Package 'LogitNet'

February 19, 2015

Version 0.1-1

Date 2009-11-20

Title Infer network based on binary arrays using regularized logistic regression

Author Pei Wang <pwang@fhcrc.org>, Dennis Chao <dchao@fhcrc.org>, Li Hsu <lih@fhcrc.org>

Maintainer Pei Wang <pwang@fhcrc.org>

Description LogitNet is developed for inferring network of binary variables under the high-dimension-low-sample-size setting

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-11-25 08:38:21

NeedsCompilation yes

R topics documented:

`LogitNet` *Fit a LogitNet model for a given tuning parameter value.*

Description

Fit a LogitNet model by performing joint sparse logistic regressions for the binary data.

Usage

```
LogitNet(X.m, weight.m, lambda, beta.ini=NULL)
```

Arguments

X.m	numeric matrix (n by p). Columns are for variables and rows are for samples. Missing values are not allowed.
weight.m	numeric matrix (p by p). This weight matrix allows the coefficients in the regression model to be penalized to varying degree, such that the spatial correlations along the genome profiles are taken into account. Can use the output from LogitNet.weight function.
lambda	numeric scalar. It gives the l_1 norm penalty parameter.
beta.ini	numeric matrix (p by p). This matrix serves as the initial estimation of the coefficient matrix. The default is NULL.

Details

LogitNet is developed for infering interaction network of binary variables. The method is based on penalized logistic regression with an extension to account for spatial correlation in the genomic instability data. (Wang et al., 2009).

Value

beta	the estimated coefficient matrix (p by p) from the LogitNet model.

Author(s)

Pei Wang, Dennis Chao, Li Hsu

References

Pei Wang, Dennis Chao, Li Hsu, "Learning oncogenic pathways from binary genomic instability data", Biometrics, (submitted 2009, July)

Examples

```
######################### get data example

data(LogitNet.data)
data.m=LogitNet.data$data.m
chromosome=LogitNet.data$chromosome
p=ncol(data.m)

######################### specify the penalty parameter
lambda.n=5
lambda.v=exp(seq(log(13), log(30), length=lambda.n))

######################### calculate the weight matrix
w.m=LogitNet.weight(data.m, chr=chromosome)

######################### perform cross validation to select lambda
if(0) ### this part will take 10 minutes.
{
```

```
try.CV=LogitNet.CV(data.m, w.m, lambda.v, fold=5)
temp=apply(try.CV[[3]], 2, sum)
index=which.max(temp)
}
index=2
######################## estimate the model at selected lambda

result=LogitNet(data.m, w.m, lambda.v[index]) ###20-30 seconds

######################## illustrate the result similar to Figure 3 of Wang et al. (2009)).

temp=result
diag(temp)=0

par(cex=1.8)
image(1:p, 1:p, temp!=0, col=c("white", "red"), axes=FALSE, xlab="Marker Loci", ylab="Marker Loci")
abline(h=(0:5)*p/6+p/6/2, col=4, lty=3, lwd=0.8)
abline(v=(0:5)*p/6+p/6/2, col=4, lty=3, lwd=0.8)
axis(1, at=c(1,1:6*100), labels=c(1,1:6*100))
axis(2, at=c(1,1:6*100), labels=c(1,1:6*100))
axis(3, at=(0:5)*p/6+p/6/2, labels=c("A", "B", "C", "D", "E", "F"), col.axis=4, tick=FALSE)
axis(4, at=(0:5)*p/6+p/6/2, labels=c("A", "B", "C", "D", "E", "F"), col.axis=4, tick=FALSE)

lab.v=c("A", "B", "C", "D", "E", "F")

cut=30
for(i in 0:4)
{
   cur=i*p/6+p/6/2
   cur2=(i+1)*p/6+p/6/2

   x.cur=c(cur-cut, cur, cur+cut, cur)
   y.cur=c(cur2, cur2-cut, cur2, cur2+cut)
   polygon(x.cur, y.cur, border=grey(0.5))
   polygon(y.cur, x.cur, border=grey(0.5))
}
```

LogitNet.CV *Fit LogitNet models with cross validation.*

Description

Fit LogitNet models for a series of tuning parameters and return the cross valication error.

Usage

```
LogitNet.CV(X.m, weight.m, lambda.v, fold=5)
```

Arguments

`X.m`	numeric matrix (n by p). Columns are for variables and rows are for samples. Missing values are not allowed.
`weight.m`	numeric matrix (p by p). This weight matrix allows the coefficients in the regression model to be penalized to varying degree, such that the spatial correlation along the genome profiles is taken into account. It can use the output from LogitNet.weight function.
`lambda.v`	numeric vector. It is a sequence of l_1 norm penalty parameters.
`fold`	numeric scaler. It specifies the fold number in cross validation. The default is 5.

Details

LogitNet.CV helps to select the tuning parameter through cross validation. LogitNet is developed for infering interaction network of binary variables. The method is based on penalized logistic regression with an extension to account for spatial correlation in the genomic instability data. (Wang, Chao and Hsu, 2009).

Value

A list with four components

`beta_reg`	A numeric array with dimention (P, P, Fold, lambda.n), which records the estimated coefficient matrix at each lambda from the penalized model. Here `lambda.n` is the lenght of `lambda.v`.
`beta_unbias`	A numeric array with dimention (P, P, Fold, lambda.n), which records the estimated coefficient matrix at each lambda from the un-penalized model (refit LogitNet only using the selected variables).
`likelihood.test`	A numeric matrix (Fold by lambda.n), which records the likelihood of each logistic regression on the testing data for each cross validation fold.
`likelihood.train`	A numeric matrix (Fold by lambda.n), which records the likelihood of each logistic regression on the training data for each cross validation fold.

Author(s)

Pei Wang, Dennis Chao, Li Hsu

References

Pei Wang, Dennis Chao, Li Hsu, "Learning oncogenic pathways from binary genomic instability data", Biometrics, (submitted 2009, July)

Examples

```
######################### get data example

data(LogitNet.data)
data.m=LogitNet.data$data.m
chromosome=LogitNet.data$chromosome
p=ncol(data.m)

######################### specify the penalty parameter
lambda.n=5
lambda.v=exp(seq(log(13), log(30), length=lambda.n))

######################### calculate the weight matrix
w.m=LogitNet.weight(data.m, chr=chromosome)

######################### perform cross validation to select lambda
if(0) ### this part will take 10 minutes.
{
try.CV=LogitNet.CV(data.m, w.m, lambda.v, fold=5)
temp=apply(try.CV[[3]], 2, sum)
index=which.max(temp)
}
index=2
######################### estimate the model at selected lambda

result=LogitNet(data.m, w.m, lambda.v[index]) ###20-30 seconds

######################### illustrate the result similar to Figure 3 of Wang et al. (2009)).

temp=result
diag(temp)=0

par(cex=1.8)
image(1:p, 1:p, temp!=0, col=c("white", "red"), axes=FALSE, xlab="Marker Loci", ylab="Marker Loci")
abline(h=(0:5)*p/6+p/6/2, col=4, lty=3, lwd=0.8)
abline(v=(0:5)*p/6+p/6/2, col=4, lty=3, lwd=0.8)
axis(1, at=c(1,1:6*100), labels=c(1,1:6*100))
axis(2, at=c(1,1:6*100), labels=c(1,1:6*100))
axis(3, at=(0:5)*p/6+p/6/2, labels=c("A", "B", "C", "D", "E", "F"), col.axis=4, tick=FALSE)
axis(4, at=(0:5)*p/6+p/6/2, labels=c("A", "B", "C", "D", "E", "F"), col.axis=4, tick=FALSE)

lab.v=c("A", "B", "C", "D", "E", "F")

cut=30
for(i in 0:4)
{
   cur=i*p/6+p/6/2
   cur2=(i+1)*p/6+p/6/2

   x.cur=c(cur-cut, cur, cur+cut, cur)
   y.cur=c(cur2, cur2-cut, cur2, cur2+cut)
```

```
   polygon(x.cur, y.cur, border=grey(0.5))
   polygon(y.cur, x.cur, border=grey(0.5))
}
```

`LogitNet.data` *Example Data for LogitNet package*

Description

A list containing an example data for package `LogitNet`

Usage

```
data(LogitNet.data)
```

Details

`data.m` is an simulated example based on the chain pathway described in Section 3 of Wang et al. 2009.

Value

LogitNet.data is a list of two components:

`data.m`	a numeric matrix consisting of 200 rows (samples) and 600 columns (genes).
`chromosome`	a numeric vector of length 600.

References

Pei Wang, Dennis Chao, Li Hsu, "Learning oncogenic pathways from binary genomic instability data", Biometrics, (submitted 2009, July)

`LogitNet.weight` *Derive the weight matrix for fitting the LogitNet model.*

Description

Derive the weight matrix for fitting the LogitNet model.

Usage

```
LogitNet.weight(X.m, chr)
```

Arguments

X.m	numeric matrix (n by p). Columns are for genes/loci and rows are for samples. Missing values are not allowed. The genes/loci are ordered according to their positions on the genome.
chr	numeric vector of length p. This vector gives the chromosome information of each gene/locus.

Details

This function returns a weight matrix charactering the spatial correlations along the genome. This matrix provides the value for one input parameter of function LogitNet(). LogitNet is developed for infering interaction network of binary variables. The method is based on penalized logistic regression with an extension to account for spatial correlation in the genomic instability data. (Wang, Chao and Hsu, 2009).

Value

w.s	numeric matrix (p by p), which characterizes the spatial correlation along the genome for each gene/locus.

Author(s)

Pei Wang, Dennis Chao, Li Hsu

References

Pei Wang, Dennis Chao, Li Hsu, "Learning oncogenic pathways from binary genomic instability data", Biometrics, (submitted 2009, July)

Examples

```
######################### obtain a data example

data(LogitNet.data)
data.m=LogitNet.data$data.m
chromosome=LogitNet.data$chromosome
p=ncol(data.m)

######################### specify the penalty parameter
lambda.n=5
lambda.v=exp(seq(log(13), log(30), length=lambda.n))

######################### calculate the weight matrix
w.m=LogitNet.weight(data.m, chr=chromosome)

######################### perform cross validation to select lambda
if(0) ### this part will take 10 minutes.
{
try.CV=LogitNet.CV(data.m, w.m, lambda.v, fold=5)
```

```
temp=apply(try.CV[[3]], 2, sum)
index=which.max(temp)
}
index=2
######################### estimate the model at selected lambda values

result=LogitNet(data.m, w.m, lambda.v[index]) ###20-30 seconds

######################### illustrate the result similar to Figure 3 of Wang et al. (2009)).

temp=result
diag(temp)=0

par(cex=1.8)
image(1:p, 1:p, temp!=0, col=c("white", "red"), axes=FALSE, xlab="Marker Loci", ylab="Marker Loci")
abline(h=(0:5)*p/6+p/6/2, col=4, lty=3, lwd=0.8)
abline(v=(0:5)*p/6+p/6/2, col=4, lty=3, lwd=0.8)
axis(1, at=c(1,1:6*100), labels=c(1,1:6*100))
axis(2, at=c(1,1:6*100), labels=c(1,1:6*100))
axis(3, at=(0:5)*p/6+p/6/2, labels=c("A", "B", "C", "D", "E", "F"), col.axis=4, tick=FALSE)
axis(4, at=(0:5)*p/6+p/6/2, labels=c("A", "B", "C", "D", "E", "F"), col.axis=4, tick=FALSE)

lab.v=c("A", "B", "C", "D", "E", "F")

cut=30
for(i in 0:4)
{
   cur=i*p/6+p/6/2
   cur2=(i+1)*p/6+p/6/2

   x.cur=c(cur-cut, cur, cur+cut, cur)
   y.cur=c(cur2, cur2-cut, cur2, cur2+cut)
   polygon(x.cur, y.cur, border=grey(0.5))
   polygon(y.cur, x.cur, border=grey(0.5))
}
```

Index