file of the function `grplasso`.

### Index:

```

grpl.control          Options for the Group Lasso Algorithm
grpl.control-class   Class "grpl.control": Options for the Group
                    Lasso Algorithm
grpl.model           Group Lasso models
grpl.model-class     Class "grpl.model": Group Lasso Models
grplasso             Function to fit a solution of a group lasso
                    problem
grplasso-package     grplasso
lambdamax            Function to find the maximal value of the
                    penalty parameter lambda
plot.grplasso        Plots the solution path of a grplasso object
predict.grplasso     Predict method for grplasso objects
splice              Dataset of human donor splice sites
  
```

## Author(s)

Lukas Meier

Maintainer: Lukas Meier <meier@stat.math.ethz.ch>

**Description**

Definition of options such as bounds on the Hessian, convergence criteria and output management for the Group Lasso algorithm.

**Usage**

```
grpl.control(save.x = FALSE, save.y = TRUE,
             update.hess = c("lambda", "always"), update.every = 3,
             inner.loops = 10, line.search = TRUE, max.iter = 500,
             tol = 5 * 10^-8, lower = 10^-2, upper = Inf, beta = 0.5,
             sigma = 0.1, trace = 1)
```

**Arguments**

save.x	a logical indicating whether the design matrix should be saved.
save.y	a logical indicating whether the response should be saved.
update.hess	should the hessian be updated in each iteration ("always")? update.hess = "lambda" will update the Hessian once for each component of the penalty parameter "lambda" based on the parameter estimates corresponding to the previous value of the penalty parameter.
update.every	Only used if update.hess = "lambda". E.g. set to 3 if you want to update the Hessian only every third grid point.
inner.loops	How many loops should be done (at maximum) when solving only the active set (without considering the remaining predictors). Useful if the number of predictors is large. Set to 0 if no inner loops should be performed.
line.search	Should line searches be performed?
max.iter	Maximal number of loops through all groups
tol	convergence tolerance; the smaller the more precise, see details below.
lower	lower bound for the diagonal approximation of the corresponding block submatrix of the Hessian of the negative log-likelihood function.
upper	upper bound for the diagonal approximation of the corresponding block submatrix of the Hessian of the negative log-likelihood function.
beta	scaling factor $\beta < 1$ of the Armijo line search.
sigma	$0 < \sigma < 1$ used in the Armijo line search.
trace	integer. 0 omits any output, 1 prints the current lambda value, 2 prints the improvement in the objective function after each sweep through all the parameter groups and additional information.

**Details**

For the convergence criteria see chapter 8.2.3.2 of Gill et al. (1981).

**Value**

An object of class `grpl.control`.

**References**

Philip E. Gill, Walter Murray and Margaret H. Wright (1981) *Practical Optimization*, Academic Press.

Dimitri P. Bertsekas (2003) *Nonlinear Programming*, Athena Scientific.

---

`grpl.control-class` Class "grpl.control": Options for the Group Lasso Algorithm

---

**Description**

Objects of class "grpl.control" define options such as bounds on the Hessian, convergence criteria and output management for the Group Lasso algorithm.

**Details**

For the convergence criteria see chapter 8.2.3.2 of Gill et al. (1981).

**Objects from the Class**

Objects can be created by calls of the form `grpl.control(...)`

**Slots**

**save.x** a logical indicating whether the design matrix should be saved.

**save.y** a logical indicating whether the response should be saved.

**update.hess** should the hessian be updated in each iteration ("always")? `update.hess = "lambda"` will update the Hessian once for each component of the penalty parameter "lambda" based on the parameter estimates corresponding to the previous value of the penalty parameter.

**update.every** Only used if `update.hess = "lambda"`. E.g. set to 3 if you want to update the Hessian only every third grid point.

**inner.loops** How many loops should be done (at maximum) when solving only the active set (without considering the remaining predictors). Useful if the number of predictors is large. Set to 0 if no inner loops should be performed.

**line.search** Should line searches be performed?

**max.iter** Maximal number of loops through all groups

**tol** convergence tolerance; the smaller the more precise.

**lower** lower bound for the diagonal approximation of the corresponding block submatrix of the Hessian of the negative log-likelihood function.

**upper** upper bound for the diagonal approximation of the corresponding block submatrix of the Hessian of the negative log-likelihood function.

**beta** scaling factor  $\beta < 1$  of the Armijo line search.

**sigma**  $0 < \sigma < 1$  used in the Armijo line search.

**trace** integer. 1 prints the current lambda value, 2 prints the improvement in the objective function after each sweep through all the parameter groups and additional information.

## References

Philip E. Gill, Walter Murray and Margaret H. Wright (1981) *Practical Optimization*, Academic Press.

Dimitri P. Bertsekas (2003) *Nonlinear Programming*, Athena Scientific.

---

 grpl.model

 Group Lasso models
 

---

## Description

Generates models to be used for the Group Lasso algorithm.

## Usage

```
grpl.model(invlink, link, nloglik, ngradient, nhessian, check,
           name = "user-specified", comment = "user-specified")
LogReg()
LinReg()
PoissReg()
```

## Arguments

invlink	a function with arguments <code>eta</code> implementing the inverse link function.
link	a function with arguments <code>mu</code> implementing the link function.
nloglik	a function with arguments <code>y</code> , <code>mu</code> and <code>weights</code> implementing the <i>negative</i> log-likelihood function.
ngradient	a function with arguments <code>x</code> , <code>y</code> , <code>mu</code> and <code>weights</code> implementing the <i>negative</i> gradient of the log-likelihood function.
nhessian	a function with arguments <code>x</code> , <code>mu</code> and <code>weights</code> implementing the <i>negative</i> hessian of the log-likelihood function.
check	a function with argument <code>y</code> to check whether the response has the correct format.
name	a character name
comment	a character comment

**Value**

An object of class `grpl.model`.

**Examples**

```
LogReg ()
```

---

```
grpl.model-class   Class "grpl.model": Group Lasso Models
```

---

**Description**

Objects of class "grpl.model" define link function, negative log-likelihood and corresponding gradient and Hessian for the model to be used in a Group Lasso problem.

**Objects from the Class**

Objects can be created by calls of the form `grpl.model (...)`

**Slots**

**invlink** a function with arguments `eta` implementing the inverse link function.

**link** a function with arguments `mu` implementing the link function.

**nloglik** a function with arguments `y`, `mu` and `weights` implementing the *negative* log-likelihood function.

**ngradient** a function with arguments `x`, `y`, `mu` and `weights` implementing the *negative* gradient of the log-likelihood function.

**nhessian** a function with arguments `x`, `mu` and `weights` implementing the *negative* hessian of the log-likelihood function.

**check** a function with argument `y` to check whether the response has the correct format.

**name** a character name

**comment** a character comment

**Methods**

**show** object

**Examples**

```
LogReg ()
```

---

grplasso

*Function to fit a solution of a group lasso problem*


---

## Description

Fits the solution of a group lasso problem for a model of type `grpl.model`.

## Usage

```
grplasso(x, ...)
```

```
## S3 method for class 'formula':
grplasso(formula, nonpen = ~ 1, data, weights = rep(1, length(y)),
         subset, na.action, lambda, coef.init, penscale = sqrt,
         model = LogReg(), center = TRUE, standardize = TRUE,
         control = grpl.control(), contrasts = NULL, ...)
```

```
## Default S3 method:
grplasso(x, y, index, weights = rep(1, length(y)), offset = rep(0,
         length(y)), lambda, coef.init = rep(0, ncol(x)),
         penscale = sqrt, model = LogReg(), center = TRUE,
         standardize = TRUE, control = grpl.control(), ...)
```

## Arguments

<code>x</code>	design matrix (including intercept)
<code>y</code>	response vector
<code>formula</code>	formula of the penalized variables. The response has to be on the left hand side of <code>~</code> .
<code>nonpen</code>	formula of the nonpenalized variables. This will be added to the <code>formula</code> argument above and doesn't need to have the response on the left hand side.
<code>data</code>	<code>data.frame</code> containing the variables in the model.
<code>index</code>	vector which defines the grouping of the variables. Components sharing the same number build a group. Non-penalized coefficients are marked with <code>NA</code> .
<code>weights</code>	vector of observation weights.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>na.action</code>	a function which indicates what should happen when the data contain 'NA's.
<code>offset</code>	vector of offset values; needs to have the same length as the response vector.
<code>lambda</code>	vector of penalty parameters. Optimization starts with the first component. See details below.
<code>coef.init</code>	initial vector of parameter estimates corresponding to the first component in the vector <code>lambda</code> .

<code>penscale</code>	rescaling function to adjust the value of the penalty parameter to the degrees of freedom of the parameter group. See the reference below.
<code>model</code>	an object of class <code>grpl.model</code> implementing the negative log-likelihood, gradient, hessian etc. See the documentation of <code>grpl.model</code> for more details.
<code>center</code>	logical. If true, the columns of the design matrix will be centered (except a possible intercept column).
<code>standardize</code>	logical. If true, the design matrix will be blockwise orthonormalized such that for each block $X^T X = n1$ (*after* possible centering).
<code>control</code>	options for the fitting algorithm, see <code>grpl.control</code> .
<code>contrasts</code>	an optional list. See the 'contrasts.arg' of 'model.matrix.default'.
<code>...</code>	additional arguments to be passed to the functions defined in <code>model</code> .

### Details

When using `grplasso.formula`, the grouping of the variables is derived from the type of the variables: The dummy variables of a factor will be automatically treated as a group.

The optimization process starts using the first component of `lambda` as penalty parameter  $\lambda$  and with starting values defined in `coef.init` for the parameter vector. Once fitted, the next component of `lambda` is considered as penalty parameter with starting values defined as the (fitted) coefficient vector based on the previous component of `lambda`.

### Value

A `grplasso` object is returned, for which `coef`, `print`, `plot` and `predict` methods exist.

<code>coefficients</code>	coefficients with respect to the <i>original</i> input variables (even if <code>standardize = TRUE</code> is used for fitting).
<code>lambda</code>	vector of lambda values where coefficients were calculated.
<code>index</code>	grouping index vector.

### Author(s)

Lukas Meier, (meier@stat.math.ethz.ch)

### References

Lukas Meier, Sara van de Geer and Peter Bühlmann (2008), *The Group Lasso for Logistic Regression*, Journal of the Royal Statistical Society, 70 (1), 53 - 71, see also <http://stat.ethz.ch/~meier/logistic-grouplasso.php>

### Examples

```
## Use the Logistic Group Lasso on the splice data set
data(splice)

## Define a list with the contrasts of the factors
contr <- rep(list("contr.sum"), ncol(splice) - 1)
names(contr) <- names(splice)[-1]
```

```

## Fit a logistic model
fit.splice <- grplasso(y ~ ., data = splice, model = LogReg(), lambda = 20,
                      contrasts = contr, center = TRUE, standardize = TRUE)

## Perform the Logistic Group Lasso on a random dataset
set.seed(79)

n <- 50 ## observations
p <- 4  ## variables

## First variable (intercept) not penalized, two groups having 2 degrees
## of freedom each

index <- c(NA, 2, 2, 3, 3)

## Create a random design matrix, including the intercept (first column)
x <- cbind(1, matrix(rnorm(p * n), nrow = n))
colnames(x) <- c("Intercept", paste("X", 1:4, sep = ""))

par <- c(0, 2.1, -1.8, 0, 0)
prob <- 1 / (1 + exp(-x %*% par))
mean(pmin(prob, 1 - prob)) ## Bayes risk
y <- rbinom(n, size = 1, prob = prob) ## binary response vector

## Use a multiplicative grid for the penalty parameter lambda, starting
## at the maximal lambda value
lambda <- lambdamax(x, y = y, index = index, penscale = sqrt,
                    model = LogReg()) * 0.5^(0:5)

## Fit the solution path on the lambda grid
fit <- grplasso(x, y = y, index = index, lambda = lambda, model = LogReg(),
               penscale = sqrt,
               control = grpl.control(update.hess = "lambda", trace = 0))

## Plot coefficient paths
plot(fit)

```

---

lambdamax

---

*Function to find the maximal value of the penalty parameter lambda*


---

### Description

Determines the value of the penalty parameter lambda when the first penalized parameter group enters the model.

### Usage

```
lambdamax(x, ...)
```

```
## S3 method for class 'formula':
lambdamax(formula, nonpen = ~1, data, weights, subset,
           na.action, coef.init, penscale = sqrt, model = LogReg(),
           standardize = TRUE, contrasts = NULL, nlminb.opt = list(), ...)

## Default S3 method:
lambdamax(x, y, index, weights = rep(1, length(y)),
           offset = rep(0, length(y)), coef.init = rep(0, ncol(x)),
           penscale = sqrt, model = LogReg(), center = TRUE,
           standardize = TRUE, nlminb.opt = list(), ...)
```

## Arguments

<code>x</code>	design matrix (including intercept)
<code>y</code>	response vector
<code>formula</code>	formula of the penalized variables. The response has to be on the left hand side of '~'.
<code>nonpen</code>	formula of the nonpenalized variables. This will be added to the <code>formula</code> argument above and doesn't need to have the response on the left hand side.
<code>data</code>	<code>data.frame</code> containing the variables in the model.
<code>index</code>	vector which defines the grouping of the variables. Components sharing the same number build a group. Non-penalized coefficients are marked with NA.
<code>weights</code>	vector of observation weights.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>na.action</code>	a function which indicates what should happen when the data contain 'NA's.
<code>offset</code>	vector of offset values.
<code>coef.init</code>	initial parameter vector. Penalized groups are discarded.
<code>penscale</code>	rescaling function to adjust the value of the penalty parameter to the degrees of freedom of the parameter group. See the reference below.
<code>model</code>	an object of class <code>grpl.model</code> implementing the negative log-likelihood, gradient, hessian etc. See <code>grpl.model</code> for more details.
<code>center</code>	logical. If true, the columns of the design matrix will be centered (except a possible intercept column).
<code>standardize</code>	logical. If true, the design matrix will be blockwise orthonormalized, such that for each block $X^T X = nI$ (*after* possible centering).
<code>contrasts</code>	an (optional) list with the contrasts for the factors in the model.
<code>nlminb.opt</code>	arguments to be supplied to <code>nlminb</code> .
<code>...</code>	additional arguments to be passed to the functions defined in <code>model</code> .

## Details

Uses `nlminb` to optimize the non-penalized parameters.

**Value**

An object of type numeric is returned.

**Examples**

```
data(splICE)
lambdamax(y ~ ., data = splICE, model = LogReg(), center = TRUE,
          standardize = TRUE)
```

---

plot.grplasso      *Plots the solution path of a grplasso object*

---

**Description**

Plots the solution path of a grplasso object.

**Usage**

```
## S3 method for class 'grplasso':
plot(x, type = "coefficients", col = NULL, ...)
```

**Arguments**

x	a grplasso object
type	type = "coefficients" plots coefficients with respect to the input variables, even if standardize = TRUE is used in grplasso.
col	a vector indicating the color of the different group paths. The length should equal the number of groups. The same ordering as in the vector index is used with the exception that the unpenalized coefficients are grouped at the beginning of the vector.
...	other parameters to be passed to the plotting functions.

**Examples**

```
data(splICE)

contr <- list(Pos.1 = "contr.sum", Pos.2 = "contr.sum")
lambda <- lambdamax(y ~ Pos.1 * Pos.2, data = splICE, model = LogReg(),
                  contrasts = contr, standardize = TRUE) * 0.8^(0:8)

fit <- grplasso(y ~ Pos.1 * Pos.2, data = splICE, model = LogReg(),
              lambda = lambda, contrasts = contr, standardize = TRUE,
              control = grpl.control(trace = 0, inner.loops = 0,
                                    update.every = 1, update.hess = "lambda"))
plot(fit, log = "x")
```

---

predict.grplasso *Predict method for grplasso objects*

---

### Description

Obtains predictions from a grplasso object.

### Usage

```
## S3 method for class 'grplasso':
predict(object, newdata, type = c("link", "response"),
        na.action = na.pass, ...)
```

### Arguments

object	a grplasso object
newdata	data.frame or design matrix of new observations
type	the type of prediction. type = "link" is on the scale of linear predictors, whereas type = "response" is on the scale of the response variable, i.e. type = "response" applies the inverse link function to the linear predictors.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	other options to be passed to the predict function.

### Value

A matrix whose *columns* correspond to the different values of the penalty parameter `lambda` of the grplasso object.

### Note

If `newdata` is given, offsets specified by `offset` in the fit by `grplasso.default` will not be included in predictions, whereas those specified by an `offset` term in the formula will be considered.

### See Also

[grplasso](#)

### Examples

```
data(splice)

contr <- rep(list("contr.sum"), ncol(splice) - 1)
names(contr) <- names(splice)[-1]

fit <- grplasso(y ~ ., data = splice, model = LogReg(), lambda = 10,
```

```
contrasts = contr, standardize = TRUE)

pred <- predict(fit)
pred.resp <- predict(fit, type = "response")

## The following points should lie on the sigmoid curve
plot(pred, pred.resp)
```

---

splice

*Dataset of human donor splice sites*

---

### Description

Dataset of 400 human donor splice sites with a sequence length of 7 base pairs.

### Usage

```
data(splice)
```

### Format

**y** binary response. True (1) or false (0) splice site.

**Pos.x** DNA letter (A, C, G, T) at position x, where x ranges from 1 to 7.

### Details

The dataset is a random subset of the MEMset Donor dataset used in Gene et al. (2004).

### References

Gene, Y. and Burge, C. (2004) *Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals*, Journal of Computational Biology, 11, 475 - 494.

### Examples

```
data(splice)
```

# Index

- \*Topic **classes**
  - grpl.control-class, 3
  - grpl.model-class, 5
- \*Topic **datasets**
  - splice, 12
- \*Topic **hplot**
  - plot.grplasso, 10
- \*Topic **methods**
  - predict.grplasso, 11
- \*Topic **misc**
  - grpl.control, 2
  - grpl.model, 4
  - lambdamax, 9
- \*Topic **models**
  - grplasso, 6
- \*Topic **package**
  - grplasso-package, 1
- \*Topic **regression**
  - grplasso, 6

grpl.control, 2, 7

grpl.control-class, 3

grpl.model, 4, 7, 10

grpl.model-class, 5

grplasso, 2, 6, 12

grplasso-package, 1

lambdamax, 9

LinReg (*grpl.model*), 4

LogReg (*grpl.model*), 4

nlminb, 10

plot.grplasso, 10

PoissReg (*grpl.model*), 4

predict.grplasso, 11

show, grpl.model-method  
(*grpl.model-class*), 5

splice, 12