Package 'JGL'

December 1, 2018

Type Package

Title Performs the Joint Graphical Lasso for Sparse Inverse Covariance Estimation on Multiple Classes

Version 2.3.1

Date 2018-11-30

Author Patrick Danaher

Maintainer Patrick Danaher <pdanaher@uw.edu>

Description The Joint Graphical Lasso is a generalized method for estimating Gaussian graphical models/ sparse inverse covariance matrices/ biological networks on multiple classes of data. We solve JGL under two penalty functions: The Fused Graphical Lasso (FGL), which employs a fused penalty to encourage inverse covariance matrices to be similar across classes, and the Group Graphical Lasso (GGL), which encourages similar network structure between classes. FGL is recommended over GGL for most applications. Reference: Danaher P, Wang P, Witten DM. (2013) <doi:10.1111/rssb.12033>.

Depends igraph

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2018-11-30 23:40:15 UTC

RoxygenNote 6.1.0

R topics documented:

`JGL-package` *Joint Graphical Lasso*

Description

Runs the Fused Graphical Lasso and the Group Graphical Lasso for network estimation and sparse inverse covariance estimation across multiple classes of data.

Details

Package:	JGL
Type:	Package
Version:	2.3
Date:	2013-04-16
License:	GPL (>= 2)
LazyLoad:	yes

The Fused Graphical Lasso (FGL) and the Group Graphical Lasso (GGL) are two methods for estimating sparse inverse covariance matrices that are similar across classes. A motivating example is the analysis of gene expression data from tumor and healthy cells: FGL and GGL allow joint estimation of gene expression conditional dependency networks in both cancer and healthy cells FGL is recommended over GGL for most purposes. The function JGL can implement either of these methods.

The JGL package includes a number of functions to help analyze estimated networks: subnetworks(), net.degree(), net.edges(), net.hubs(), net.neighbors(), print.jgl() and plot.jgl(). These functions rely on the igraph package.

A large number of other functions are called by the above functions, and are not generally useful when called by the user.

Author(s)

Patrick Danaher

Maintainer: Patrick Danaher - pdanaher at uw dot edu

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run FGL:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1)
str(fgl.results)
print.jgl(fgl.results)
## get subnetwork membership of FGL results:
subnetworks(fgl.results$theta)
```

---

crit *Calculate the critical value of the FGL objective funciton.*

Description

crit() calculates the critical value of the FGL objective funciton. It is used to confirm that the FGL algorithm is converging.

Usage

```
crit(theta, S, n, lam1, lam2, penalize.diagonal)
```

Arguments

theta	A list of pXp inverse covariance matrices.
S	A list of pXp empirical covariance matrices.
n	A vector of sample sizes to attribute to each of the K data matrices. n controls the relative weights of the classes: for example, with n==c(1,1), each class's theta will be penalized equally.
lam1	The tuning parameter for the graphical lasso penalty.
lam2	The tuning parameter for the fused lasso penalty.
penalize.diagonal	Logical value determing whether the graphical lasso penalty should also be applied to the diagonal of the inverse covariance matrices.

Details

A function called by FGL to calculate the critical value of the objective function.

Value

crit, the critical value of the list of inverse covariance matrices.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

`example.data` *Toy two-class gene expression dataset.*

Description

A dataset with 200 genes and 2 classes of data, each with 100 observations. The two classes' data matrices are stored in a list.

Usage

```
data(example.data)
```

Format

The format is: List of 2 $: num [1:100, 1:200] 0.395 -2.03 -1.704 -0.469 1.75 ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: chr [1:200] "gene 1" "gene 2" "gene 3" "gene 4" ... $: num [1:100, 1:200] -1.548 1.45 -0.812 -0.589 0.69 ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: chr [1:200] "gene 1" "gene 2" "gene 3" "gene 4" ...

Examples

```
data(example.data)
str(example.data)
```

gcrit *Calculate the critical value of the GGL objective funciton.*

Description

gcrit() calculates the critical value of the GGL objective funciton. It is used to confirm that the GGL algorithm is converging.

Usage

```
gcrit(theta, S, n, lam1, lam2, penalize.diagonal)
```

Arguments

theta	A list of pXp inverse covariance matrices.
S	A list of pXp empirical covariance matrices.
n	A vector of sample sizes to attribute to each of the K data matrices. n controls the relative weights of the classes: for example, with n==c(1,1), each class's theta will be penalized equally.
lam1	The tuning parameter for the graphical lasso penalty.
lam2	The tuning parameter for the group lasso penalty.
penalize.diagonal	Logical, determining whether the penalties will be applied to the diagonal elements of the theta matrices.

Details

A function called by GGL to calculate the critical value of the objective function.

Value

crit, the critical value of the list of inverse covariance matrices.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

JGL *Joint Graphical Lasso*

Description

Solve the Joint Graphical Lasso

Usage

```
JGL(Y,penalty="fused",lambda1,lambda2,rho=1,weights="equal",penalize.diagonal=FALSE,
maxiter=500,tol=1e-5,warm=NULL,return.whole.theta=FALSE, screening="fast",
truncate = 1e-5)
```

Arguments

Y	A list of nXp data matrices.
penalty	Determines whether lambda2 controls a "fused" or "group" lasso penalty. Must take value "fused" or "group".
lambda1	The tuning parameter for the graphical lasso penalty.
lambda2	The tuning parameter for the fused or group lasso penalty.
rho	A step size parameter. Large values decrease step size.
weights	Determines the putative sample size of each class's data. Allowed values: a vector with length equal to the number of classes; "equal", giving each class weight 1; "sample.size", giving each class weight corresponding to its sample size.
penalize.diagonal	If penalty=="fused", determines whether lambda1 is applied to the diagonal of theta. If penalty=="group", determines whether lambda1 and lambda2 are applied to the diagonal of theta.
maxiter	Maximum number of iterations.
tol	Determines convergence criterion.
warm	Input a warm start to theta in the form of a K-length list of pXp matrices.
return.whole.theta	If TRUE, each class's inverse covariance matrix is returned whole. If FALSE, the inverse covariance matrix is only returned over the connected nodes, and only the diagonal of the matrix is returned over the unconnected nodes.
screening	"fast" or "memory.efficient". Use of "fast" is recommended unless the number of features prohibits storage of a pXp matrix. For very high dimension data, screening="memory.efficient" will allow a solution with a much longer computation time.
truncate	Defaults to 1e-5. At convergence, all values of theta below this number will be set to zero.

Details

This function can solve both the Fused Graphical Lasso and the Group Graphical Lasso.

Value

`theta`	A list of the estimated inverse covariance matrices, over all nodes if return.whole.theta==TRUE and over only the connected nodes if return.whole.theta==FALSE
`diag.theta.unconnected`	Returned only if return.whole.theta==FALSE. A list of vectors, each vector the estimated diagonal of an inverse covariance matrix over the unconnected nodes.
`connected`	A logical vector identifying whether each node is connected.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run fgl:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1)
str(fgl.results)
print.jgl(fgl.results)
## run ggl:
ggl.results = JGL(Y=example.data,penalty="group",lambda1=.15,lambda2=.2,return.whole.theta=TRUE)
str(ggl.results)
print.jgl(ggl.results)
```

---

net.degree *List the degree of every node in all classes.*

Description

For each class, lists the degree of every node.

Usage

```
net.degree(theta)
```

Arguments

theta	A list of pXp matrices, each an estimated sparse inverse covariance matrix. (For example, the result of FGL or GGL.)

Value

degree, a list of p-length vectors, each giving the degree of all p nodes in the network for the corresponding class.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run fgl:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1)
## get degree list:
net.degree(fgl.results$theta)
```

---

net.edges *List the edges in a network*

Description

For each class, list every pair of connected nodes.

Usage

```
net.edges(theta)
```

Arguments

theta	A list of pXp matrices, each an estimated sparse inverse covariance matrix. (For example, the result of FGL or GGL.)

Value

edges, a K-length list, each element of the list an igraph.es object detailing all pairs of connected nodes in the class.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run fgl:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1)
## get edges list:
net.edges(fgl.results$theta)
```

---

net.hubs *Get degrees of most connected nodes.*

Description

List the degrees of the most connected nodes in each class.

Usage

```
net.hubs(theta, nhubs = 10)
```

Arguments

theta	A list of pXp matrices, each an estimated sparse inverse covariance matrix. (For example, the result of FGL or GGL.)
nhubs	The number of hubs to be identified. net.hubs() will list the degree of the nhubs most connected nodes in each class.

Value

hubs, a list of length K, each element of which is a vector giving the degree of the most connected nodes in the corresponding class.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run fgl:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1)
## get hubs list:
net.hubs(fgl.results$theta)
```

---

net.neighbors *Get network neighbors of a node*

Description

For each class, returns the names of the nodes connected to a given node.

Usage

```
net.neighbors(theta, index)
```

Arguments

theta	A list of pXp matrices, each an estimated sparse inverse covariance matrix. (For example, the result of FGL or GGL.)
index	The row number of the node to be investigated.

Value

neighbors, a list of length K, each element of which is a vector of the row names of the nodes neighboring the node of interest.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run fgl:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1,return.whole.theta=TRUE)
## get neighbors of gene 195:
net.neighbors(fgl.results$theta,index=195)
```

`screen.fgl` *Quickly identify connected features in the solution to FGL*

Description

Applies the FGL screening rule to identify (before running FGL) which features are connected (have degree > 0 in any class) or unconnected in the solution. screen.fgl returns exactly the right list of connected nodes when K=2. When K is larger than 2, screen.fgl applies a weaker condition that screens out many, but not all unconnected nodes. This algorithm is set up to be memory-efficient, but not fast: it can be applied to very large dimension datasets, but it will take time to run.

Usage

```
screen.fgl(Y, lambda1, lambda2, weights = "equal")
```

Arguments

`Y`	A list of nXp data matrices.
`lambda1`	The tuning parameter for the graphical lasso penalty. Must be greater than or equal to 0.
`lambda2`	The tuning parameter for the fused lasso penalty. Must be greater than or equal to 0.
`weights`	The weights to assign to each class. The higher a class's weights, the weaker the effect of the penalties on its estimated inverse covariance matrix. If "equal", the classes are weighted equally, regardless of sample size. If "sample.size", the classes are weighted by sample size. Custom weightings are achievable by entering a vector of K weights.

Value

connected, a logical vector identifying the connected nodes.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## which nodes will be connected?
screen.fgl(example.data,lambda1=.2,lambda2=.1,weights="equal")
```

`screen.ggl` *Quickly identify connected features in the solution to GGL*

Description

Applies the GGL screening rule to identify (before running GGL) which features are connected (have degree > 0 in any class) in the solution. This algorithm is set up to be memory-efficient, but not fast: it can be applied to very large dimension datasets, but it will take time to run.

Usage

```
screen.ggl(Y, lambda1, lambda2, weights = "equal")
```

Arguments

`Y`	A list of nXp data matrices.
`lambda1`	The tuning parameter for the graphical lasso penalty. Must be greater than or equal to 0.
`lambda2`	The tuning parameter for the group lasso penalty. Must be greater than or equal to 0.
`weights`	The weights to assign to each class. The higher a class's weights, the weaker the effect of the penalties on its estimated inverse covariance matrix. If "equal", the classes are weighted equally, regardless of sample size. If "sample.size", the classes are weighted by sample size. Custom weightings are achievable by entering a vector of K weights.

Value

connected, a logical vector identifying the connected nodes.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## which nodes will be connected?
screen.ggl(example.data,lambda1=.3,lambda2=.3,weights="equal")
```

`subnetworks` *Identify subnetwork membership*

Description

For each class, returns lists of all features belonging to subnetworks. (A subnetwork is defined as a collection of features C for which theta[C,!C]==0, and within which no further subnetworks can be identified. In other words, a block in the block diagonal structure of theta, or a set of features that can be connected through theta's edges.)

Usage

```
subnetworks(theta)
```

Arguments

`theta`	A list of pXp matrices, each an estimated sparse inverse covariance matrix. (For example, the result of FGL or GGL.)

Value

A list length K, each element of which is a list of subnetworks in class K. Each subnetwork is represented as a vector of feature names.

Author(s)

Patrick Danaher

References

Patrick Danaher, Pei Wang and Daniela Witten (2011). The joint graphical lasso for inverse covariance estimation across multiple classes. http://arxiv.org/abs/1111.0324

Examples

```
## load an example dataset with K=two classes, p=200 features, and n=100 samples per class:
data(example.data)
str(example.data)
## run fgl:
fgl.results = JGL(Y=example.data,penalty="fused",lambda1=.25,lambda2=.1)
## identify subnetworks
subnetworks(fgl.results$theta)
```

Index