# Package 'semiArtificial'

July 2, 2014

**Title** Generator of semi-artificial data

**Version** 1.2.0

**Date** 2014-03-15

**Author** Marko Robnik-Sikonja

**Maintainer** Marko Robnik-Sikonja <marko.robnik@fri.uni-lj.si>

**Description** Package semiArtificial contains methods to generate and evaluate semi-artificial data sets. Based on a given data set different methods learn data properties using machine learning algorithms and generate new data with the same properties.
The package currently includes the following data generator:
-a RBF network based generator using rbfDDA from RSNNS package.
More generators are planned in near future. Data evaluation support tools include:
-single attribute based statistical evaluation: mean, median,standard deviation, skewness, kurtosis, KS test, Hellinger distance -evaluation based on clustering using Adjusted Rand Index (ARI) -evaluation based on classification performance with various learning models, eg, random forests.

**License** GPL-3

**URL** http://lkm.fri.uni-lj.si/rmarko/software/

**Imports** RSNNS,CORElearn,MASS,nnet,cluster,mclust,fpc,stats,timeSeries,timeDate

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-03-17 20:36:07

## R topics documented:

---

`semiArtificial-package`
*Generation and evaluation of semi-artificial data*

---

## Description

The package semiArtificial contains methods to generate and evaluate semi-artificial data sets. Different data generators take a data set as an input, learn its properties using machine learning algorithms and generates new data with the same properties.

## Details

The package currently includes the following data generators:

- a RBF network based generator using rbfDDA model from RSNNS package.

Data evaluation support tools include:

- statistical evaluation: mean, median,standard deviation, skewness, kurtosis,
- evaluation based on clustering using Adjusted Rand Index (ARI),
- evaluation based on classification with random forests.

Further software may be available from `http://lkm.fri.uni-lj.si/rmarko/software`.

## Author(s)

Marko Robnik-Sikonja

## References

Marko Robnik-Sikonja: Not enough data? Generate it!. *Technical Report, University of Ljubljana, Faculty of Computer and Information Science*, 2014

Other references are available from `http://lkm.fri.uni-lj.si/rmarko/papers/`

## See Also

`rbfDataGen`, `newdata.RBFgenerator`, `dataSimilarity`, `ariCompare`, `performanceCompare`.

ariCompare *Evaluate clustering similarity of two data sets with ARI*

## Description

Similarity of two data sets is compared with Adjusted Rand Index (ARI).

## Usage

```
ariCompare(data1, data2)
```

## Arguments

| | |
|---|---|
| data1 | A data.frame containing the reference data. |
| data2 | A data.frame with the same number and names of columns as data1. |

## Details

The function compares data stored in data1 with data2 by first performing partitioning around medoids (PAM) clustering on data1. Instances from data2 are than assigned to the cluster with the closest medoid. In second step PAM clustering is performed on data2 and instances from data1 are assigned to the clusters with closest medoids. The procedure gives us two clusterings on the same instances which we can compare with ARI. The higher the value of ARI the more similar the two data sets.

## Value

The method returns a value of ARI.

## Author(s)

Marko Robnik-Sikonja

## See Also

newdata.RBFgenerator.

## Examples

```
# use iris data set

# create RBF generator
irisGenerator<- rbfDataGen(Species~.,iris)

# use the generator to create new data
irisNew <- newdata(irisGenerator, size=200)

# compare ARI computed on clustering with original and new data
```

```
ariCompare(iris, irisNew)
```

---

`dataSimilarity` *Evaluate statistical similarity of two data sets*

---

## Description

Use mean, standard deviation, skewness, kurtosis, Hellinger distance and KS test to compare similarity of two data sets.

## Usage

```
dataSimilarity(data1, data2, dropDiscrete=NA)
```

## Arguments

| | |
|---|---|
| `data1` | A `data.frame` containing the reference data. |
| `data2` | A `data.frame` with the same number and names of columns as `data1`. |
| `dropDiscrete` | A vector discrete attribute indices to skip in comparison. Typically we might skip class, because its distribution was forced by the user. |

## Details

The function compares data stored in `data1` with `data2` on per attribute basis by computing several statistics: mean, standard deviation, skewness, kurtosis, Hellinger distance and KS test.

## Value

The method returns a list of statistics computed on both data sets:

| | |
|---|---|
| `equalInstances` | The number of instances in `data2` equal to the instances in `data1`. |
| `stats1num` | A matrix with rows containing statistics (mean, standard deviation, skewness, and kurtosis) computed on numeric attributes of `data1`. |
| `stats2num` | A matrix with rows containing statistics (mean, standard deviation, skewness, and kurtosis) computed on numeric attributes of `data2`. |
| `ksP` | A vector with p-values of Kolmogorov-Smirnov two sample tests, performed on matching attributes from `data1` and `data2`. |
| `freq1` | A list with value frequencies for discrete attributes in `data1`. |
| `freq2` | A list with value frequencies for discrete attributes in `data2`. |
| `dfreq` | A list with differences in frequencies of discrete attributes' values between `data1` and `data2`. |
| `dstatsNorm` | A matrix with rows containing difference between statistics (mean, standard deviation, skewness, and kurtosis) computed on [0,1] normalized numeric attributes for `data1` and `data2`. |
| `hellingerDist` | A vector with Hellinger distances between matching attributes from `data1` and `data2`. |

## Author(s)

Marko Robnik-Sikonja

## See Also

newdata.RBFgenerator.

## Examples

```
# use iris data set, split into training and testing data
set.seed(12345)
train <- sample(1:nrow(iris),size=nrow(iris)*0.5)
irisTrain <- iris[train,]
irisTest <- iris[-train,]

# create RBF generator
irisGenerator<- rbfDataGen(Species~.,irisTrain)

# use the generator to create new data
irisNew <- newdata(irisGenerator, size=100)

# compare statistics of original and new data
dataSimilarity(irisTest, irisNew)
```

---

# newdata.RBFgenerator *Generate semi-artificial data with RBF generator*

---

## Description

Using a generator build with rbfDataGen the method generates size new instances.

## Usage

```
## S3 method for class 'RBFgenerator'
newdata(object, size, var=c("estimated","Silverman"),
                                classProb=NULL, defaultSpread=NULL, ... )
```

## Arguments

| | |
|---|---|
| object | An object of type RBFgenerator containing a generator structure as returned by rbfDataGen. |
| size | A number of instances to generate. |
| var | The method of kernel width (variance) estimation. Supported options are "estimated" and "Silverman". |
| classProb | A vector of desired class value probability distribution. Default value classProb=NULL uses probability distribution of the generator training instances. |

| | |
|---|---|
| defaultSpread | A numeric value replacing zero spread in case var="estimated" is used. Default value defaultSpread=NULL keeps zero spread values. |
| ... | Additional parameters passed to future extended versions of the function. |

## Details

The function uses the object structure as returned by rbfDataGen containing descriptions of the Gaussian kernels which model the original data. The kernels are used to generate a required number of new instances.

The kernel width of provided kernels can be set in two ways. By setting var="estimated" the estimated spread of the training instances that have the maximal activation value for the particular kernel is used. Using var="Silverman" width is set by the generalization of Silverman's rule of thumb to multivariate case (unreliable for larger dimensions).

## Value

The method returns a data.frame object with required number of instances.

## Author(s)

Marko Robnik-Sikonja

## See Also

rbfDataGen.

## Examples

```
# inspect properties of the iris data set
plot(iris, col=iris$Species)
summary(iris)

# create RBF generator
irisGenerator<- rbfDataGen(Species~.,iris)

# use the generator to create new data
irisNew <- newdata(irisGenerator, size=200)

#inspect properties of the new data
plot(irisNew, col = irisNew$Species) #plot generated data
summary(irisNew)
```

performanceCompare *Evaluate similarity of two data sets based on classification performance*

## Description

Classification performance (accuracy, AUC,...) of two data sets is used to compare similarity of two data sets.

## Usage

```
performanceCompare(data1, data2, formula, model="rf", stat="accuracy", ...)
```

## Arguments

| | |
|---|---|
| data1 | A data.frame containing the reference data. |
| data2 | A data.frame with the same number and names of columns as data1. |
| formula | A formula specifying the response and predictive variables. |
| model | A predictive model used for performance comparison. The default value "rf" stands for random forest, but any classification model supported by function CoreModel in CORElearn package can be used. |
| stat | A statistics used as performance indicator. The default value is "accuracy" but other values supported by modelEval from CORElearn package can be used eg. "AUC" or "brierScore". |
| ... | Additional parameters passed to CoreModel function. |

## Details

The function compares data stored in data1 with data2 by comparing models constructed on data1 and evaluated on both data1 and data2 with models built on data2 and evaluated on both data1 and data2. The difference between these performances are indicative on similarity of the data sets if used in machine learning and data mining. The performance indicator used is determined by parameter stat.

## Value

The method returns a list of performance indicators computed on both data sets:

| | |
|---|---|
| diff.m1 | The difference between performance of model built on data1 (and evaluated on both data1 and data2.) |
| diff.m2 | The difference between performance of model built on data2 (and evaluated on both data1 and data2.) |
| perf.m1d1 | The performance of model built on data1 on data1. |
| perf.m1d2 | The performance of model built on data1 on data2. |
| perf.m2d1 | The performance of model built on data2 on data1. |
| perf.m2d2 | The performance of model built on data2 on data2. |

## Author(s)

Marko Robnik-Sikonja

## See Also

newdata.RBFgenerator.

## Examples

```
# use iris data set

# create RBF generator
irisGenerator<- rbfDataGen(Species~.,iris)

# use the generator to create new data
irisNew <- newdata(irisGenerator, size=200)

# compare statistics of original and new data
performanceCompare(iris, irisNew, Species~.)
```

---

# rbfDataGen *A data generator based on RBF network*

---

## Description

Using given formula and data the method builds a RBF network and extracts its properties thereby preparing a data generator which can be used with newdata.RBFgenerator method to generate sem-artificial data.

## Usage

```
rbfDataGen(formula, data, eps=1e-4, minSupport=1, nominal=c("encodeBinary","asInteger"))
```

## Arguments

| | |
|---|---|
| formula | A formula specifying the response and variables to be modeled. |
| data | A data frame with training data. |
| eps | The minimal probability considered in data generator to be larger than 0. |
| minSupport | The minimal number of instances defining a Gaussian kernel to copy the kernel to the data generator. |
| nominal | The way how to treat nominal features. The option "asInteger" converts factors into integers and treats them as numeric features. The option "encodeBinary" converts each nominal attribute into a set of binary features, which encode the nominal value, eg., for three valued attribute three binary attributes are constructed, each encoding a presence of one nominal value with 0 or 1. |

## Details

Parameter formula is used as a mechanism to select features (attributes) and the prediction variable (response, class). Only simple terms can be used and interaction terms are not supported. The simplest way is to specify just the response variable using e.g. class ~ .. See examples below.

A RBF network is build using rbfDDA from RSNNS package. The learned Gaussian kernels are extracted and used in data generation with newdata.RBFgenerator method.

## Value

The created model is returned as a structure of class RBFgenerator, containing the following items:

| | |
|---|---|
| noGaussians | The number of extracted Gaussian kernels. |
| centers | A matrix of Gaussian kernels' centers, with one row for each Gaussian kernel. |
| probs | A vector of kernel probabilities. Probabilities are defined as relative frequencies of training set instances with maximal activation in the given kernel. |
| unitClass | A vector of class values, one for each kernel. |
| bias | A vector of kernels' biases, one for each kernel. The bias is multiplied by the kernel activation to produce output value of given RBF network unit. |
| spread | A matrix of estimated variances for the kernels, one row for each kernel. The j-th value in i-th row represents the variance of training instances for j-th attribute with maximal activation in i-th Gaussian. |
| gNoActivated | A vector containing numbers of training instances with maximal activation in each kernel. |
| noAttr | The number of attributes in training data. |
| datNames | A vector of attributes' names. |
| originalNames | A vector of original attribute names. |
| attrClasses | A vector of attributes' classes (ie, data types like numeric or factor). |
| attrLevels | A list of levels for discrete attributes (with class factor). |
| attrOrdered | A vector of type logical indicating whether the attribute is ordered (only possible for attributes of type factor. |
| normParameters | A list of parameters for normalization of attributes to [0,1]. |
| noCol | The number of columns in the internally generated data set. |
| isDiscrete | A vector of type logical, each value indicating whether a respective attribute is discrete. |
| noAttrGen | The number of attributes to generate. |
| nominal | The value of parameter nominal. |

## Author(s)

Marko Robnik-Sikonja

## References

Marko Robnik-Sikonja: Not enough data? Generate it!. *Technical Report, University of Ljubljana, Faculty of Computer and Information Science*, 2014

Other references are available from http://lkm.fri.uni-lj.si/rmarko/papers/

## See Also

newdata.RBFgenerator.

## Examples

```
# use iris data set, split into training and testing, inspect the data
set.seed(12345)
train <- sample(1:nrow(iris),size=nrow(iris)*0.5)
irisTrain <- iris[train,]
irisTest <- iris[-train,]

# inspect properties of the original data
plot(irisTrain, col=iris$Species)
summary(irisTrain)

# create rbf generator
irisGenerator<- rbfDataGen(Species~.,irisTrain)

# use the generator to create new data
irisNew <- newdata(irisGenerator, size=200)

#inspect properties of the new data
plot(irisNew, col = irisNew$Species) #plot generated data
summary(irisNew)
```

# Index