Package 'cvxclustr'

February 19, 2015

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

Author Eric C. Chi, Kenneth Lange

Version 1.1.1

License CC BY-NC-SA 4.0

Title Splitting methods for convex clustering

Description Alternating Minimization Algorithm (AMA) and Alternating Direction Method of Multipliers (ADMM) splitting methods for convex clustering.

Depends R (>= 2.15.2), Matrix, igraph

Suggests ggplot2

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-07-28 08:28:25

R topics documented:

AMA_step_size — *Compute step size Anderson-Morely upper bound on the largest eigenvalue of the Laplacian*

Description

AMA_step_size computes a step size based on the better of two bounds derived by Anderson and Morely.

Usage

```
AMA_step_size(w, n)
```

Arguments

w	vector of weights
n	number of points to cluster

Examples

```
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
w <- kernel_weights(X,phi)
w <- knn_weights(w,k,n)
AMA_step_size(w,n)
```

compactify_edges — *Construct indices matrices*

Description

compactify_edges constructs M1, M2, and ix index matrices. Note that storage conventions are different for ama and admm.

Usage

```
compactify_edges(w, n, method = "ama")
```

Arguments

w	weights vector
n	number of points to cluster
method	'ama' or 'admm'

`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, w, n, method = "ama")
```

Arguments

V	Matrix of centroid differences
w	Weights vector
n	Number of points to cluster
method	'ama' or 'admm'

Examples

```
## Clusterpaths for Mammal Dentition
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
w <- kernel_weights(X,phi)
w <- knn_weights(w,k,n)
gamma <- seq(0.0,43, length.out=100)

## Perform clustering
nu <- AMA_step_size(w,n)
sol <- cvxclust_path_ama(X,w,gamma,nu=nu)

## Construct adjacency matrix
A <- create_adjacency(sol$V[[10]],w,n)
G <- graph.adjacency(A, mode = 'upper')
plot(G,vertex.label=as.character(mammals[,1]),vertex.label.cex=0.65,vertex.label.font=2)
```

`create_clustering_problem` *Create a random clustering problem*

Description

`create_clustering_problem` makes a random clustering problem for testing purposes.

Usage

```
create_clustering_problem(p, n, seed = 12345, nnn = 3, method = "ama")
```

Arguments

`p`	Dimension of space of points to be clustered
`n`	Number of points
`seed`	Random number seed
`nnn`	Number of nearest neighbors
`method`	'ama' or 'admm'

Examples

```
p <- 10
n <- 20
seed <- 12345
rnd_problem_admm <- create_clustering_problem(p,n,seed)
```

`cvxclust` *Convex Clustering Path via Variable Splitting Methods*

Description

`cvxclust` estimates the convex clustering path via variable splitting methods: ADMM and AMA. This function is a wrapper function that calls either `cvxclust_path_admm` or `cvxclust_path_ama` (the default) to perform the computation. Required inputs include a data matrix `X` (rows are features; columns are samples), a vector of weights `w`, and a sequence of regularization parameters `gamma`. Two penalty norms are currently supported: 1-norm and 2-norm. Both ADMM and AMA admit acceleration schemes at little additional computation. Acceleration is turned on by default.

Usage

```
cvxclust(X, w, gamma, method = "ama", nu = 1, tol = 0.001,
  max_iter = 10000, type = 2, accelerate = TRUE)
```

Arguments

`X`	The data matrix to be clustered. The rows are the features, and the columns are the samples.
`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.
`method`	Algorithm to use: "admm" or "ama"
`gamma`	A sequence of regularization parameters.
`nu`	A positive penalty parameter for quadratic deviation term.
`tol`	The convergence tolerance.
`max_iter`	The maximum number of iterations.
`type`	An integer indicating the norm used: 1 = 1-norm, 2 = 2-norm.
`accelerate`	If `TRUE` (the default), acceleration is turned on.

Value

`U` A list of centroid matrices.

`V` A list of centroid difference matrices.

`Lambda` A list of Lagrange multiplier matrices.

Author(s)

Eric C. Chi, Kenneth Lange

See Also

`cvxclust_path_ama` and `cvxclust_path_admm` for estimating the clustering path with AMA or ADMM. `kernel_weights` and `knn_weights` compute useful weights. To extract cluster assignments from the clustering path use `create_adjacency` and `find_clusters`.

Examples

```
## Clusterpaths for Mammal Dentition
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
w <- kernel_weights(X,phi)
w <- knn_weights(w,k,n)
gamma <- seq(0.0,43, length.out=100)

## Perform clustering
sol <- cvxclust(X,w,gamma)
```

```
## Plot the cluster path
library(ggplot2)
svdX <- svd(X)
pc <- svdX$u[,1:2,drop=FALSE]
pc.df <- as.data.frame(t(pc)%*%X)
nGamma <- sol$nGamma
df.paths <- data.frame(x=c(),y=c(), group=c())
for (j in 1:nGamma) {
  pcs <- t(pc)%*%sol$U[[j]]
  x <- pcs[1,]
  y <- pcs[2,]
  df <- data.frame(x=pcs[1,], y=pcs[2,], group=1:n)
  df.paths <- rbind(df.paths,df)
}
X_data <- as.data.frame(t(X)%*%pc)
colnames(X_data) <- c("x","y")
X_data$Name <- mammals[,1]
data_plot <- ggplot(data=df.paths,aes(x=x,y=y))
data_plot <- data_plot + geom_path(aes(group=group),colour='grey30',alpha=0.5)
data_plot <- data_plot + geom_text(data=X_data,aes(x=x,y=y,label=Name),
  position=position_jitter(h=0.125,w=0.125))
data_plot <- data_plot + geom_point(data=X_data,aes(x=x,y=y),size=1.5)
data_plot <- data_plot + xlab('Principal Component 1') + ylab('Principal Component 2')
data_plot + theme_bw()

## Output Cluster Assignment at 10th gamma
A <- create_adjacency(sol$V[[10]],w,n)
find_clusters(A)

## Visualize Cluster Assignment
G <- graph.adjacency(A, mode = 'upper')
plot(G,vertex.label=as.character(mammals[,1]),vertex.label.cex=0.65,vertex.label.font=2)
```

`cvxclustr` *Convex Clustering via Splitting Methods*

Description

Clustering is a fundamental problem in science and engineering. Many classic methods such as k-means, Gaussian mixture models, and hierarchical clustering, however, employ greedy algorithms which can be entrapped in local minima, sometimes drastical suboptimal ones at that. Recently introduced convex relaxations of k-means and hierarchical clustering shrink cluster centroids toward one another and ensure a unique global minimizer. This package provides two variable splitting methods

- Alternating Method of Multipliers (ADMM)
- Alternating Minimization Algorithm (AMA)

for solving this convex formulation of the clustering problem. We seek the centroids u_i that minimize

$$\frac{1}{2}\sum_i ||x_i - u_i||_2^2 + \gamma \sum_l w_l ||u_{l1} - u_{l2}||$$

Two penalty norms are currently supported: 1-norm and 2-norm.

Details

The two main functions are `cvxclust_path_admm` and `cvxclust_path_ama` which compute the cluster paths using the ADMM and AMA methods respectively. The function `cvxclust` is a wrapper function that calls either `cvxclust_path_admm` or `cvxclust_path_ama` (the default) to perform the computation.

The functions `kernel_weights` and `knn_weights` can be used in sequence to compute weights that can improve the quality of the clustering paths.

The typical usage consists of three steps:

- Compute weights `w`.
- Generate a geometrically increasing regularization parameter sequence. Unfortunately a closed form expression for the minimum amount of penalization to get complete coalescence is currently unknown.
- Call `cvxclust` using the data `X`, weights `w`, and regularization parameter sequence `gamma`.

Cluster assignments can also be retrieved from the solution to the convex clustering problem. Both `cvxclust_path_admm` and `cvxclust_path_ama` output an object of class `cvxclustobject`. A cluster assignment can be extracted in two steps:

- Call `create_adjacency` to construct an adjacency matrix from the centroid differences variable `V`.
- Call `find_clusters` to extract the connected components of the adjacency matrix.

Author(s)

Eric C. Chi, Kenneth Lange

References

Eric C. Chi and Kenneth Lange. Splitting Methods for Convex Clustering. Journal of Computational and Graphical Statistics, in press. http://arxiv.org/abs/1304.0499.

cvxclust_admm *Convex clustering via ADMM*

Description

cvxclust_admm performs convex clustering via ADMM. This is an R wrapper function around C code. Dimensions of various arguments are as follows:

- n is the number of data points
- p is the number of features
- k is the number non-zero weights.

Note that the indices matrices 'M1', 'M2', and 'ix' take on values starting at 0 to match the indexing conventions of C.

Usage

```
cvxclust_admm(X, Lambda, ix, M1, M2, s1, s2, w, gamma, nu, max_iter = 100,
  type = 2, tol_abs = 1e-05, tol_rel = 1e-04, accelerate = TRUE)
```

Arguments

X	The p-by-n data matrix whose columns are to be clustered.
Lambda	The p-by-k matrix of Lagrange multipliers.
ix	The k-by-2 matrix of index pairs.
M1	Index set used to track nonzero weights.
M2	Index set used to track nonzero weights.
s1	Index set used to track nonzero weights.
s2	Index set used to track nonzero weights.
w	A vector of k positive weights.
gamma	The regularization parameter controlling the amount of shrinkage.
nu	Augmented Lagrangian penalty parameter
max_iter	The maximum number of iterations.
type	An integer indicating the norm used: 1 = 1-norm, 2 = 2-norm.
tol_abs	The convergence tolerance (absolute).
tol_rel	The convergence tolerance (relative).
accelerate	If TRUE (the default), acceleration is turned on.

Value

`U` A list of centroid matrices.

`V` A list of centroid difference matrices.

`Lambda` A list of Lagrange multiplier matrices.

`nu` The final step size used.

`primal` The primal residuals.

`dual` The dual residuals.

`tol_primal` The primal residual tolerances.

`tol_dual` The dual residual tolerances.

`iter` The number of iterations taken.

Author(s)

Eric C. Chi, Kenneth Lange

Examples

```
## Create random problems
p <- 10
n <- 20
seed <- 12345
nProbs <- 10
errors <- double(nProbs)
for (i in 1:nProbs) {
  seed <- seed + sample(1:1e2,1)
  rnd_problem <- create_clustering_problem(p,n,seed=seed,method='admm')
  X <- rnd_problem$X
  ix <- rnd_problem$ix
  M1 <- rnd_problem$M1
  M2 <- rnd_problem$M2
  s1 <- rnd_problem$s1
  s2 <- rnd_problem$s2
  w  <- rnd_problem$w
  nK <- length(w)
  Lambda <- matrix(rnorm(p*nK),p,nK)
  gamma <- 0.1
  nu <- 1
  max_iter <- 1e6
  tol_abs <- 1e-15
  tol_rel <- 1e-15
  sol_admm_acc <- cvxclust_admm(X,Lambda,ix,M1,M2,s1,s2,w,gamma,nu,max_iter=max_iter,
    tol_abs=tol_abs,tol_rel=tol_rel,accelerate=TRUE)
  sol_admm <- cvxclust_admm(X,Lambda,ix,M1,M2,s1,s2,w,gamma,nu,max_iter=max_iter,
    tol_abs=tol_abs,tol_rel=tol_rel,accelerate=FALSE)
  errors[i] <- norm(as.matrix(sol_admm_acc$U-sol_admm$U),'i')
}
```

cvxclust_ama *Convex clustering via AMA*

Description

cvxclust_ama performs convex clustering via AMA. This is an R wrapper function around C code. Dimensions of various arguments are as follows:

- n is the number of data points
- p is the number of features
- k is the number non-zero weights.

Note that the indices matrices 'M1', 'M2', and 'ix' take on values starting at 0 to match the indexing conventions of C.

Usage

```
cvxclust_ama(X, Lambda, ix, M1, M2, s1, s2, w, gamma, nu, type = 2,
  max_iter = 100, tol = 1e-04, accelerate = TRUE)
```

Arguments

X	The p-by-n data matrix whose columns are to be clustered.
Lambda	The p-by-k matrix of Lagrange multipliers.
ix	The k-by-2 matrix of index pairs.
M1	Index set used to track nonzero weights.
M2	Index set used to track nonzero weights.
s1	Index set used to track nonzero weights.
s2	Index set used to track nonzero weights.
w	A vector of k positive weights.
gamma	The regularization parameter controlling the amount of shrinkage.
nu	The initial step size parameter when backtracking is applied. Otherwise it is a fixed step size in which case there are no guarantees of convergence if it exceeds 2/ncol(X).
type	An integer indicating the norm used: 1 = 1-norm, 2 = 2-norm.
max_iter	The maximum number of iterations.
tol	The convergence tolerance.
accelerate	If TRUE (the default), acceleration is turned on.

Value

U A list of centroid matrices.

V A list of centroid difference matrices.

Lambda A list of Lagrange multiplier matrices.

nu The final step size used.

primal The primal objective evaluated at the final iterate.

dual The dual objective evaluated at the final iterate.

iter The number of iterations taken.

Author(s)

Eric C. Chi, Kenneth Lange

Examples

```
## Create random problem
seed <- 12345
p <- 10
n <- 20
rnd_problem <- create_clustering_problem(p,n,seed)
X <- rnd_problem$X
ix <- rnd_problem$ix
M1 <- rnd_problem$M1
M2 <- rnd_problem$M2
s1 <- rnd_problem$s1
s2 <- rnd_problem$s2
w  <- rnd_problem$w
nK <- nrow(ix)
Lambda <- matrix(rnorm(p*nK),p,nK)
gamma <- 0.1
nu <- 1.999/n
max_iter <- 1e6
tol <- 1e-15
sol_ama <- cvxclust_ama(X,Lambda,ix,M1,M2,s1,s2,w,gamma,nu,max_iter=max_iter,tol=tol)
```

---

cvxclust_path_admm *Convex Clustering Path via ADMM*

Description

cvxclust_path_admm estimates the convex clustering path via ADMM. Required inputs include a data matrix X (rows are features; columns are samples), a vector of weights w, and a sequence of regularization parameters gamma. Two penalty norms are currently supported: 1-norm and 2-norm. ADMM admits acceleration by extrapolated steps akin to those in FISTA. This speed-up is employed by default.

Usage

```
cvxclust_path_admm(X, w, gamma, nu = 1, tol_abs = 1e-05, tol_rel = 1e-04,
  max_iter = 10000, type = 2, accelerate = TRUE)
```

Arguments

`X`	The data matrix to be clustered. The rows are the features, and the columns are the samples.
`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.
`gamma`	A sequence of regularization parameters.
`nu`	A positive penalty parameter for quadratic deviation term.
`tol_abs`	The convergence tolerance (absolute).
`tol_rel`	The convergence tolerance (relative).
`max_iter`	The maximum number of iterations.
`type`	An integer indicating the norm used: 1 = 1-norm, 2 = 2-norm.
`accelerate`	If `TRUE` (the default), acceleration is turned on.

Value

`U` A list of centroid matrices.

`V` A list of centroid difference matrices.

`Lambda` A list of Lagrange multiplier matrices.

Author(s)

Eric C. Chi, Kenneth Lange

See Also

`cvxclust_path_ama` for estimating the clustering path with AMA. `kernel_weights` and `knn_weights` compute useful weights. To extract cluster assignments from the clustering path use `create_adjacency` and `find_clusters`.

Examples

```
## Clusterpaths for Mammal Dentition
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
w <- kernel_weights(X,phi)
w <- knn_weights(w,k,n)
```

```
gamma <- seq(0.0,43, length.out=100)

## Perform clustering
sol <- cvxclust_path_admm(X,w,gamma)

## Plot the cluster path
library(ggplot2)
svdX <- svd(X)
pc <- svdX$u[,1:2,drop=FALSE]
pc.df <- as.data.frame(t(pc)%*%X)
nGamma <- sol$nGamma
df.paths <- data.frame(x=c(),y=c(), group=c())
for (j in 1:nGamma) {
  pcs <- t(pc)%*%sol$U[[j]]
  x <- pcs[1,]
  y <- pcs[2,]
  df <- data.frame(x=pcs[1,], y=pcs[2,], group=1:n)
  df.paths <- rbind(df.paths,df)
}
X_data <- as.data.frame(t(X)%*%pc)
colnames(X_data) <- c("x","y")
X_data$Name <- mammals[,1]
data_plot <- ggplot(data=df.paths,aes(x=x,y=y))
data_plot <- data_plot + geom_path(aes(group=group),colour='grey30',alpha=0.5)
data_plot <- data_plot + geom_text(data=X_data,aes(x=x,y=y,label=Name),
  position=position_jitter(h=0.125,w=0.125))
data_plot <- data_plot + geom_point(data=X_data,aes(x=x,y=y),size=1.5)
data_plot <- data_plot + xlab('Principal Component 1') + ylab('Principal Component 2')
data_plot + theme_bw()

## Output Cluster Assignment at 10th gamma
A <- create_adjacency(sol$V[[10]],w,n,method='admm')
find_clusters(A)

## Visualize Cluster Assignment
G <- graph.adjacency(A, mode = 'upper')
plot(G,vertex.label=as.character(mammals[,1]),vertex.label.cex=0.65,vertex.label.font=2)
```

cvxclust_path_ama *Convex Clustering Path via AMA*

Description

cvxclust_path_ama estimates the convex clustering path via the Alternating Minimization Algorithm. Required inputs include a data matrix X (rows are features; columns are samples), a vector of weights w, and a sequence of regularization parameters gamma. Two penalty norms are currently supported: 1-norm and 2-norm. AMA is performing proximal gradient ascent on the dual function, and therefore can be accelerated with FISTA. This speed-up is employed by default.

Usage

```
cvxclust_path_ama(X, w, gamma, nu = 1, tol = 0.001, max_iter = 10000,
  type = 2, accelerate = TRUE)
```

Arguments

`X`	The data matrix to be clustered. The rows are the features, and the columns are the samples.
`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.
`gamma`	A sequence of regularization parameters.
`nu`	The initial step size parameter when backtracking is applied. Otherwise it is a fixed step size in which case there are no guarantees of convergence if it exceeds `2/ncol(X)`.
`tol`	The convergence tolerance.
`max_iter`	The maximum number of iterations.
`type`	An integer indicating the norm used: 1 = 1-norm, 2 = 2-norm.
`accelerate`	If `TRUE` (the default), acceleration is turned on.

Value

`U` A list of centroid matrices.

`V` A list of centroid difference matrices.

`Lambda` A list of Lagrange multiplier matrices.

Author(s)

Eric C. Chi, Kenneth Lange

See Also

`cvxclust_path_admm` for estimating the clustering path with ADMM. `kernel_weights` and `knn_weights` compute useful weights. To extract cluster assignments from the clustering path use `create_adjacency` and `find_clusters`.

Examples

```
## Clusterpaths for Mammal Dentition
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
w <- kernel_weights(X,phi)
```

```
w <- knn_weights(w,k,n)
gamma <- seq(0.0,43, length.out=100)

## Perform clustering
nu <- AMA_step_size(w,n)
sol <- cvxclust_path_ama(X,w,gamma,nu=nu)

## Plot the cluster path
library(ggplot2)
svdX <- svd(X)
pc <- svdX$u[,1:2,drop=FALSE]
pc.df <- as.data.frame(t(pc)%*%X)
nGamma <- sol$nGamma
df.paths <- data.frame(x=c(),y=c(), group=c())
for (j in 1:nGamma) {
  pcs <- t(pc)%*%sol$U[[j]]
  x <- pcs[1,]
  y <- pcs[2,]
  df <- data.frame(x=pcs[1,], y=pcs[2,], group=1:n)
  df.paths <- rbind(df.paths,df)
}
X_data <- as.data.frame(t(X)%*%pc)
colnames(X_data) <- c("x","y")
X_data$Name <- mammals[,1]
data_plot <- ggplot(data=df.paths,aes(x=x,y=y))
data_plot <- data_plot + geom_path(aes(group=group),colour='grey30',alpha=0.5)
data_plot <- data_plot + geom_text(data=X_data,aes(x=x,y=y,label=Name),
  position=position_jitter(h=0.125,w=0.125))
data_plot <- data_plot + geom_point(data=X_data,aes(x=x,y=y),size=1.5)
data_plot <- data_plot + xlab('Principal Component 1') + ylab('Principal Component 2')
data_plot + theme_bw()

## Output Cluster Assignment at 10th gamma
A <- create_adjacency(sol$V[[10]],w,n)
find_clusters(A)

## Visualize Cluster Assignment
G <- graph.adjacency(A, mode = 'upper')
plot(G,vertex.label=as.character(mammals[,1]),vertex.label.cex=0.65,vertex.label.font=2)
```

`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

Examples

```
## Clusterpaths for Mammal Dentition
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
w <- kernel_weights(X,phi)
w <- knn_weights(w,k,n)
gamma <- seq(0.0,43, length.out=100)

## Perform clustering
nu <- AMA_step_size(w,n)
sol <- cvxclust_path_ama(X,w,gamma,nu=nu)

## Construct adjacency matrix
A <- create_adjacency(sol$V[[10]],w,n)
find_clusters(A)
```

`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.

Author(s)

Eric C. Chi, Kenneth Lange

`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.

Author(s)

Eric C. Chi, Kenneth Lange

`mammals` *Tally of types of teeth in some mammals.*

Description

A dataset containing how many of eight kinds of teeth various mammals have. This data set is a subset of the mammals dentition data in the homals package.

Format

A data frame with 27 rows and 8 variables

References

Jan de Leeuw, Patrick Mair (2009). Gifi Methods for Optimal Scaling in R: The Package homals. Journal of Statistical Software, August 2009, Volume 31, Issue 4. `http://www.jstatsoft.org/v31/i04/paper`.

`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

Examples

```
## Clusterpaths for Mammal Dentition
data(mammals)
X <- as.matrix(mammals[,-1])
X <- t(scale(X,center=TRUE,scale=FALSE))
n <- ncol(X)

## Pick some weights and a sequence of regularization parameters.
k <- 5
phi <- 0.5
```

```
w <- kernel_weights(X,phi)
w <- knn_weights(w,k,n)

A <- weights_graph(w,n)
find_clusters(A)

## Visualize Cluster Assignment
G <- graph.adjacency(A, mode = 'upper')
plot(G,vertex.label=as.character(mammals[,1]),vertex.label.cex=0.65,vertex.label.font=2)
```

Index