

Package ‘gmum.r’

October 4, 2015

Version 0.2.1

Date 2015-09-25

Title GMUM Machine Learning Group Package

Type Package

Author Wojciech Czarnecki, Stanislaw Jastrzebski, Marcin Data, Igor Sieradzki, Mateusz Bruno-Kaminski, Karol Jurek, Piotr Kowenzowski, Michal Pletty, Konrad Talik, Maciej Zgliczynski

Maintainer Stanislaw Jastrzebski <staszek.jastrzebski@gmail.com>

Description Direct R interface to Support Vector Machine libraries ('LIBSVM' and 'SVM-Light') and efficient C++ implementations of Growing Neural Gas and models developed by 'GMUM' group (Cross Entropy Clustering and 2eSVM).

License MIT + file LICENSE

Repository CRAN

Imports ggplot2 (>= 1.0.0), stats, igraph, SparseM, Matrix, httr, Rcpp (>= 0.11.6), methods, grid

LinkingTo Rcpp, RcppArmadillo, BH

NeedsCompilation yes

LazyData yes

Suggests car, caret, e1071, klaR, testthat, mlbench

URL <https://github.com/gmum/gmum.r>

BugReports <https://github.com/gmum/gmum.r/issues>

Date/Publication 2015-10-04 23:41:44

R topics documented:

calculateCentroids	3
caret.gmumSvmLinear	4
caret.gmumSvmPoly	4
caret.gmumSvmRadial	5
CEC	6
cec.ellipsegauss	7

cec.ellipsegauss.extra	8
cec.mouse1.spherical	8
cec.mouse1.spherical.extra	8
centers	8
clustering	9
convertToIGraph	9
covMatrix	10
energy	10
errorStatistics	11
findClosests	11
get.wine.dataset.X	12
get.wine.dataset.y	12
getDataset	13
GNG	13
gng.plot.layout.v2d	15
gng.preset.cube	15
gng.preset.plane	16
gng.preset.sphere	16
gngLoad	17
gngSave	17
insertExamples	18
isRunning	18
iterations	19
logClusters	19
logEnergy	20
meanError	20
MultiClassSVM-class	21
node	21
numberNodes	22
OptimizedGNG	22
pause	23
plot.Rcpp_CecModel	24
plot.Rcpp_GNGServer	25
plot.Rcpp_SVMClient	26
predict	27
predict.gng	27
predict.Rcpp_SVMClient	28
predictComponent	28
print.Rcpp_CecModel	29
print.Rcpp_GNGServer	30
print.Rcpp_SVMClient	30
Rcpp_CecConfiguration-class	30
Rcpp_CecModel-class	31
Rcpp_GNGServer-class	31
Rcpp_SVMClient-class	31
run	32
runAll	32
runOneIteration	33

calculateCentroids 3

summary.Rcpp_CecModel	33
summary.Rcpp_GNGServer	34
summary.Rcpp_SVMClient	34
SVM	35
svm.accuracy	37
svm.breastcancer.dataset	38
svm.transduction	38
terminate	38
Tset	38

Index 39

`calculateCentroids` *calculateCentroids*

Description

Using passed `community.detection` finds communities and for each community pick node with biggest betweenness score

Usage

```
calculateCentroids(object,  
  community.detection.algorithm = spinglass.community)
```

Arguments

`object` GNG object
`community.detection.algorithm`
 Used algorithm from `igraph` package, by default `spinglass.community`

Details

Get centroids

Examples

```
## Not run:  
gng <- GNG(gng.preset.sphere(100))  
print(node(gng, calculateCentroids(gng)[1])$pos)  
  
## End(Not run)
```

caret.gmumSvmLinear *Caret model representation for SVM with linear kernel*

Description

Supply as parameter "method" in the caret::train function

Usage

```
caret.gmumSvmLinear
```

Format

List of caret specific values

Examples

```
## Not run:
model <- train(Class ~ ., data = training,
method = caret.gmumSvmLinear,
preProc = c("center", "scale"),
tuneLength = 8,
trControl = fitControl,
tuneGrid = expand.grid(C=10^(c(-4:4)), gamma=10^(c(-4:4))),
core = "libsvm", # gmum.R parameter - pick library
verbosity = 0 # no outputs
)

## End(Not run)
```

caret.gmumSvmPoly *Caret model representation for SVM with linear kernel*

Description

Supply as parameter "method" in the caret::poly function

Usage

```
caret.gmumSvmPoly
```

Format

List of caret specific values

Examples

```
## Not run:
model <- train(Class ~ ., data = training,
method = caret.gmumSvmPoly,
preProc = c("center", "scale"),
tuneLength = 8,
trControl = fitControl,
tuneGrid = expand.grid(C=10^(c(-4:4)), gamma=10^(c(-4:4))),
core = "libsvm", # gmum.R parameter - pick library
verbosity = 0 # no outputs
)

## End(Not run)
```

caret.gmumSvmRadial *Caret model representation for SVM with radial kernel*

Description

Supply as parameter "method" in the caret::train function

Usage

```
caret.gmumSvmRadial
```

Format

List of caret specific values

Examples

```
## Not run:
model <- train(Class ~ ., data = training,
method = caret.gmumSvmRadial,
preProc = c("center", "scale"),
tuneLength = 8,
trControl = fitControl,
tuneGrid = expand.grid(C=10^(c(-4:4)), gamma=10^(c(-4:4))),
core = "libsvm", # gmum.R parameter - pick library
verbosity = 0 # no outputs
)

## End(Not run)
```

CEC

*Cross-Entropy Clustering***Description**

Create CEC model object

Usage

```
CEC(x = NULL, k = 0, method.type = "standard", method.init = "kmeans++",
    params.r = 0, params.cov = matrix(0), params.centroids = NULL,
    params.mix = NULL, params.function = NULL, control.nstart = 10,
    control.eps = 0.05, control.itmax = 25, log.energy = TRUE,
    log.ncluster = TRUE, seed = NULL)
```

Arguments

x	Numeric matrix of data.
k	Initial number of clusters.
method.type	Type of clustering (Gaussian family). <ol style="list-style-type: none"> 'diagonal' Gaussians with diagonal covariance. The clustering will try to divide the data into ellipsoid with radiuses parallel to coordinate axes 'fixed_spherical' Spherical (radial) Gaussian densities (additional parameter - radius) 'fixed_covariance' The clustering will have the tendency to divide the data into clusters resembling the unit circles in the Mahalanobis distance (additional parameter - covaraince matrix required) 'func' Own function dependent on m and sigma (additional parameter) 'standard' We divide dataset into ellipsoid-like clusters without any preferences (default) 'spherical' The clustering will try to divide the data into circles of arbitrary sizes
method.init	Method to initialize clusters. <ol style="list-style-type: none"> 'centroids' 'kmeans++' 'random'
params.r	Radius for spherical family.
params.cov	Covariance matrix for covariance family.
params.centroids	List of centroids.
params.mix	List of cluster with mixed Gaussian types.
params.function	User energy function

control.nstart How many times to perform algorithm.
 control.eps What change of value should terminate algorithm.
 control.itmax Maximum number of iterations at each start.
 log.energy Records collected energy of all clusters in each iteration.
 log.ncluster Records number of clusters in each iteration.
 seed User seed

Examples

```
## Not run:
CEC(k=3, x=dataset)

CEC(k=3, x=dataset, control.nstart=10, method.type='spherical', control.eps=0.05)

CEC(k=2, x=dataset, method.type='spherical', method.init='centroids',
    params.centroids=list(c(-0.5,0.5),c(0,0)))

CEC(k=5, x=dataset, method.type='fixed_spherical', params.r=0.01,
    control.nstart=10, control.eps=0.07)

CEC(k=5, x=dataset, method.type='fixed_covariance',
    params.cov=matrix(c(0.03,0,0,0.01),2), control.nstart=10, control.eps=0.06)

CEC(k=1, x=dataset, method.type='func',
    params.function='name_of_my_own_function')

fixed_spherical_cluster_param = list(method.type = 'fixed_spherical', params.r = 0.001),
covariance_cluster_param = list(method.type = 'fixed_covariance',
    params.cov=matrix(c(0.05, 0, 0, 0.001), 2))
CEC(x = dataset, params.mix = list(covariance_cluster_param,
    fixed_spherical_cluster_param, fixed_spherical_cluster_param,
    fixed_spherical_cluster_param, fixed_spherical_cluster_param), control.nstart = 10)

p1 = list(method.type='spherical', k=3)
p2 = list(method.type='diagonal', k=2)
CEC(x=dataset, params.mix=list(p1, p2))

## End(Not run)
```

 cec.ellipsegauss

 cec.ellipsegauss

Description

Simple dataset consisting in data drawn from set of elliptical gaussians

cec.ellipsegauss.extra
cec.ellipsegauss.extra

Description

Extra information for dataset cec.ellipsegauss (energy and cluster assignment)

cec.mouse1.spherical *cec.mouse1.spherical*

Description

Simple mouse-shaped dataset

cec.mouse1.spherical.extra
cec.mouse1.spherical.extra

Description

Extra information for dataset cec.mouse1.extra (energy and cluster assignment)

centers *centers*

Description

Print centers of clusters

Usage

```
centers(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:  
centers(c)  
  
## End(Not run)
```

clustering	<i>clustering</i>
------------	-------------------

Description

Print labels assigned

Print labels assigned

Gets vector with node indexes assigned to examples in the dataset

Usage

```
clustering(c)
```

```
## S3 method for class 'Rcpp_CecModel'
```

```
clustering(c)
```

```
## S3 method for class 'Rcpp_GNGServer'
```

```
clustering(c)
```

Arguments

c Object with clusters

Examples

```
## Not run:  
clustering(c)
```

```
## End(Not run)
```

```
## Not run:  
clustering(c)
```

```
## End(Not run)
```

```
gng <- GNG(gng.preset.sphere(100))
```

```
clustering(gng)
```

convertToIGraph	<i>convertToIGraph</i>
-----------------	------------------------

Description

Converts GNG to igraph object, where every vertex contains attributes `gng.index`, `error`, `data.label` and 3 first spatial coordinates (as attributes `v0`, `v1`, `v2`). Additionally utility attribute is present if utility GNG is used.

Usage

```
convertToIGraph(object, calculate.dist = TRUE)
```

Arguments

object GNG object
 calculate.dist If true will calculate all n^2 distances in the graph

covMatrix	<i>covMatrix</i>
-----------	------------------

Description

Print covariances of clusters

Usage

```
covMatrix(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:
covMatrix(c)

## End(Not run)
```

energy	<i>energy</i>
--------	---------------

Description

Print result energy of clustering found

Usage

```
energy(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:
energy(c)

## End(Not run)
```

errorStatistics	<i>errorStatistics</i>
-----------------	------------------------

Description

Gets vector with errors for every second of execution

Usage

```
errorStatistics(object)
```

Arguments

object	GNG object
--------	------------

Examples

```
gng <- GNG(gng.preset.sphere(100))
errorStatistics(gng)
```

findClosests	<i>findClosests</i>
--------------	---------------------

Description

Finds closest node from given list to vector. Often used together with calculateCentroids

Usage

```
findClosests(object, node.ids, x)
```

Arguments

object	GNG object
node.ids	List of indexes of nodes in gng.
x	Can be either vector or data.frame.

Details

Find closest node

Examples

```
## Not run:  
gng <- GNG(gng.preset.sphere(100))  
# Find closest centroid to c(1,1,1)  
found.centroids <- calculateCentroids(gng)  
findClosests(gng, found.centroids, c(1,1,1))  
  
## End(Not run)
```

`get.wine.dataset.X` *get.wine.dataset.X*

Description

Retrieves wine dataset design matrix from UCI repository

Usage

```
get.wine.dataset.X(scale = TRUE)
```

Arguments

`scale` if TRUE will perform feature scaling

`get.wine.dataset.y` *get.wine.dataset.y*

Description

Retrieves wine dataset labels from UCI repository

Usage

```
get.wine.dataset.y()
```

getDataset	<i>getDataset</i>
------------	-------------------

Description

Print input dataset

Usage

```
getDataset(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:
getDataset(c)

## End(Not run)
```

GNG	<i>Constructor of GrowingNeuralGas object.</i>
-----	--

Description

Construct GNG object. Can be used to train offline, or online.

Usage

```
GNG(x = NULL, labels = c(), beta = 0.99, alpha = 0.5,
    max.nodes = 1000, eps.n = 6e-04, eps.w = 0.05, max.edge.age = 200,
    train.online = FALSE, max.iter = 200, dim = -1,
    min.improvement = 0.001, lambda = 200, verbosity = 0, seed = -1,
    k = NULL)
```

Arguments

x Passed data (matrix of data.frame) for offline training

labels Every example can be associated with labels that are added to nodes later. By default empty

beta Decrease the error variables of all node nodes by this fraction (forgetting rate). Default 0.99

alpha	Decrease the error variables of the nodes neighboring to the newly inserted node by this fraction. Default 0.5
max.nodes	Maximum number of nodes (after reaching this size it will continue running, but won't add new nodes)
eps.n	How strongly adapt neighbour node. Default 0.0006
eps.w	How strongly adapt winning node. Default 0.05
max.edge.age	Maximum edge age. Decrease to increase speed of change of graph topology. Default 200
train.online	default FALSE. If used will run in online fashion
max.iter	Uf training offline will stop if exceeds max.iter iterations. Default 200
dim	Used for training online, specifies training example size
min.improvement	Used for offline (default) training. Controls stopping criterion, decrease if training stops too early. Default 1e-3
lambda	Every lambda iteration is added new vertex. Default 200
verbosity	How verbose should the process be, as integer from [0, 6], default: 0
seed	Seed for internal randomization
k	Utility constant, by default turned off. Good value is 1.3. Constant controlling speed of erasing obsolete nodes, see http://sund.de/netze/applets/gng/full/tex/DemoGNG/node20.html

Examples

```
## Not run:
X <- gng.preset.sphere(100)
y <- round(runif(100))
# Train in an offline manner
gng <- GNG(X, labels=y, max.nodes=20)
# Plot
plot(gng)

# Train in an online manner with utility (erasing obsolete nodes)
gng <- GNG(max.nodes=20, train.online=TRUE, k=1.3, dim=3)
insertExamples(gng, X, labels=y)
run(gng)
Sys.sleep(10)
terminate(gng)
# Plot
plot(gng)

## End(Not run)
```

`gng.plot.layout.v2d` *Use first two spatial coordinates as position in layout*

Description

Use first two spatial coordinates as position in layout

Usage

```
gng.plot.layout.v2d(g)
```

Arguments

`g` GNG object

Note

You can pass any igraph layout algorithm to plot

`gng.preset.cube` *gng.preset.cube*

Description

Generate sample cube dataset

Usage

```
gng.preset.cube(N, r = 0.5, center = c(0.5, 0.5, 0.5))
```

Arguments

`N` Number of points
`r` Length of the side of cube
`center` Center of the plane

Details

Cube preset dataset

Examples

```
X <- gng.preset.cube(100)  
gng <- GNG(X)
```

`gng.preset.plane` *gng.preset.plane*

Description

Generate sample plane dataset

Usage

```
gng.preset.plane(N, side = 0.5, center = c(0.5, 0.5, 0.5))
```

Arguments

N	Number of points
side	Length of the side of plane
center	Center of the plane

Details

Plane preset dataset

Examples

```
X <- gng.preset.plane(100)
gng <- GNG(X)
```

`gng.preset.sphere` *gng.preset.sphere*

Description

Generate sample sphere dataset

Usage

```
gng.preset.sphere(N, r = 0.5, center = c(0.5, 0.5, 0.5))
```

Arguments

N	Number of points
r	Radius of the sphere
center	Center of the sphere

Details

Sphere preset dataset

Examples

```
X <- gng.preset.sphere(100)
gng <- GNG(X)
```

<code>gngLoad</code>	<i>gngLoad</i>
----------------------	----------------

Description

Writes model to a disk space efficient binary format.

Usage

```
gngLoad(filename)
```

Arguments

filename	Binary file location
----------	----------------------

Details

Load model from binary format

<code>gngSave</code>	<i>gngSave</i>
----------------------	----------------

Description

Writes model to a disk space efficient binary format.

Usage

```
gngSave(object, filename)
```

Arguments

object	GNG object
filename	File where binary will be saved

Details

Save model to binary format

insertExamples	<i>insertExamples</i>
----------------	-----------------------

Description

Insert examples with optional labels.

Usage

```
insertExamples(object, examples, labels = c())
```

Arguments

object	GNG object
examples	matrix or data.frame with rows as examples. Note: if training online make sure number of columns matches dim parameter passed to GNG constructor.
labels	vector of labels, that will be associated with nodes in the graph. GNG will assign to each node a mean of labels of closest examples.

Note

It copies your examples twice in RAM. You might want to use `object$.insertExamples`.

Examples

```
X <- gng.preset.sphere(100)
gng <- GNG(X, train.online=TRUE)
# Add more examples
X = gng.preset.sphere(100)
insertExamples(gng, X)
```

isRunning	<i>isRunning</i>
-----------	------------------

Description

Returns TRUE if GNG object is training

Usage

```
isRunning(object)
```

Arguments

object	GNG object
--------	------------

Details

Check if GNG is running

Examples

```
gng <- GNG(gng.preset.sphere(100))
# FALSE, because did not pass train.online to constructor
print(isRunning(gng))
```

iterations	<i>iterations</i>
------------	-------------------

Description

Print how many iterations it took to learn CEC model

Usage

```
iterations(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:
iterations(c)

## End(Not run)
```

logClusters	<i>logClusters</i>
-------------	--------------------

Description

Print number of clusters that has been recorded at each stage of learning. Data is recorded only if you have chosen to when you created CEC model object.

Usage

```
logClusters(c)
```

Arguments

c object Trained CEC model object.

Format

NULL

Examples

```
## Not run:
logClusters(c)

## End(Not run)
```

logEnergy	<i>logEnergy</i>
-----------	------------------

Description

Print energy that has been recorded at each stage of learning. Data is recorded only if you have chosen to when you created CEC model object.

Usage

```
logEnergy(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:
logEnergy(c)

## End(Not run)
```

meanError	<i>meanError</i>
-----------	------------------

Description

Gets mean error of the graph (note: blocks the execution, O(n))

Usage

```
meanError(object)
```

Arguments

object GNG object

Examples

```
gng <- GNG(gng.preset.sphere(100))
meanError(gng)
```

MultiClassSVM-class *Class MultiClassSVM*

Description

Class MultiClassSVM defines a multiclass SVM model class.

node	<i>node</i>
------	-------------

Description

Retrieves node from resulting graph

Usage

```
node(x, gng_id)
```

Arguments

x	GNG object
gng_id	Id of the node to retrieve. This is the id returned by functions like predict, or centroids

Details

Get GNG node

Examples

```
gng <- GNG(gng.preset.sphere(100))
print(node(gng, 10)$pos)
```

numberNodes	<i>numberNodes</i>
-------------	--------------------

Description

Get current number of nodes in the graph

Usage

```
numberNodes(object)
```

Arguments

object	GNG object
--------	------------

OptimizedGNG	<i>Constructor of Optimized GrowingNeuralGas object.</i>
--------------	--

Description

Construct simplified and optimized GNG object. Can be used to train offline, or online. Data dimensionality shouldn't be too big, if it is consider using dimensionality reduction techniques.

Usage

```
OptimizedGNG(x = NULL, labels = c(), beta = 0.99, alpha = 0.5,
max.nodes = 1000, eps.n = 6e-04, eps.w = 0.05, max.edge.age = 200,
train.online = FALSE, max.iter = 200, dim = 0,
min.improvement = 0.001, lambda = 200, verbosity = 0, seed = -1,
value.range = c(0, 1))
```

Arguments

x	Passed data (matrix of data.frame) for offline training
labels	Every example can be associated with labels that are added to nodes later. By default empty
beta	Decrease the error variables of all node nodes by this fraction (forgetting rate). Default 0.99
alpha	Decrease the error variables of the nodes neighboring to the newly inserted node by this fraction. Default 0.5
max.nodes	Maximum number of nodes (after reaching this size it will continue running, but new nodes won't be added)
eps.n	Strength of adaptation of neighbour node. Default 0.0006
eps.w	Strength of adaptation of winning node. Default 0.05

max.edge.age	Maximum edge age. Decrease to increase speed of change of graph topology. Default 200
train.online	If used will run in online fashion. Default FALSE
max.iter	If training offline will stop if exceeds max.iter iterations. Default 200
dim	Used for training online, specifies dataset example dimensionality
min.improvement	Used for offline (default) training. Controls stopping criterion, decrease if training stops too early. Default 1e-3
lambda	New vertex is added every lambda iterations. Default 200
verbosity	How verbose should the process be, as integer from [0, 6], default: 0
seed	Seed for internal randomization
value.range	All example features should be in this range, required for optimized version of the algorithm. Default (0, 1)

Examples

```
## Not run:
# Train online optimizedGNG. All values in this dataset are in the range (-4.3, 4.3)
X <- gng.preset.sphere(100)
gng <- OptimizedGNG(train.online = TRUE, value.range=c(min(X), max(X)), dim=3, max.nodes=20)
insertExamples(gng, X)
run(gng)
Sys.sleep(10)
pause(gng)

## End(Not run)
```

pause

pause

Description

Pause algorithm

Usage

pause(object)

Arguments

object GNG object

Examples

```
gng <- GNG(gng.preset.sphere(100))
pause(gng)
print(isRunning(gng))
```

plot.Rcpp_CecModel *plot*

Description

Plot clustering found on 2D plot coloring by cluster.

Usage

```
## S3 method for class 'Rcpp_CecModel'  
plot(x, slice = c(), pca = FALSE,  
     ellipses = FALSE, centers = FALSE, ...)
```

Arguments

x	CEC model object.
slice	List of dimentions chosen for display since plot is 2D.
pca	Apply PCA or not
ellipses	Outline clusters.
centers	Marks center of every cluster.
...	other arguments not used by this method.

Details

Plot CEC

Examples

```
## Not run:  
plot(cec)  
plot(cec, slice=c(1,3), ellipses=TRUE)  
plot(cec, slice=c(1,2,3))  
plot(cec, ellipses=TRUE, centers=FALSE)  
plot(cec, pca=TRUE, ellipses=TRUE, centers=FALSE)  
  
## End(Not run)
```

plot.Rcpp_GNGServer *plot GNG object*

Description

Plot resulting graph using igraph plotting

Usage

```
## S3 method for class 'Rcpp_GNGServer'
plot(x, vertex.color = gng.plot.color.cluster,
     layout = layout.fruchterman.reingold, mode = gng.plot.2d,
     vertex.size = 3, ...)
```

Arguments

x	GNG object
vertex.color	How to color vertexes. Possible values: "fast.cluster" (vertex color is set to fastgreedy.community clustering), "label" (rounds to integer label if present), list of integers (colors vertices according to provided list), "none" (every node is white),
layout	igraph layout to be used when plotting. Defaults to layout.fruchterman.reingold. Other good choice is using gng.plot.layout.v2d, which returns two first spatial coordinates.
mode	"2d" (igraph plot) "2d.errors" (igraph plot with mean error log plot)
vertex.size	Size of plotted vertices
...	other arguments not used by this method.

Details

Plot GNG

Note

If you want to "power-use" plotting and plot for instance a subgraph, you might be interested in exporting igraph with convertToIGraph function

Examples

```
## Not run:
gng <- GNG(scaled.wine)
# Plots igraph using first 2 coordinates and colors according to clusters
plot(gng, mode=gng.plot.2d.errors, layout=gng.plot.layout.v2d, vertex.color=gng.plot.color.cluster)

# For more possibilities see gng.plot.* constants

## End(Not run)
```

plot.Rcpp_SVMClient *Plot SVM object*

Description

Plots trained svm data and models discriminative

Usage

```
## S3 method for class 'Rcpp_SVMClient'
plot(x, X = NULL, mode = "normal", cols = c(1,
      2), radius = 3, radius.max = 10, ...)
```

Arguments

x	Trained SVM object
X	Optional new data points to be predicted and plotted in one of the following formats: <code>data.frame</code> , <code>data.matrix</code> ; default: <code>NULL</code>
mode	Which plotting mode to use as string, available are: <ul style="list-style-type: none"> • 'normal' - default mode, plots data in cols argument and a linear decision boundry in available • 'pca' - preforms PCA decomposition and draws data in a subspace of first 2 dimensions from the PCA • 'contour' - countour plot for non-linear kernels
cols	Data dimensions to be plotted as vector of length 2, default: <code>c(1, 2)</code>
radius	Radius of the plotted data points as float, default: 3
radius.max	Maximum radius of data points can be plotted, when model is trained with example weights as float, default: 10
...	other arguments not used by this method.

Examples

```
## Not run:
# here we ause svm is a trained SVM model
plot(svm)
plot(svm, X=x, cols=c(1,3))
plot(svm, mode="pca", radius=5)

## End(Not run)
```

predict	<i>predict</i>
---------	----------------

Description

Classify a new point according to the model (returns index of cluster where given point belong to)

Usage

```
## S3 method for class 'Rcpp_CecModel'
predict(object, x, ...)
```

Arguments

object	Trained CEC model object.
x	Given point.
...	other arguments not used by this method.

predict.gng	<i>predict</i>
-------------	----------------

Description

Retrieves prediction from trained GNG model

Usage

```
## S3 method for class 'Rcpp_GNGServer'
predict(object, x, ...)
```

Arguments

object	Trained model
x	Vector or matrix of examples
...	other arguments not used by this method

Details

Predict

Examples

```
gng <- GNG(gng.preset.sphere(100))
predict(gng, c(1,2,2))
```

```
predict.Rcpp_SVMClient
```

Predict using SVM object

Description

Returns predicted classes or distance to discriminative for provided test examples.

Usage

```
## S3 method for class 'Rcpp_SVMClient'
predict(object, x_test, decision.function = FALSE,
  ...)
```

Arguments

object	Trained SVM object
x_test	Unlabeled data, in one of the following formats: data.frame, data.matrix, SparseM::matrix.csr, Matrix::Matrix, slam::simple_triplet_matrix
decision.function	Uf TRUE returns SVMs decision function (distance of a point from discriminant) instead of predicted labels, default: FALSE
...	other arguments not used by this method.

Examples

```
## Not run:
# firstly, SVM model needs to be trained
svm <- SVM(x, y, core="libsvm", kernel="linear", C=1)
# then we can use it to predict unknown samples
predcit(svm, x_test)

## End(Not run)
```

```
predictComponent
```

predictComponent

Description

Finds connected component closest to given vector(s). On the first execution of function strongly connected components are calculated using igraph::cluster function.

Usage

```
predictComponent(object, x)
```

Arguments

object	GNG object
x	Can be either vector or data.frame.

Details

Find closest component

Examples

```
gng <- GNG(gng.preset.sphere(100))
# Find closest component to c(1,1,1)
predictComponent(gng, c(1,1,1))
```

`print.Rcpp_CecModel` *print*

Description

Print basic information about clusters found. Presents a structure of the cec results object (clusters found)

Usage

```
## S3 method for class 'Rcpp_CecModel'
print(x, ...)
```

Arguments

x	CEC object model.
...	other arguments not used by this method.

Details

Print CEC

```
print.Rcpp_GNGServer  print
```

Description

Print basic information about GNG object

Usage

```
## S3 method for class 'Rcpp_GNGServer'  
print(x, ...)
```

Arguments

x	GNG object model.
...	other arguments not used by this method.

```
print.Rcpp_SVMClient  Print SVM object
```

Description

Prints short summary of a trained model.

Usage

```
## S3 method for class 'Rcpp_SVMClient'  
print(x, ...)
```

Arguments

x	Trained SVM object
...	other arguments not used by this method.

```
Rcpp_CecConfiguration-class  
  Class Rcpp_CecConfiguration.
```

Description

Class Rcpp_CecConfiguration defines a CEC model configuration class.

Rcpp_CecModel-class *Class Rcpp_CecModel.*

Description

Class Rcpp_CecModel defines a CEC model class.

Rcpp_GNGServer-class *Class Rcpp_GNGServer.*

Description

Class Rcpp_GNGServer defines a GNGServer class.

Methods

getClustering() Rcpp::NumericVector getClustering()
getCurrentIteration() unsigned int getCurrentIteration() const
getDatasetSize() unsigned int getDatasetSize() const
getErrorStatistics() Rcpp::NumericVector getErrorStatistics()
getMeanError() double getMeanError()
getNode(...) Rcpp::List getNode(int)
getNumberNodes() unsigned int getNumberNodes() const

Rcpp_SVMClient-class *Class Rcpp_SVMClient.*

Description

Class Rcpp_SVMClient defines a SVM model class.

run	<i>run</i>
-----	------------

Description

Run algorithm (in parallel)

Usage

```
run(object)
```

Arguments

object GNG object

Examples

```
gng <- GNG(gng.preset.sphere(100))
run(gng)
print(isRunning(gng))
```

runAll	<i>runAll</i>
--------	---------------

Description

Starts whole algorithm again with same parameters

Usage

```
runAll(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:
runAll(c)

## End(Not run)
```

runOneIteration	<i>runOneIteration</i>
-----------------	------------------------

Description

runs one iteration of algorithm

Usage

```
runOneIteration(c)
```

Arguments

c object Trained CEC model object.

Examples

```
## Not run:  
runOneIteration(c)  
  
## End(Not run)
```

```
summary.Rcpp_CecModel summary
```

Description

Print detailed information about CEC model object

Usage

```
## S3 method for class 'Rcpp_CecModel'  
summary(object, ...)
```

Arguments

object CEC model object.
... other arguments not used by this method.

Details

Summary CEC

summary.Rcpp_GNGServer

summary

Description

Print basic information about GNG object

Usage

```
## S3 method for class 'Rcpp_GNGServer'  
summary(object, ...)
```

Arguments

object	GNG object model.
...	other arguments not used by this method.

Details

Summary of GNG object

summary.Rcpp_SVMClient

Summary of SVM object

Description

Prints short summary of a trained model.

Usage

```
## S3 method for class 'Rcpp_SVMClient'  
summary(object, ...)
```

Arguments

object	Trained SVM object
...	other arguments not used by this method.

SVM *Create SVM object*

Description

Create and train SVM model object.

Usage

```
SVM(x, ...)
```

```
## S3 method for class 'formula'
```

```
SVM(formula, data, ...)
```

```
## Default S3 method:
```

```
SVM(x, y, core = "libsvm", kernel = "linear",
     prep = "none", transductive.learning = FALSE,
     transductive.posratio = -1, C = 1, gamma = if (is.vector(x)) 1 else
     1/ncol(x), coef0 = 0, degree = 3, class.weights = NULL,
     example.weights = NULL, cache_size = 100, tol = 0.001, max.iter = -1,
     verbosity = 4, class.type = "one.versus.all", svm.options = "", ...)
```

Arguments

x	Training data without labels in one of the following formats: <code>data.frame</code> , <code>data.matrix</code> , <code>SparseM::matrix.csr</code> , <code>Matrix::Matrix</code> , <code>slam::simple_triplet_matrix</code>
...	other arguments not used by this method.
formula	Can be passed with data instead of x, y pair, formula needs to point to labels column, for example: <code>target~</code> .
data	Can be passed instead of x, y pair with formula to mark the labels column, supported formats are: <code>data.frame</code> , <code>data.matrix</code>
y	Labels in one of the following formats: factor, vector. Recommended type is factor
core	Support Vector Machine library to use in training, available are: 'libsvm', 'svmlight'; default: 'libsvm'
kernel	Kernel type as string, available are: 'linear', 'poly', 'rbf', 'sigmoid'; default: 'linear' <ul style="list-style-type: none"> • linear: $x' * w$ • poly: $(\text{gamma} * x' * w + \text{coef0})^{\text{degree}}$ • rbf: $\exp(-\text{gamma} * x - w ^2)$ • sigmoid: $\tanh(\text{gamma} * x' * w + \text{coef0})$
prep	Preprocess method as string, available are: 'none', '2e'; default: 'none'. For more information on 2eSVM see: http://www.sciencedirect.com/science/article/pii/S0957417414004138

<code>transductive.learning</code>	Option got SVM model to deduce missing labels from the dataset, default: FALSE NOTE: this feature is only available with svmLight library, missing labels are marked as 'TR', if none are found and transductive to TRUE, label 0 will be interpreted as missing
<code>transductive.posratio</code>	Fraction of unlabeled examples to be classified into the positive class as float from [0, 1], default: the ratio of positive and negative examples in the training data
<code>C</code>	Cost/complexity parameter, default: 1
<code>gamma</code>	Parameter for poly, rbf and sigmoid kernels, default: 1/n_features
<code>coef0</code>	For poly and sigmoid kernels, default: 0
<code>degree</code>	For poly kernel, default: 3
<code>class.weights</code>	Named vector with weight for each class, default: NULL
<code>example.weights</code>	Vector of the same length as training data with weights for each training example, default: NULL NOTE: this feature is only supported with svmLight library
<code>cache_size</code>	Cache memory size in MB, default: 100
<code>tol</code>	Tolerance of termination criterion, default: 1e-3
<code>max.iter</code>	Depending on library: <ul style="list-style-type: none"> • libsvm: number of iterations after which the training process is killed (it can end earlier if desired tolerance is met), default: 1e6 • svmLight: number of iterations after which if there is no progress training is killed, default: -1 (no limit)
<code>verbosity</code>	How verbose should the process be, as integer from [1, 6], default: 4
<code>class.type</code>	Multiclass algorithm type as string, available are: 'one.versus.all', 'one.versus.one'; default: 'one.versus.one'
<code>svm.options</code>	enables to pass all svmLight command line arguments for more advanced options, for details see http://svmlight.joachims.org/

Value

SVM model object

Examples

```
## Not run:
# train SVM from data in x and labels in y
svm <- SVM(x, y, core="libsvm", kernel="linear", C=1)

# train SVM using a dataset with both data and labels and a formula pointing to labels
formula <- target ~ .
svm <- SVM(formula, data, core="svmlight", kernel="rbf", gamma=1e3)

# train a model with 2eSVM algorithm
data(svm_breast_cancer_dataset)
```

```

ds <- svm.breastcancer.dataset
svm.2e <- SVM(x=ds[,-1], y=ds[,1], core="libsvm", kernel="linear", prep = "2e", C=10);
# more at \url{http://r.gmum.net/samples/svm.2e.html}

# train SVM on a multiclass data set
data(iris)
# with "one vs rest" strategy
svm.ova <- SVM(Species ~ ., data=iris, class.type="one.versus.all", verbosity=0)
# or with "one vs one" strategy
svm.ovo <- SVM(x=iris[,1:4], y=iris[,5], class.type="one.versus.one", verbosity=0)

# we can use svmights sample weighting feature, suppose we have weights vector
# with a weight for every sample in the training data
weighted.svm <- SVM(formula=y~., data=df, core="svmlight", kernel="rbf", C=1.0,
                    gamma=0.5, example.weights=weights)

# svmlight allows us to determine missing labels from a dataset
# suppose we have a labels y with missing labels marked as zeros
svm.transduction <- SVM(x, y, transductive.learning=TRUE, core="svmlight")

# for more in-depth examples visit \url{http://r.gmum.net/getting_started.html}

## End(Not run)

```

svm.accuracy

Measure accuracy score of a prediction

Description

Calculates accuracy of a prediction, returns percent of correctly predicted examples over all test examples.

Usage

```
svm.accuracy(prediction, target)
```

Arguments

prediction	factor or 1 dim vector with predicted classes
target	factor or 1 dim vector with true classes
	#'

Examples

```

## Not run:
# firstly, SVM model needs to be trained
svm <- SVM(x, y, core="libsvm", kernel="linear", C=1)
# then we can use it to predict unknown samples
p <- predcit(svm, x_test)

```

```
acc <- svm.accuracy(p, y)

## End(Not run)
```

```
svm.breastcancer.dataset
      svm.breastcancer.dataset
```

Description

UCI breast cancer dataset

```
svm.transduction      svm.transduction
```

Description

Dataset used in transduction demo on website

```
terminate      terminate
```

Description

Terminate algorithm

Usage

```
terminate(object)
```

Arguments

object GNG object

Examples

```
gng <- GNG(gng.preset.sphere(100))
terminate(gng)
```

```
Tset      Tset
```

Description

Simple dataset in the form of T letter

Index

*Topic **datasets**

- caret.gmumSvmLinear, 4
- caret.gmumSvmPoly, 4
- caret.gmumSvmRadial, 5
- logClusters, 19

*Topic **data**

- cec.ellipsegauss, 7
- cec.ellipsegauss.extra, 8
- cec.mouse1.spherical, 8
- cec.mouse1.spherical.extra, 8
- svm.breastcancer.dataset, 38
- svm.transduction, 38
- Tset, 38

- calculateCentroids, 3
- caret.gmumSvmLinear, 4
- caret.gmumSvmPoly, 4
- caret.gmumSvmRadial, 5
- CEC, 6
- cec.ellipsegauss, 7
- cec.ellipsegauss.extra, 8
- cec.mouse1.spherical, 8
- cec.mouse1.spherical.extra, 8
- centers, 8
- centers, Rcpp_CecModel-method (centers), 8
- clustering, 9
- convertToIGraph, 9
- covMatrix, 10
- covMatrix, Rcpp_CecModel-method (covMatrix), 10
- energy, 10
- energy, Rcpp_CecModel-method (energy), 10
- errorStatistics, 11
- findClosests, 11
- get.wine.dataset.X, 12
- get.wine.dataset.y, 12

- getDataset, 13
- getDataset, Rcpp_CecModel-method (getDataset), 13
- GNG, 13
- gng.plot.layout.v2d, 15
- gng.preset.cube, 15
- gng.preset.plane, 16
- gng.preset.sphere, 16
- gngLoad, 17
- gngSave, 17
- insertExamples, 18
- isRunning, 18
- iterations, 19
- iterations, Rcpp_CecModel-method (iterations), 19
- logClusters, 19
- logClusters, Rcpp_CecModel-method (logClusters), 19
- logEnergy, 20
- logEnergy, Rcpp_CecModel-method (logEnergy), 20
- meanError, 20
- MultiClassSVM-class, 21
- node, 21
- numberNodes, 22
- OptimizedGNG, 22
- pause, 23
- plot.Rcpp_CecModel, 24
- plot.Rcpp_GNGServer, 25
- plot.Rcpp_SVMClient, 26
- predict, 27
- predict, Rcpp_CecModel-method (predict), 27
- predict.gng, 27
- predict.Rcpp_CecModel (predict), 27

- predict.Rcpp_GNGServer (predict.gng), [27](#)
- predict.Rcpp_SVMClient, [28](#)
- predictComponent, [28](#)
- print.Rcpp_CecModel, [29](#)
- print.Rcpp_GNGServer, [30](#)
- print.Rcpp_SVMClient, [30](#)

- Rcpp_CecConfiguration-class, [30](#)
- Rcpp_CecModel-class, [31](#)
- Rcpp_GNGServer-class, [31](#)
- Rcpp_SVMClient-class, [31](#)
- run, [32](#)
- runAll, [32](#)
- runAll,Rcpp_CecModel-method (runAll), [32](#)
- runOneIteration, [33](#)
- runOneIteration,Rcpp_CecModel-method
(runOneIteration), [33](#)

- summary.Rcpp_CecModel, [33](#)
- summary.Rcpp_GNGServer, [34](#)
- summary.Rcpp_SVMClient, [34](#)
- SVM, [35](#)
- svm.accuracy, [37](#)
- svm.breastcancer.dataset, [38](#)
- svm.transduction, [38](#)

- terminate, [38](#)
- Tset, [38](#)