Package ‘fastcox’

February 19, 2015

Title Lasso and elastic-net penalized Cox's regression in high
dimensions models using the cocktail algorithm

Version 1.1.1

Date 2011-12-13

Author Yi Yang <yiyang@umn.edu>, Hui Zou <hzou@stat.umn.edu>

Maintainer Yi Yang <yiyang@umn.edu>

Depends Matrix

Description We implement a cocktail algorithm, a good mixture of coordinate decent, the majorization-minimization principle and the strong rule, for computing the solution paths of the elastic net penalized Cox's proportional hazards model.

License GPL-2

URL http://code.google.com/p/fastcox/

Repository CRAN

Date/Publication 2013-10-24 18:01:49

NeedsCompilation yes

R topics documented:

- fastcox-package
- cocktail
- cv.cocktail
- FHT
- plot.cocktail
- plot.cv.cocktail
- predict.cocktail
- print.cocktail

Index 14
Description

We introduce a cocktail algorithm, a good mixture of coordinate decent, the majorization-minimization principle and the strong rule, for computing the solution paths of the elastic net penalized Cox’s proportional hazards model.

Details

Package: fastcox
Type: Package
Version: 1.0.0
Date: 2012-03-26
Depends: Matrix
License: GPL (version 2)
URL: http://code.google.com/p/fastcox/

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


Examples

data(FHT)
m1<-cocktail(x=FHT$x,y=FHT$y,d=FHT$status,alpha=0.5)
predict(m1,newx=FHT$x[1:5,],s=c(0.01,0.005))
predict(m1,type="nonzero")
plot(m1)
cocktail

Fits the regularization paths for the elastic net penalized Cox's model

Description

Fits a regularization path for the elastic net penalized Cox's model at a sequence of regularization parameters lambda.

Usage

```r
cocktail(x, y, d,
nlambda = 100,
lambda.min = ifelse(nobs < nvars, 1e-2, 1e-4),
lambda = NULL,
alpha = 1,
pf = rep(1, nvars),
exclude,
dfmax = nvars + 1,
pmax = min(dfmax * 1.2, nvars),
standardize = TRUE,
eps = 1e-6,
maxit = 3e4)
```

Arguments

- **x**: matrix of predictors, of dimension \(N \times p\); each row is an observation vector.
- **y**: a survival time for Cox models. Currently tied failure times are not supported.
- **d**: censor status with 1 if died and 0 if right censored.
- **nnlambda**: the number of lambda values - default is 100.
- **lambda.min**: given as a fraction of lambda.max - the smallest value of lambda for which all coefficients are zero. The default depends on the relationship between \(N\) (the number of rows in the matrix of predictors) and \(p\) (the number of predictors). If \(N > p\), the default is \(0.0001\), close to zero. If \(N < p\), the default is \(0.01\). A very small value of lambda.min will lead to a saturated fit. It takes no effect if there is user-defined lambda sequence.
- **lambda**: a user supplied lambda sequence. Typically, by leaving this option unspecified users can have the program compute its own lambda sequence based on nnlambda and lambda.min. Supplying a value of lambda overrides this. It is better to supply a decreasing sequence of lambda values than a single (small) value, if not, the program will sort user-defined lambda sequence in decreasing order automatically.
- **alpha**: The elasticnet mixing parameter, with \(0 < \alpha \leq 1\). See details.
- **pf**: separate penalty weights can be applied to each coefficient of \(\beta\) to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and results in that variable always being included in the model. Default is 1 for all variables (and implicitly infinity for variables listed in exclude). See details.
exclude
indices of variables to be excluded from the model. Default is none. Equivalent to an infinite penalty factor.

dfmax
limit the maximum number of variables in the model. Useful for very large \( p \), if a partial path is desired. Default is \( p + 1 \).

pmax
limit the maximum number of variables ever to be nonzero. For example once \( \beta \) enters the model, no matter how many times it exits or re-enters model through the path, it will be counted only once. Default is \( \min(dfmax + 1, p) \).

standardize
logical flag for variable standardization, prior to fitting the model sequence. If \( \text{true} \), \( x \) matrix is normalized such that sum squares of each column \( \sum_{i=1}^{N} x_{ij}^2/N = 1 \). Note that \( x \) is always centered (i.e. \( \sum_{i=1}^{N} x_{ij} = 0 \)) no matter \text{standardize} is \( \text{TRUE} \) or \( \text{FALSE} \). The coefficients are always returned on the original scale. Default is \( \text{true} \).

eps
convergence threshold for coordinate majorization descent. Each inner coordinate majorization descent loop continues until the relative change in any coefficient (i.e. \( \max_j |\beta^\text{new}_j - \beta^\text{old}_j|^2 \)) is less than \( \text{eps} \). Defaults value is \( 1e^{-6} \).

maxit
maximum number of outer-loop iterations allowed at fixed lambda value. Default is \( 1e4 \). If models do not converge, consider increasing \text{maxit}.

Details
The algorithm estimates \( \beta \) based on observed data, through elastic net penalized log partial likelihood of Cox’s model.

\[
\arg \min (-\loglik(Data, \beta) + \lambda * P(\beta))
\]

It can compute estimates at a fine grid of values of \( \lambda \)s in order to pick up a data-driven optimal \( \lambda \) for fitting a ‘best’ final model. The penalty is a combination of l1 and l2 penalty:

\[
P(\beta) = (1 - \alpha)/2||\beta||_2^2 + \alpha||\beta||_1.
\]

\( \alpha=1 \) is the lasso penalty. For computing speed reason, if models are not converging or running slow, consider increasing \text{eps}, decreasing \text{nlambda}, or increasing \text{lambda.min} before increasing \text{maxit}.

FAQ:

Question: “I am not sure how are we optimizing alpha. I can get optimal lambda for each value of alpha. But how do I select optimum alpha?”

Answer: cv.cocktail only finds the optimal lambda given alpha fixed. So to chose an good alpha you need to fit cv on a grid of alpha, say (0.1,0.3, 0.6, 0.9, 1) and choose the one corresponds to the lowest predicted deviance.

Question: “I understand your are referring to minimizing the quantity cv.cocktail$cvm, the mean ‘cross-validated error’ to optimize alpha and lambda as you did in your implementation. However, I don’t know what the equation of this error is and this error is not referred to in your paper either. Do you mind explaining what this is?”

Answer: We first define the log partial-likelihood for the Cox model. Assume \( \hat{\beta}[k] \) is the estimate fitted on \( k \)-th fold, define the log partial likelihood function as

\[
L(Data, \hat{\beta}[k]) = \sum_{s=1}^{S} x_{is}^T \hat{\beta}[k] - \log(\sum_{t \in R_s} \exp(x_{it}^T \hat{\beta}[k])).
\]
Then the log partial-likelihood deviance of the $k$-th fold is defined as

$$D[\text{Data}, k] = -2(L(\text{Data}, \beta[k])).$$

We now define the measurement we actually use for cross validation: it is the difference between the log partial-likelihood deviance evaluated on the full dataset and that evaluated on the on the dataset with $k$-th fold excluded. The cross-validated error is defined as

$$CV - ERR[k] = D(\text{Data}[\text{full}], k) - D(\text{Data}[\text{th fold excluded}], k).$$

Value

An object with S3 class `cocktail`.

- **call**: the call that produced this object
- **beta**: a $p \times \text{length}(\lambda)$ matrix of coefficients, stored as a sparse matrix (dgCMatrix class, the standard class for sparse numeric matrices in the Matrix package). To convert it into normal type matrix use `as.matrix()`. 
- **lambda**: the actual sequence of lambda values used
- **df**: the number of nonzero coefficients for each value of lambda.
- **dim**: dimension of coefficient matrix (ices)
- **npasses**: total number of iterations (the most inner loop) summed over all lambda values
- **jerr**: error flag, for warnings and errors, 0 if no error.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References

http://code.google.com/p/fastcox/

See Also

- `plot.cocktail`

Examples

```r
data(FHT)
m1 <- cocktail(x=FHT$x, y=FHT$y, d=FHT$status, alpha=0.5)
predict(m1, newx=FHT$x[1:5,], s=c(0.01, 0.005))
predict(m1, type="nonzero")
plot(m1)
```
cv.cocktail  

\[ \text{Cross-validation for cocktail} \]

Description

Does k-fold cross-validation for cocktail, produces a plot, and returns a value for lambda. This function is modified based on the cv function from the glmnet package.

Usage

\[
\text{cv.cocktail}(x, y, d, \lambda = \text{NULL}, \text{n folds}=5, \text{fold id}, \ldots)
\]

Arguments

- **x**: matrix of predictors, of dimension \(N \times p\); each row is an observation vector.
- **y**: a survival time for Cox models. Currently tied failure times are not supported.
- **d**: censor status with 1 if died and 0 if right censored.
- **\lambda**: optional user-supplied lambda sequence; default is NULL, and cocktail chooses its own sequence.
- **\text{n folds}**: number of folds - default is 5. Although \text{n folds} can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is \(\text{n folds}=3\).
- **\text{fold id}**: an optional vector of values between 1 and \text{n fold} identifying what fold each observation is in. If supplied, \text{n fold} can be missing.
- **\ldots**: other arguments that can be passed to cocktail.

Details

The function runs cocktail \(\text{n folds}+1\) times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The average error and standard deviation over the folds are computed.

Value

an object of class \text{cv.cocktail} is returned, which is a list with the ingredients of the cross-validation fit.

- **\lambda**: the values of lambda used in the fits.
- **\text{cvm}**: the mean cross-validated error - a vector of length \(\text{length}(\lambda)\).
- **\text{cvsd}**: estimate of standard error of \text{cvm}.
- **\text{cvup}**: upper curve = \text{cvm+cvsd}.
- **\text{cvlo}**: lower curve = \text{cvm-cvsd}.
- **\text{n zero}**: number of non-zero coefficients at each lambda.
- **\text{name}**: a text string indicating partial likelihood (for plotting purposes).
cocktail.fit  a fitted cocktail object for the full data.
lambda.min  The optimal value of lambda that gives minimum cross validation error cvm.
lambda.1se  The largest value of lambda such that error is within 1 standard error of the minimum.

Author(s)
Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References
http://code.google.com/p/fastcox/

http://www.jstatsoft.org/v33/i01/

See Also
cocktail, plot.cv.cocktail.

Examples

data(FHT)
cv1<-cv.cocktail(x=FHT$x[,1:10],y=FHT$y,d=FHT$status,alpha=0.5,nfolds=3)
cv1
plot(cv1)

FHT  FHT data introduced in Simon et al. (2011).

Description
The FHT data set has n = 50 observations and p = 100 predictors. The covariance between predictors Xj and Xj' has the same correlation 0.5. See details in Simon et al. (2011).

Usage
data(FHT)
**Format**

This list object named "FHT" contains the following data:

- **x**: a covariate matrix with 50 rows and 100 columns
- **y**: the distinct failure times
- **status**: the censoring indicator (status = 1 indicates no censoring and status = 0 indicates right censoring)

**References**

[http://www.jstatsoft.org/v33/i01/](http://www.jstatsoft.org/v33/i01/)

[http://www.jstatsoft.org/v39/i05/](http://www.jstatsoft.org/v39/i05/)

**Examples**

```r
data(FHT)
```

---

**plot.cocktail**  
*Plot coefficients from a "cocktail" object*

**Description**

Produces a coefficient profile plot of the coefficient paths for a fitted `cocktail` object. This function is modified based on the `plot` function from the `glmnet` package.

**Usage**

```r
## S3 method for class 'cocktail'
plot(x, xvar = c("norm", "lambda"), color = FALSE, label = FALSE, ...)
```

**Arguments**

- **x**: fitted `cocktail` model
- **xvar**: what is on the X-axis. "norm" plots against the L1-norm of the coefficients, "lambda" against the log-lambda sequence.
- **color**: if TRUE, plot the curves with rainbow colors. FALSE is gray colors. Default is FALSE
- **label**: if TRUE, label the curves with variable sequence numbers. Default is FALSE
- **...**: other graphical parameters to plot
Details

A coefficient profile plot is produced.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References

http://code.google.com/p/fastcox/

*Journal of Statistical Software,* Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/

http://www.jstatsoft.org/v39/i05/

Examples

data(FHT)
m1<cocktail(x=FHT$x,y=FHT$y,d=FHT$status, alpha=0.5)
par(mfrow=c(1,3))
plot(m1) # plots against the L1-norm of the coefficients
plot(m1,xvar="lambda",label=TRUE) # plots against the log-lambda sequence
plot(m1,color=TRUE)

---

plot.cv.cocktail  plot the cross-validation curve produced by cv.cocktail

Description

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the lambda values used. This function is modified based on the plot.cv function from the glmnet package.

Usage

```r
## S3 method for class 'cv.cocktail'
plot(x, sign.lambda, ...)
```
Arguments

- `x`: fitted `cv.cocktail` object
- `sign.lambda`: either plot against `log(lambda)` (default) or its negative if `sign.lambda=-1`.
- `...`: other graphical parameters to plot

Details

A plot is produced.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


See Also

- `cv.cocktail`

Description

Similar to other predict methods, this function predicts fitted values, link function and more from a fitted `cocktail` object. This function is modified based on the `predict` function from the `glmnet` package.

Usage

```r
## S3 method for class 'cocktail'
predict(object, newx, s=NULL, type=c("link", "response", "coefficients", "nonzero"), ...)
```
Arguments

- **object**: fitted cocktail model object.
- **newx**: matrix of new values for x at which predictions are to be made. Must be a matrix.
- **s**: value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
- **type**: type of prediction required.
  - Type "link" gives the linear predictors for Cox’s model.
  - Type "response" gives the fitted relative-risk for Cox’s model.
  - Type "coefficients" computes the coefficients at the requested values for s.
  - Type "nonzero" returns a list of the indices of the nonzero coefficients for each value of s.
- **...**: Not used. Other arguments to predict.

Details

- s is the new vector at which predictions are requested. If s is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right lambda indices.

Value

- The object returned depends on type.

Author(s)

- Yi Yang and Hui Zou
- Maintainer: Yi Yang <yiyang@umn.edu>

References

  
  http://code.google.com/p/fastcox/

  
  http://www.jstatsoft.org/v33/i01/

  
  http://www.jstatsoft.org/v39/i05/
See Also

`coef` method

Examples

data(FHT)
m1 <- cocktail(x=FHT$x, y=FHT$y, d=FHT$status, alpha=0.5)
predict(m1, type="nonzero")
predict(m1, s=c(0.01, 0.005), type="coefficients")
predict(m1, newx=FHT$x[1:5], type="response")
predict(m1, newx=FHT$x[1:3], s=0.01, type="link")

print.cocktail  

print a cocktail object

Description

Print a summary of the cocktail path at each step along the path. This function is modified based on 
the print function from the glmnet package.

Usage

```r
# S3 method for class 'cocktail'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

- `x` fitted `cocktail` object
- `digits` significant digits in printout
- `...` additional print arguments

Details

The call that produced the `cocktail` object is printed, followed by a two-column matrix with 
columns `df` and `lambda`. The `df` column is the number of nonzero coefficients.

Value

a two-column matrix, the first columns is the number of nonzero coefficients and the second column 
is `lambda`.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>
References

http://code.google.com/p/fastcox/

http://www.jstatsoft.org/v33/i01/

http://www.jstatsoft.org/v39/i05/

Examples

data(FHT)
m1<-cocktail(x=FHT$x,y=FHT$y,d=FHT$status, alpha=0.5)
print(m1)
Index

+ Topic **datasets**
  FHT, 7
+ Topic **models**
  cocktail, 3
  cv.cocktail, 6
  plot.cocktail, 8
  plot.cv.cocktail, 9
  predict.cocktail, 10
  print.cocktail, 12
+ Topic **package**
  fastcox-package, 2
+ Topic **regression**
  cocktail, 3
  cv.cocktail, 6
  plot.cocktail, 8
  plot.cv.cocktail, 9
  predict.cocktail, 10
  print.cocktail, 12

cocktail, 3, 5–8, 10–12
coef, 12
cv.cocktail, 6, 6, 10
cv.survpath (cv.cocktail), 6

fastcox-package, 2
FHT, 7

plot.cocktail, 8
plot.cv.cocktail, 7, 9
predict.cocktail, 10
predict.survpath (predict.cocktail), 10
print.cocktail, 12