Package 'apple'

July 2, 2014

Type Package

Title Approximate Path for Penalized Likelihood Estimators

Version 0.3

Date 2011-11-11

Author Yi Yu, Yang Feng

Maintainer Yi Yu <yuyi@fudan.edu.cn>

Depends MASS

Description Approximate Path for Penalized Likelihood Estimators for Generalized Linear Models penalized by LASSO or MCP

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2012-01-20 15:18:41

NeedsCompilation yes

R topics documented:

apple 2
cv.apple 4
plot.apple 6
predict.apple 7

Index **9**

apple *Approximate Path for Penalized Likelihood Estimator*

Description

Fit a generalized linear model via penalized maximum likelihood. The regularization path is computed for the LASSO or MCP penalty at a grid of values for the regularization parameter lambda. Can deal all shapes of data, including very large sparse data matrices. Fits binomial-logistic and poisson regression models.

Usage

```
apple(X, y, family="binomial", penalty = "LASSO", gamma, cha.poi = 1,
eps = 1e-15, lam.list, lambda.min.ratio, max.iter = 100, max.num,
n.lambda = 100)
```

Arguments

X	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable, of dimension nobs x 1. non-negative counts for family="poisson", binary for family="binomial".
family	response type.
penalty	LASSO and MCP are provided.
gamma	the MCP concavity parameter.
cha.poi	the value used to change from Newton Raphson correction to Coordinate Descent correction, which is the α in the following inequality, $k > \alpha\sqrt{n}$, where k is the size of current active set. when this inequality holds, the correction method changes from Newton Raphson to Coordinate Descent.
eps	the precision used to test the convergence.
lam.list	a user supplied λ sequence. typical usage is to have the program compute its own lambda sequence based on lambda.min.ratio and n.lambda. supplying a value of λ overrides this.
lambda.min.ratio	optional input. smallest value for lambda, as a fraction of max.lam, the (data derived) entry value. the default depends on the sample size n relative to the number of variables p. if n > p, the default is 0.0001. otherwise, the default is 0.01.
max.iter	maximum number of iteration in the computation.
max.num	optional input. maximum number of nonzero coefficients.
n.lambda	the number of lambda values.

Value

a0	intercept vector of length(lambda).
beta	nvar x length(lambda) matrix of coefficients.
lambda	the list of lambda derived the solution path.
ebic	the list of EBIC values.
ebic.loc	the location of the EBIC selected solution in the path.
family	the family of the supplied dataset.

Author(s)

Yi Yu and Yang Feng

References

Yi Yu and Yang Feng, APPLE: Approximate Path for Penalized Likelihood Estimator, manuscript.

See Also

plot.apple, cv.apple and predict.apple

Examples

```
p=10
n=200
d=5
coefs=c(3,1.5,0,0,2)
intercept=0
beta=rep(0,p)
beta[1:d]=coefs
X=matrix(rnorm(p*n), nrow=n)
mu=1/(1+exp(-X %*% beta-intercept))
y=rbinom(n,1,mu)


fit.apple=apple(X, y, family= "binomial")


plot(fit.apple)
```

cv.apple *cross validation for apple path*

Description

Does K-fold cross validation for apple.

Usage

```
cv.apple(X, y, family="binomial", penalty="LASSO", gamma,  K = 10,
alpha=0, seed = 1, cha.poi = 1, eps = 1e-15, lambda.min.ratio,
max.iter = 100, max.num, n.lambda = 100)
```

Arguments

X	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable, of dimension nobs x 1. non-negative counts for family="poisson", binary for family="binomial".
family	response type.
penalty	LASSO and MCP are provided.
gamma	the MCP concavity parameter.
K	number of folds used in cross validation. The default it 10.
alpha	weight used in the cross validation cost function, with $Q(\lambda) = \alpha Dev(\lambda) + (1-\alpha)s(\lambda)\log n/n$.
seed	random seed used to sample the training sets and test sets
cha.poi	the value used to change from Newton Raphson correction to Coordinate Descent correction, which is the α in the following inequality, $k > \alpha\sqrt{n}$, where k is the size of current active set. when this inequality holds, the correction method changes from Newton Raphson to Coordinate Descent.
eps	the precision used to test the convergence.
lambda.min.ratio	optional input. smallest value for lambda, as a fraction of max.lam, the (data derived) entry value. the default depends on the sample size n relative to the number of variables p. if n > p, the default is 0.0001. otherwise, the default is 0.01.
max.iter	maximum number of iteration in the computation.
max.num	optional input. maximum number of nonzero coefficients.
n.lambda	the number of lambda values.

Value

cv	list of cross validation loss
lambda	list of lambda
a0	the list of intercept
beta	the list of coefficients
cv.loc	location of cv selected solution in the path
ebic.loc	the location of the EBIC selected solution in the path
cv.beta	cross validation selected beta
ebic.beta	ebic selected beta
cv.a0	cv selected intercept
ebic.a0	ebic selected intercept

Author(s)

Yi Yu and Yang Feng

References

Yi Yu and Yang Feng, APPLE: Approximate Path for Penalized Likelihood Estimator, manuscript.

See Also

plot.apple, apple and predict.apple

Examples

```
p=10
n=200
d=5
coefs=c(3,1.5,0,0,2)
intercept=0
beta=rep(0,p)
beta[1:d]=coefs
X=matrix(rnorm(p*n), nrow=n)
mu=1/(1+exp(-X %*% beta-intercept))
y=rbinom(n,1,mu)


fit.cv=cv.apple(X, y, family="binomial", alpha=0.25, K=5)


plot(fit.cv)
```

`plot.apple` *plot apple path*

Description

Plot the solution path generated from apple.

Usage

```
## S3 method for class 'apple'
plot(x, col = "black", add = FALSE,
main = "apple", type = "l", lty = 1, ...)
```

Arguments

`x`	the object used to plot the path. object derived by `"apple"` and `"cv.apple"` are both available here. for `"cv.apple"` object, ebic and cv selected solutions are also lined in the graph.
`col`	color used to plot the paths, for details please see the usage of `col` in `plot`.
`add`	whether or not to add this plot to an existing one.
`main`	title of the plot. default is `main="apple"`
`type`	what type of plot should be drawn, for details please see the usage of `plot`.
`lty`	the line type, for details please see the usage of `par`
`...`	see `matplot`.

Details

if the object is a cv result, then both of ebic and cv selected result will be marked on the graph by solid and dotted vertical lines, respectively.

Author(s)

Yi Yu and Yang Feng

References

Yi Yu and Yang Feng, APPLE: Approximate Path for Penalized Likelihood Estimator, manuscript.

See Also

`apple`, `cv.apple` and `predict.apple`

predict.apple *Model prediction based on a fitted apple/cv.apple object.*

Description

Similar to other predict methods, this function returns predictions from a fitted "apple" or "cv.apple" object.

Usage

```
## S3 method for class 'apple'
predict(object, X, which = 1:length(object$lambda),
type = c("link", "response", "class"),...)
```

Arguments

object	fitted "apple" or "cv.apple" model object.
X	matrix of values at which predictions are to be made.
which	indices of the penalty parameter lambda at which predictions are required. by default, all indices are returned.
type	type of prediction: "link" returns the linear predictors; "response" gives the fitted values; "class" returns the binomial outcome with the highest probability.
...	see matplot.

Value

The object returned depends on type.

Author(s)

Yi Yu and Yang Feng

References

Yi Yu and Yang Feng, APPLE: Approximate Path for Penalized Likelihood Estimator, manuscript.

See Also

apple, cv.apple and plot.apple

Examples

```
p=10
n=200
d=5
coefs=c(3,1.5,0,0,2)
intercept=0
beta=rep(0,p)
beta[1:d]=coefs
set.seed(2)
X=matrix(rnorm(p*n), nrow=n)
mu=1/(1+exp(-X %*% beta-intercept))
y=rbinom(n,1,mu)


fit.apple=apple(X, y, family="binomial")


set.seed(3)
testX=matrix(rnorm(p*n), nrow=n)

predict(fit.apple,testX,type="link")
predict(fit.apple,testX,type="response")
predict(fit.apple,testX,type="class")


fit=cv.apple(X, y, family="binomial", alpha=0)
predict(fit.apple,testX,type="link", which = fit$cv.loc)
predict(fit.apple,testX,type="response", which = fit$cv.loc)
predict(fit.apple,testX,type="class", which = fit$cv.loc)
```

Index

apple, 2, *5–7*

cv.apple, *3*, 4, *6, 7*

plot.apple, *3*, *5*, 6, *7*
predict.apple, *3*, *5*, *6*, 7