Package 'cvxbiclustr'

June 28, 2015

Title Convex Biclustering Algorithm

Version 0.0.1

Author Eric C. Chi, Genevera I. Allen, Richard G. Baraniuk

Maintainer Eric C. Chi <ecchi1105@gmail.com>

Description An iterative algorithm for solving a convex formulation of the biclustering problem.

Depends R (>= 3.1.3), Matrix, igraph,

License MIT + file LICENSE

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-06-28 12:56:52

R topics documented:

`biclust_smooth` *Bicluster Smooth*

Description

`biclust_smooth` computes the bicluster estimates given row and column partitions

Usage

```
biclust_smooth(X, clusters_row, clusters_col)
```

Arguments

`X`	Data matrix
`clusters_row`	Row cluster object
`clusters_col`	Column cluster object

See Also

`get_subgroup_means_full`

`cobra` *Convex Biclustering Algorithm*

Description

`cobra` computes a convex biclustering path via Dykstra-like Proximal Algorithm

Usage

```
cobra(X, E_row, E_col, w_row, w_col, gamma, max_iter = 100, tol = 0.001)
```

Arguments

`X`	The data matrix to be clustered. The rows are the features, and the columns are the samples.
`E_row`	Edge-incidence matrix for row graph
`E_col`	Edge-incidence matrix for column graph
`w_row`	Vector of weights for row graph
`w_col`	Vector of weights for column graph
`gamma`	A sequence of regularization parameters for row and column shrinkage
`max_iter`	Maximum number of iterations
`tol`	Stopping criterion

Examples

```
## Create bicluster path
## Example: Lung
X <- lung
X <- X - mean(X)
X <- X/norm(X,'f')

## Create annotation for heatmap
types <- colnames(lung)
ty <- as.numeric(factor(types))
cols <- rainbow(4)
YlGnBu5 <- c('#ffffd9','#c7e9b4','#41b6c4','#225ea8','#081d58')
hmcols <- colorRampPalette(YlGnBu5)(256)

## Construct weights and edge-incidence matrices
phi <- 0.5; k <- 5
wts <- gkn_weights(X,phi=phi,k_row=k,k_col=k)
w_row <- wts$w_row
w_col <- wts$w_col
E_row <- wts$E_row
E_col <- wts$E_col

## Connected Components of Row and Column Graphs
wts$nRowComp
wts$nColComp

#### Initialize path parameters and structures
nGamma <- 5
gammaSeq <- 10**seq(0,3,length.out=nGamma)

## Generate solution path
sol <- cobra_validate(X,E_row,E_col,w_row,w_col,gammaSeq)

ix <- 4
heatmap(sol$U[[ix]],col=hmcols,labRow=NA,labCol=NA,ColSideCol=cols[ty])
```

`cobra_internal` *Convex Biclustering via Dykstra-like Proximal Algorithm*

Description

`cobra_internal` is an R wrapper around C code.

Usage

```
cobra_internal(X, Lambda_row, Lambda_col, E_row, E_col, w_row, w_col, gamma,
  max_iter = 100, tol = 0.001)
```

Arguments

X	Data matrix
Lambda_row	Initial guess of row Langrage multipliers
Lambda_col	Initial guess of column Langrage multipliers
E_row	Edge-incidence matrix for row graph
E_col	Edge-incidence matrix for column graph
w_row	Vector of weights for row graph
w_col	Vector of weights for column graph
gamma	Regularization parameter for shrinkage
max_iter	Maximum number of AMA iterations
tol	stopping tolerance

---

cobra_pod *MM algorithm for Convex Biclustering with Missing Data*

Description

cobra_pod performs convex biclustering on incomplete data matrices using an MM algorithm.

Usage

```
cobra_pod(X, Lambda_row, Lambda_col, E_row, E_col, w_row, w_col, Theta,
  max_iter = 100, tol = 0.001, max_iter_inner = 1000, tol_inner = 1e-04)
```

Arguments

X	The data matrix to be clustered. The rows are the features, and the columns are the samples.
Lambda_row	Initial guess of row Langrage multipliers
Lambda_col	Initial guess of column Langrage multipliers
E_row	Edge-incidence matrix for row graph
E_col	Edge-incidence matrix for column graph
w_row	Vector of weights for row graph
w_col	Vector of weights for column graph
Theta	A vector of missing indices - row major order
max_iter	Maximum number of iterations
tol	Stopping criterion
max_iter_inner	Maximum number of inner cobra iterations
tol_inner	Stopping criterion for inner cobra loop

Examples

```
## Create bicluster path
## Example: Lung
X <- lung
X <- X - mean(X)
X <- X/norm(X,'f')

## Create annotation for heatmap
types <- colnames(lung)
ty <- as.numeric(factor(types))
cols <- rainbow(4)
YlGnBu5 <- c('#ffffd9','#c7e9b4','#41b6c4','#225ea8','#081d58')
hmcols <- colorRampPalette(YlGnBu5)(256)

## Construct weights and edge-incidence matrices
phi <- 0.5; k <- 5
wts <- gkn_weights(X,phi=phi,k_row=k,k_col=k)
w_row <- wts$w_row
w_col <- wts$w_col
E_row <- wts$E_row
E_col <- wts$E_col

## Connected Components of Row and Column Graphs
wts$nRowComp
wts$nColComp

## Generate random initial dual variables
set.seed(12345)
n <- ncol(X); p <- nrow(X)
m_row <- nrow(E_row); m_col <- nrow(E_col)
Lambda_row <- matrix(rnorm(n*m_row),n,m_row)
Lambda_col <- matrix(rnorm(p*m_col),p,m_col)

#### Initialize path parameters and structures
gam <- 200

## Create random mask
nMissing <- floor(0.1*n*p)
Theta <- sample(1:(n*p), nMissing, replace=FALSE)

sol <- cobra_pod(X,Lambda_row,Lambda_col,E_row,E_col,gam*w_row,gam*w_col,Theta)

heatmap(sol$U,col=hmcols,labRow=NA,labCol=NA,ColSideCol=cols[ty])
```

`cobra_validate` *Perform validation to select regularization parameter*

Description

`cobra_validate` performs an MM algorithm wrapper to do parameter selection.

Usage

```
cobra_validate(X, E_row, E_col, w_row, w_col, gamma, Lambda_row = matrix(0, n,
  nrow(E_row)), Lambda_col = matrix(0, p, nrow(E_col)), fraction = 0.1,
  max_iter = 100, tol = 0.001, max_iter_inner = 1000, tol_inner = 1e-04)
```

Arguments

X	Data matrix
E_row	Edge-incidence matrix for row graph
E_col	Edge-incidence matrix for column graph
w_row	Vector of weights for row graph
w_col	Vector of weights for column graph
gamma	Regularization parameter for shrinkage
Lambda_row	Initial guess of row Langrage multipliers
Lambda_col	Initial guess of column Langrage multipliers
fraction	Fraction of entries for hold out
max_iter	Maximum number of iterations
tol	Stopping criterion
max_iter_inner	Maximum number of inner cobra iterations
tol_inner	Stopping criterion for inner cobra loop

Examples

```
## Create bicluster path
## Example: Lung
X <- lung
X <- X - mean(X)
X <- X/norm(X,'f')

## Create annotation for heatmap
types <- colnames(lung)
ty <- as.numeric(factor(types))
cols <- rainbow(4)
YlGnBu5 <- c('#ffffd9','#c7e9b4','#41b6c4','#225ea8','#081d58')
hmcols <- colorRampPalette(YlGnBu5)(256)

## Construct weights and edge-incidence matrices
phi <- 0.5; k <- 5
wts <- gkn_weights(X,phi=phi,k_row=k,k_col=k)
w_row <- wts$w_row
w_col <- wts$w_col
E_row <- wts$E_row
E_col <- wts$E_col

## Connected Components of Row and Column Graphs
wts$nRowComp
wts$nColComp
```

```
#### Initialize path parameters and structures
nGamma <- 7
gammaSeq <- 10**seq(0,1,length.out=nGamma)

## Generate solution path
sol <- cobra_validate(X,E_row,E_col,w_row,w_col,gammaSeq,fraction=0.01)

## Plot validation error
verr <- sol$validation_error
plot(verr)

## Heatmap of data smoothed at the model selected to minimize validation error
ix <- which.min(verr)
groups_row <- sol$groups_row[[ix]]
groups_col <- sol$groups_col[[ix]]
M <- biclust_smooth(X,groups_row,groups_col)
heatmap(M,col=hmcols,labRow=NA,labCol=NA,ColSideCol=cols[ty])
```

`compactify_edges` *Construct indices matrices*

Description

`compactify_edges` constructs M1, M2, and ix index matrices.

Usage

```
compactify_edges(w, n)
```

Arguments

`w`	weights vector
`n`	number of points to cluster

`convert` *Convert weights, as COO matrix, to CSC matrix and weights vector*

Description

`convert` Takes a weights triple (COO matrix) and converts it to a sparse edge-incidence matrix (CSC matrix) and weights vector.

Usage

```
convert(W)
```

Arguments

W	COO matrix of weights: (i,j,w[ij])

Examples

```
W <- matrix(0,3,3)
W[1,] <- c(1,2,1)
W[2,] <- c(1,3,2)
W[3,] <- c(2,3,3)

sol <- convert(W)
```

---

create_adjacency *Create adjacency matrix from V*

Description

create_adjacency creates an n-by-n sparse adjacency matrix from the matrix of centroid differences.

Usage

```
create_adjacency(V, Phi)
```

Arguments

V	Matrix of centroid differences
Phi	Edge-incidence matrix

---

create_edge_incidence *Edge-Incidence Matrix of Weights Graph*

Description

Construct the edge-incidence matrix of the weights graph.

Usage

```
create_edge_incidence(w, n)
```

Arguments

w	Weights vector
n	Number of points being clustered

find_clusters *Find clusters*

Description

find_clusters uses breadth-first search to identify the connected components of the corresponding adjacency graph of the centroid differences vectors.

Usage

```
find_clusters(A)
```

Arguments

A	adjacency matrix

get_subgroup_means *Get Subgroup Means*

Description

get_subgroup_means computes the subgroup means on the training set for validation.

Usage

```
get_subgroup_means(X, Theta, clusters_row, clusters_col)
```

Arguments

X	Data matrix
Theta	validation index set - vector
clusters_row	Row cluster object
clusters_col	Column cluster object

See Also

get_subgroup_means_full

get_subgroup_means_full *Get Subgroup Means*

Description

`get_subgroup_means_full` computes the subgroup means.

Usage

```
get_subgroup_means_full(X, clusters_row, clusters_col)
```

Arguments

`X`	Data matrix
`clusters_row`	Row cluster object
`clusters_col`	Column cluster object

See Also

`get_subgroup_means`

get_validation *Select Validation Set for a Matrix*

Description

`get_validation` selects a random sample of matrix indices for regularization parameter selection.

Usage

```
get_validation(p, n, fraction = 0.1, seed = 12345)
```

Arguments

`p`	Number of rows
`n`	Number of columns
`fraction`	Fraction of entries for hold out
`seed`	Seed for random number generator

Examples

```
n <- 5
p <- 4
fraction <- 0.1
Theta <- get_validation(p,n,fraction)

M <- matrix(rnorm(n*p),p,n)
YT <- t(M)
YT[Theta$ThetaV] <- NA
Y <- t(YT)
Theta
```

---

gkn_weights *Gaussian Kernel + k-Nearest Neighbor Weights*

Description

gkn_weights combines Gaussian kernel weights with k-nearest neighbor weights

Usage

```
gkn_weights(X, phi = 0.5, k_row = 5, k_col = 5)
```

Arguments

X	The data matrix to be clustered. The rows are the features, and the columns are the samples.
phi	The nonnegative parameter that controls the scale of kernel weights
k_row	The number of row nearest neighbors
k_col	The number of column nearest neighbors

---

kernel_weights *Compute Gaussian Kernel Weights*

Description

kernel_weights computes Gaussian kernel weights given a data matrix X and a scale parameter phi. Namely, the lth weight w[l] is given by

$$w[l] = exp(-phi||X[,i] - X[,j]||^2)$$

, where the lth pair of nodes is (i,j).

Usage

```
kernel_weights(X, phi = 1)
```

Arguments

X	The data matrix to be clustered. The rows are the features, and the columns are the samples.
phi	The nonnegative parameter that controls the scale of kernel weights

Value

A vector *w* of weights for convex clustering.

---

knn_weights *"Thin" a weight vector to be positive only for its k-nearest neighbors*

Description

knn_weights takes a weight vector w and sets the ith component w[i] to zero if either of the two corresponding nodes is not among the other's k nearest neighbors.

Usage

```
knn_weights(w, k, n)
```

Arguments

w	A vector of nonnegative weights. The ith entry w[i] denotes the weight used between the ith pair of centroids. The weights are in dictionary order.
k	The number of nearest neighbors
n	The number of data points.

Value

A vector *w* of weights for convex clustering.

---

lung *Lung cancer data*

Description

Lung cancer gene expression data set

Usage

```
data(lung)
```

Format

This data set contain 56 samples and gene expression values of 200 out of 12 625 genes measured using the Affymetrix 95av2 GeneChip. The subset of genes had the 200 highest sample variances among the originally measured genes. The samples comprise 20 pulmonary carcinoid samples (Carcinoid), 13 colon cancer metastasis samples (Colon), 17 normal lung samples (Normal) and 6 small cell lung carcinoma samples (SmallCell). The rownames are affymetrix gene ids.

Source

http://www.pnas.org/content/98/24/13790/suppl/DC1

References

Bhattacharjee, A., Richards, W. G., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., Loda, M., Weber, G., Mark, E. J., Lander, E. S., Wong, W., Johnson, B. E., Golub, T. R., Sugarbaker, D. J., and Meyerson, M. (2001). Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proceedings of the National Academy of Sciences of the United States of America.

---

weights_graph *Weights Graph Adjacency Matrix*

Description

Constructs the adjacency matrix of the weights graph. This is useful to determine the connectivity of the weights graph.

Usage

```
weights_graph(w, n)
```

Arguments

w	Weights vector
n	Number of points being clustered

Index