

Package ‘logilasso’

April 17, 2009

Type Package

Title Analysis of sparse contingency tables with penalization approaches

Version 0.1.0

Date 2006-11-13

Author Corinne Dahinden

Maintainer Corinne Dahinden <dahinden@stat.math.ethz.ch>

Description Analysis of sparse contingency tables with penalization approaches

Depends methods, Rgraphviz

Suggests gRbase

License GPL

Repository CRAN

Date/Publication 2007-11-13 19:31:54

R topics documented:

logilasso-package	2
graphmod	2
levelcv	4
logilasso	6
plot.logilasso	8
predict.logilasso	9
traceplot.logilasso	11

Index	13
--------------	-----------

logilasso-package *logilasso*

Description

Fits loglinear models to data arising from sparse contingency tables by penalizing the multinomial likelihood.

Details

Package: logilasso
 Type: Package
 Version: 1.0
 Date: 2006-11-13
 Depends: methods, dynamicGraph
 License: GPL
 Built: R 2.4.0; ; 2006-11-13 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 [logilasso](#).

Index:

logilasso	Fits a loglinear model or/and performs cross-validation
levelcv	Performs cross-validation for the specified number of interactions
traceplot	Plots the solution path from λ_{\max} to λ_{\min} for all components of the solution vector β
graphmod	Plots a graphical model
plot.logilasso	Plot method for a logilasso object
predict.logilasso	Predict method for a logilasso object

Author(s)

Corinne Dahinden

Maintainer: Corinne Dahinden <dahinden@stat.math.ethz.ch>

graphmod

Function to plot a graphical model.

Description

Function to plot a graphical model for objects of class `predlogilasso`, `cvlogilasso` and `levellogilasso`. For details about graphical models, see reference.

Usage

```
graphmod(obj, grenze = 0, nnames = NULL, ...)

## S3 method for class 'predlogilasso':
graphmod(obj, grenze = 0, nnames = NULL, ... )

## S3 method for class 'cvlogilasso':
graphmod(obj, grenze=0, nnames = NULL, lambda = NULL, ... )
## S3 method for class 'levellogilasso':
graphmod(obj, grenze=0, nnames = NULL, lambda = NULL, to.which.int = NULL, ... )
```

Arguments

obj	Object of class <code>predlogilasso</code> , <code>cvlogilasso</code> or <code>levellogilasso</code> .
grenze	Components of beta which are smaller than <code>grenze</code> times the biggest component of beta are truncated to zero.
nnames	Character vector. Names of the nodes in the graphical model.
lambda	For object of class <code>cvlogilasso</code> or <code>levellogilasso</code> : Should not be specified except if the graphical model for a specific value of <code>lambda</code> should be drawn. Otherwise this value is assessed by cross-validation.
to.which.int	For objects of class <code>levellogilasso</code> : Should not be specified except if the graphical model for a specific interaction number is desired. Otherwise assessed by cross-validation.
...	Additional arguments to be passed to the plotting function.

References

Steffen L. Lauritzen, *Graphical Models*, Oxford University Press, 1996

Examples

```
library(gRbase)
data(reinis)

fit <- logilasso(reinis, lambdainit=1)
pred <- predict(fit, lambda=0.5)
# Graphical model of an object of class predlogilasso
graphmod(pred, nnames=as.character(c(1:6)))

fit <- logilasso(reinis, lambdainit=1, cvfold=3)
# Graphical model of an object of class cvlogilasso
graphmod(fit, nnames=c("one", "two", "three", "four", "five", "six"))
```

levelcv

*Cross-Validation for different interaction levels***Description**

Performs cross-validation for interactions involving $k, k+1, \dots, k+u$ factors which can be specified by the arguments `from.which.int=k` and `to.which.int=k+u`. For each interaction order $k+j$ a `cvfold` cross-validation is performed. Returns an object of class `levellogilasso` which is a subclass of `cvlogilasso`.

Usage

```
levelcv(Y , combX = NULL , from.which.int = 1, to.which.int=NULL,
epsilon = 0.1, lambdainit =1, lambdamin = 0.1, trace = 1,
method = "group11", cvfold=10, Newton = TRUE, stopkrit = 1e-8)
```

Arguments

The function can be initialized in 2 different ways: 1. `Y` as a contingency table. 2. `Y` together with a combination matrix `combX` (see example).

<code>Y</code>	The vector containing the counts for each cell in the contingency table. Either given as table or as vector together with a matrix <code>combX</code> , where each component of the vector corresponds to a combination of factors in the corresponding row of <code>combX</code> .
<code>combX</code>	Matrix of dimension <code>length(Y) x number of factors</code> . For each component of <code>Y</code> the corresponding combinations of the level is given as a row entry. See example.
<code>from.which.int</code>	Numeric. The smallest number of factors whose interaction is considered. For example if it is =1, the algorithm starts considering all main effects models.
<code>to.which.int</code>	Up to which interaction should the solution be calculated. <code>to.which.int=n</code> indicate that interaction involving <code>n</code> factors are considered.
<code>epsilon</code>	The step length for lambda. In each step, lambda diminished by epsilon.
<code>lambdainit</code>	The upper bound for lambda, where the solution path for beta starts.
<code>lambdamin</code>	The lower bound for lambda, where the solution path ends.
<code>trace</code>	Defines what is printed out during the calculation. 0 = nothing, 1 = Current cv-fold 2 = additionally points for each lambda 3 = Every 10th step writes the active set 4 = Additionally the new active set are written out, whenever a component enters the active set
<code>method</code>	Is either "group11", "l1" or "l2", depending on the desired penalization of the coefficients.
<code>cvfold</code>	If <code>cvfold</code> is larger than 1, cross-validation is performed.
<code>Newton</code>	Logical. If <code>Newton=TRUE</code> , Newton steps are performed, otherwise the function <code>optim</code> is used.
<code>stopkrit</code>	Convergence tolerance; the smaller the more precise.

Details

A `levellogilasso` object is returned. This is a list of objects of class `cvlogilasso` for each interaction from `from.which.int` to `to.which.int`. For objects of that class, the methods `plot`, `predict`, `traceplot` and `graphmod` are available.

Value

The elements of the list of class `cvlogilasso` each consists of the following entries:

<code>loss</code>	A loss matrix of dimension <code>cvfold</code> x number of assessed lambdas in the solution path. For each part of the data left out, the loss for all lambdas of the solution path is calculated. The <code>loss</code> is <code>NULL</code> in case of <code>cvfold=1</code>
<code>path</code>	A matrix. In the second row the newly active or newly inactive components of beta are listed to the corresponding lambda in the first row of the matrix. Is <code>NULL</code> for the class <code>cvlogilasso</code>
<code>betapath</code>	A matrix of dimension <code>length(beta)</code> x lambda in the solution path. The columns consist of the betas for the different lambdas in <code>lambdapath</code> . Is <code>NULL</code> for the class <code>cvlogilasso</code> .
<code>lambdapath</code>	A vector of all lambdas for which the solution was calculated.
<code>X</code>	The design matrix used to fit the log-linear model.
<code>nrfac</code>	Number of factors.

References

Corinne Dahinden, Giovanni Parmigiani, Mark Emerik and Peter Buehlmann available at <http://stat.ethz.ch/~dahinden/Paper/BMC.pdf>

Examples

```
library(gRbase)
data(reinis)
levellogi <- levelcv(reinis, lambdainit=1, lambdamin=0.1, from.which.int=1, to.which.int=3, cvfold=3)

## The methods predict, plot, traceplot and graphmod are defined for the resulting
## object of class cvlogilasso
predlevel <- predict(levellogi)
plot(predlevel)
traceplot(predlevel)
graphmod(predlevel)

### Different initialization: Y and combX
### 5 factors: All have 2 levels
### nlev would be c(2,2,2,2,2)
Y <- c(4,1,3,2,9)
combX <- rbind(c(1,0,1,1,0), c(1,0,0,1,1), c(0,1,0,0,1), c(0,0,1,0,0), c(1,1,0,0,1))
### 4 observations wit level 1 of factor 1, level 0 of factor two, level 1 of factor
### 3 and so on.
levellogi2 <- levelcv(Y, combX=combX, from.which.int=1, to.which.int=3, cvfold=3)
```

logilasso

*Fits a log-linear model and/or performs cross-validation.***Description**

Fits a log-linear interaction model ($\log(p)=X*\beta$) penalizing the log-likelihood function assuming a multinomial sampling scheme. Penalization can be chosen as either an l1, l2 or a group l1 penalty. In addition cross-validation is performed if `cvfold` is chosen larger than 1. The returned objects are of class `logilasso` if no cross-validation has been performed or of class `cvlogilasso`, which is a subclass of `logilasso`, if cross-validation has been performed.

Usage

```
logilasso(Y, combX = NULL, to.which.int=NULL, epsilon = 0.1,
          lambdainit = 1, lambdamin = 0.1, trace = 1, method = "groupl1",
          cvfold=1, Newton = TRUE, stopkrit = 1e-8, sse = NULL, X = NULL)
```

Arguments

There are 2 different ways of initializing the function: 1. `Y` as a contingency table. 2. `Y` together with a combination matrix `combX` (see example).

<code>Y</code>	The vector containing the counts for each cell in the contingency table. Either given as table or as vector together with a matrix <code>combX</code> , where each component of the vector corresponds to a combination of factors in the corresponding row of <code>combX</code> .
<code>combX</code>	Matrix of dimension <code>length(Y) x number of factors</code> . For each component of <code>Y</code> the corresponding combinations of the level is given as a row entry. See example.
<code>to.which.int</code>	Up to which interaction should the solution be calculated. <code>to.which.int=n</code> indicate that interaction involving <code>n</code> factors are considered.
<code>epsilon</code>	The step length of <code>lambda</code> . In each step the penalization parameter <code>lambda</code> is decreased by <code>epsilon</code> .
<code>lambdainit</code>	The upper bound for <code>lambda</code> , where the solution path for <code>beta</code> starts.
<code>lambdamin</code>	The lower bound for <code>lambda</code> , where the solution path ends.
<code>trace</code>	Defines what is printed out during the calculation. 0 = nothing, 1 = Current <code>cvfold</code> 2 = additionally points for each <code>lambda</code> 3 = Every 10th step writes the active set 4 = Additionally the new active set are written out, whenever a component enters the active set
<code>method</code>	Is either "groupl1", "l1" or "l2", depending on the penalization of the coefficients.
<code>cvfold</code>	If <code>cvfold</code> is larger than 1, cross-validation is performed.
<code>Newton</code>	Logical. If <code>Newton=TRUE</code> , Newton steps are performed, otherwise the function <code>optim</code> is used.

stopkrit	Convergence tolerance; the smaller the more precise, see details below.
sse	.Random.seed vector.
X	The design matrix X which is used for fitting a log-linear model. Should not be specified by the user.

Details

For the convergence criteria see chapter 8.2.3.2 of Gill et al. (1981). *Practical Optimization*, Academic Press.

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

For the resulting objects of class `logilasso` and `cvlogilasso` the methods `plot`, `predict` and `traceplot` are available. If Cross-Validation was performed (`cvfold>1`), then in addition the method `graphmod` is applicable, which plots a graphical model.

Value

A `cvlogilasso` object is returned in case `cvfold` is larger than 1. A `logilasso` object is returned in case `cvfold` is equal to 1. The class `cvlogilasso` is a subclass of `logilasso`.

loss	A loss matrix of dimension <code>cvfold</code> x number of assessed lambdas in the solution path. For each part of the data left out, the loss for all lambdas of the solution path is calculated. The <code>loss</code> is NULL in case of <code>cvfold=1</code>
path	A matrix. In the second row the newly active or newly inactive components of beta are listed to the corresponding lambda in the first row of the matrix. Is NULL for the class <code>cvlogilasso</code>
betapath	A matrix of dimension <code>length(beta)</code> x number of assessed lambda in the solution path. The columns consist of the betas for the different lambdas in <code>lambdapath</code> . Is NULL for the class <code>cvlogilasso</code> .
lambdapath	A vector of all lambdas for which the solution was calculated.
X	The design matrix used to fit the log-linear model.
nrfac	Number of factors.

Author(s)

Corinne Dahinden, <dahinden@stat.math.ethz.ch>

References

Corinne Dahinden, Giovanni Parmigiani, Mark Emerik and Peter Buehlmann available at <http://stat.ethz.ch/~dahinden/Paper/BMC.pdf>

Examples

```
## Use logilasso on the reinis dataset provided in the
## package gRbase
library(gRbase)
data(reinis)
```

```

## Fit a log-linear model for lambdas between 1 and 0.1
## No cross-validation is performed
fit <- logilasso(reinis,lambdainit=1,lambdaamin=0.1)

### Different initialization: Y and combX
### 5 factors: All have 2 levels
Y <- c(4,1,3,2,9)
combX <- rbind(c(1,0,1,1,0),c(1,0,0,1,1),c(0,1,0,0,1),c(0,0,1,0,0),c(1,1,0,0,1))
### 4 observations wit level 1 of factor 1, level 0 of factor two, level 1 of factor
### 3 and so on.
### The rows of combX correspond to a the levels of the five factors
### Must be numeric with 0/1/2/... and so on.
fit2 <- logilasso(Y,combX)

## Trajectories from lambdainit to lambda optimal
plot(fit2)
traceplot(fit2)

## Predict functions
pred <- predict(fit,lambda=0.3)

## Perform 3-fold cross-validation
fitcv <- logilasso(reinis,lambdainit=1,lambdaamin=0.1,cvfold=3)

## Plots a graphical model with the lambda calculated by cross-validation.
plot(fitcv)
graphmod(fitcv)

predcv <- predict(fitcv)

```

plot.logilasso *Plot function*

Description

Plot function for objects of class logilasso, predlogilasso and predlogispez.

Usage

```

## S3 method for class 'logilasso':
plot(x, grenze = 0, ...)
## S3 method for class 'predlogilasso':
plot(x, grenze = 0, ...)
## S3 method for class 'predlogispez':
plot(x, ...)

```

Arguments

x	An object of class logilasso, predlogilasso or predlogispez.
grenze	Up to which fraction of the maximal absolute component of beta, the individual components are treated as zero. If 1, then all beta components are treated as zero, if 0, then all components which do not equal zero are treated as nonzeros.
...	Additional arguments to the plot function.

Details

For objects of class predlogispez the trajectories of the components of the beta vectors are plotted against the lambdas. For objects of class predlogilasso the corresponding undirected graphical model (CIG) is plotted.

For objects of the class logilasso, first the predict() function is applied and the resulting object of either class predlogispez or predlogilasso is plotted as described above.

Examples

```
library(gRbase)
data(reinis)

fit <- logilasso(reinis, lambdainit=1, lambdamin=0.1)
plot(fit)

fitcv <- logilasso(reinis, lambdainit=1, lambdamin=0.1, cvfold=3)
plot(fitcv)

levellogi <- levelcv(reinis, lambdainit=1, lambdamin=0.1, to.which.int=3, cvfold=3)
plot(levellogi)

pred <- predict(fit, lambda=0.3)
plot(pred)
```

predict.logilasso *Predicts the interaction vector beta of a loglinear interaction model.*

Description

Predicts the interaction vector(s) beta of a loglinear model $\log(p) = X * \text{beta}$, fitted either by logilasso or levelcv. If lambda is specified, this lambda is taken to predict the beta for this value of lambda. If no value for lambda is specified, then the optimal value calculated by cross-validation is taken for objects of class cvlogilasso. For objects of class logilasso where no cross-validation was performed, the whole solution path for all lambdas is returned.

Usage

```
## S3 method for class 'logilasso':
predict(object, lambda = NULL, ...)
## S3 method for class 'cvlogilasso':
predict(object, lambda = NULL, ...)
## S3 method for class 'levellogilasso':
predict(object, lambda = NULL, to.which.int =
NULL, ...)
```

Arguments

object	Object of class levellogilasso, cvlogilasso or class logilasso.
lambda	Value for the penalization parameter lambda, for which the corresponding beta should be calculated.
to.which.int	The number of factors the model should be predicted for.
...	Additional arguments to predict function.

Value

Is either an object of class `predlogilasso` if a value for `lambda` was specified or if the optimal `lambda` can be assessed by cross-validation. Otherwise, if no value for `lambda` was specified and at the same time `cvfold` was chosen to be 1 (no cross-validation) it is of class `predlogispez`. The difference between these two classes is described below.

beta	A predicted value for beta if the object is of class <code>predlogilasso</code> . For the class <code>predlogispez</code> this is a matrix consisting of the columns beta for the whole solution path.
lambda	The lambda(s) for which the beta(s) was/were calculated.
probs	Probabilities according to the model $\text{probs} = \exp(X \cdot \text{beta})$
nls	Negative likelihood score. For details see http://stat.ethz.ch/~dahinden/Paper/BMC.pdf
betapath	The whole solution originally calculated path. For objects of class <code>predlogispez</code> this equals beta.
lambdapath	The lambdas corresponding to the value betapath.
losspath	nls for the whole solution path.

See Also

[logilasso](#)

Examples

```
library(gRbase)
data(reinis)

fit <- logilasso(reinis, lambdainit=1, lambdamin=0.1)
pred1 <- predict(fit, lambda=0.5)
```

```

pred2 <- predict(fit)

fitcv <- logilasso(reinis, lambdainit=1, lambdamin=0.1, cvfold=3)
predcv1 <- predict(fitcv)

levellogi <- levelcv(reinis, lambdainit=1, lambdamin=0.1, to.which.int=3, cvfold=3)
predlevel <- predict(levellogi)

## Methods plot and graphmod exist for all predicted models
## Except for pred2, there is no graphmod method, because no
## lambda was specified

plot(predcv1)
graphmod(predcv1)

```

```
traceplot.logilasso
```

Plots the trajectories.

Description

The trace of all components of beta is plotted against all lambdas which were considered in the solution path. The function can be applied to an object of class `logilasso`. This object can be of various subclasses such as `cvlogilasso`, `predlogilasso` or `predlogilassospez`. Possibly the solution path only consists of one lambda, then the components of the corresponding beta are plotted.

Usage

```
traceplot(beta, ...)
```

Arguments

<code>beta</code>	An object of class <code>logilasso</code> .
<code>...</code>	Additional arguments to the plot function.

Examples

```

library(gRbase)
data(reinis)

fit <- logilasso(reinis, lambdainit=, lambdamin=0.1)
traceplot(fit)

fit2 <- logilasso(reinis, lambdainit=1, lambdamin=0.1, cvfold=3)
traceplot(fit2)

fit3 <- levelcv(reinis, lambdainit=1, lambdamin=0.1, to.which.int=3, cvfold=3)
traceplot(fit3)

```

```
traceplot(predict(fit3))
```

Index

- *Topic **hplot**
 - graphmod, [2](#)
 - plot.logilasso, [8](#)
 - traceplot.logilasso, [10](#)
- *Topic **methods**
 - predict.logilasso, [9](#)
- *Topic **models**
 - levelcv, [3](#)
 - logilasso, [5](#)
- *Topic **package**
 - logilasso-package, [1](#)
- *Topic **regression**
 - logilasso, [5](#)

graphmod, [2](#)

levelcv, [3](#)

logilasso, [2](#), [5](#), [10](#)

logilasso-package, [1](#)

plot.logilasso, [8](#)

plot.predlogilasso
(*plot.logilasso*), [8](#)

plot.predlogispez
(*plot.logilasso*), [8](#)

predict.cvlogilasso
(*predict.logilasso*), [9](#)

predict.levellogilasso
(*predict.logilasso*), [9](#)

predict.logilasso, [9](#)

traceplot (*traceplot.logilasso*),
[10](#)

traceplot.logilasso, [10](#)