Package 'ganGenerativeData'

May 24, 2021

Type Package

Title Generate Generative Data for a Data Source

Version 1.1.1

Date 2021-05-22

Author Werner Mueller

Maintainer Werner Mueller <werner.mueller5@chello.at>

Description Generative Adversarial Networks are applied to generate generative data for a data source. In iterative training steps the distribution of generated data converges to that of the data source. Reference: Goodfellow et al. (2014) <arXiv:1406.2661v1>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.3), tensorflow (>= 2.0.0)

LinkingTo Rcpp

RoxygenNote 7.0.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-05-24 16:30:02 UTC

R topics documented:

ganGenerativeData-package 2
dsActivateColumns 5
dsCreateWithDataFrame 6
dsDeactivateColumns 7
dsGetActiveColumnNames 7
dsGetInactiveColumnNames 8
dsGetNumberOfRows 8
dsGetRow 9
dsRead 9
dsWrite 10
gdGenerate 10
gdGetNumberOfRows 11

gdGetRow 12
gdPlot2dProjection 12
gdRead 13

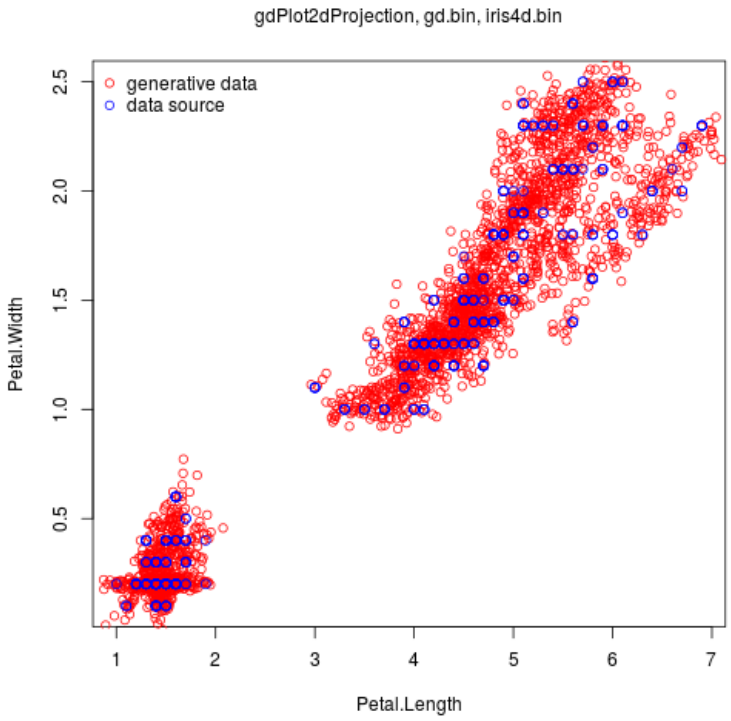
Index **14**

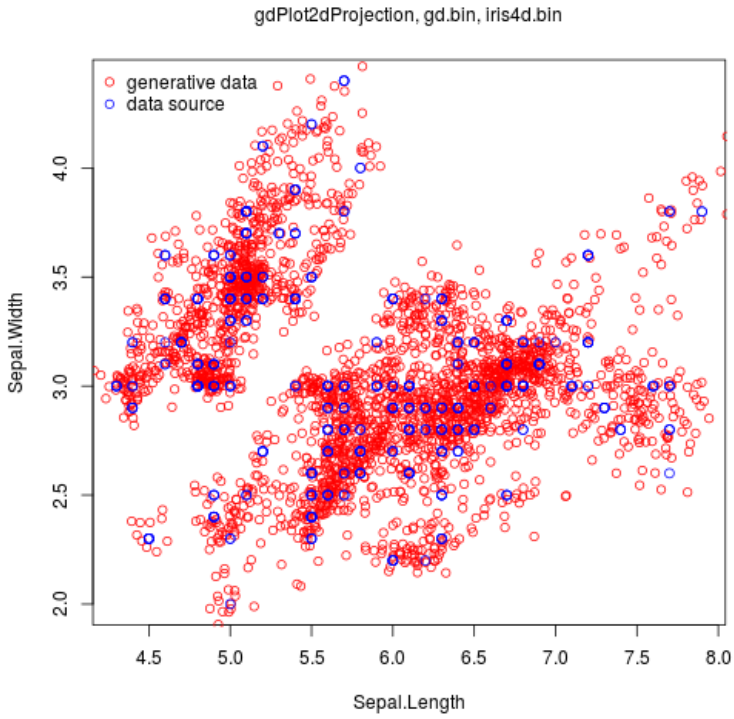
`ganGenerativeData-package`

Generate generative data for a data source

Description

Generative Adversarial Networks are applied to generate generative data for a data source. In iterative training steps the distribution of generated data converges to that of the data source. Two-dimensional projections of generative data for the iris dataset are shown in the inserted images:





Details

The API includes functions for topics "definition of data source" and "generation of generative data". Main function of first topic is `dsCreateWithDataFrame()` which creates a data source with passed data frame. Main function of second topic is `gdGenerate()` which generates generative data for a data source.

1. Definition of data source

`dsCreateWithDataFrame()` Create a data source with passed data frame.

`dsActivateColumns()` Activate columns of a data source in order to include them in generation of generative data. By default columns are active.

`dsDeactivateColumns()` Deactivate columns of a data source in order to exclude them in generation of generative data. Note that in this version only columns with values of type double or float can be used in generation of generative data. All columns with values of other type have to be deactivated.

`dsGetActiveColumnNames()` Get names of active columns of a data source.

`dsGetInactiveColumnNames()` Get names of inactive columns of a data source.

`dsWrite()` Write created data source including settings of active columns to a file in binary format. This file will be used as input in functions of topic "generation of generative data".

`dsRead()` Read a data source from a file that was written with `dsWrite()`.

`dsGetNumberOfRows()` Get number of rows in a data source.

`dsGetRow()` Get a row in a data source.

2. Generation of generative data

`gdGenerate()` Read a data source from a file, generate generative data for the data source in iterative training steps and write generated generative data to a file in binary format.

`gdRead()` Read generative data and data source from corresponding files. Read in generative data and data source are accessed in `gdPlot2dProjection()`, generative data in `gdGetRow()`.

`gdPlot2dProjection()` Create an image showing a two-dimensional projection of generative data and data source and write it to a file.

`gdGetNumberOfRows()` Get number of rows in generative data.

`gdGetRow()` Get a row in generative data.

Author(s)

Werner Mueller

Maintainer: Werner Mueller <werner.mueller5@chello.at>

References

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014), *"Generative Adversarial Nets"*, <arXiv:1406.2661v1>

Examples

```
# Generate generative data for iris dataset

# Load and install library tensorflow
## Not run:
library(tensorflow)
install_tensorflow()
## End(Not run)
```

```
# Load library
library(ganGenerativeData)

# 1. Definition of data source for iris dataset

# Create a data source with built in iris dataframe.
dsCreateWithDataFrame(inDataFrame = iris)

# Deactivate column for index 5 in order to exclude it in generation of generative data.
dsDeactivateColumns(c(5))

# Get active columns Sepal.Length, Sepal.Width, Petal.Length, Petal.Width.
dsGetActiveColumnNames()

# Write data source including settings of active columns to file "iris4d.bin" in binary format.
dsWrite("iris4d.bin")

# 2. Generation of generative data for iris data source

# Read data source from file "iris4d.bin",
# generate generative data in iterative training steps and
# write generated generative data to file "gd.bin".
gdGenerate("iris4d.bin", "gd.bin", 2500, 0.95, c(1, 2))

# Read generative data from file "gd.bin" and data source from "iris4d.bin"
gdRead("gd.bin", "iris4d.bin")

# Create an image showing a two-dimensional projection of generative data and
# data source for column indices 1, 2 and
# write it to file "gd12d.png"
gdPlot2dProjection("gd12d.png", c(1, 2), 55000, 2500)

# Create an image showing a two-dimensional projection of generative data and
# data source for column indices 3, 4 and
# write it to file "gd34d.png"
gdPlot2dProjection("gd34d.png", c(3, 4), 55000, 2500)

# Get number of rows in generative data
gdGetNumberOfRows()

# Get first row in generative data
gdGetRow(1)
```

`dsActivateColumns` *Activate columns*

Description

Activate columns of a data source in order to include them in generation of generative data. By default columns are active.

Usage

```
dsActivateColumns(columnVector)
```

Arguments

columnVector	Vector of column indices

Value

None

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsGetActiveColumnNames()
dsDeactivateColumns(c(5))
dsGetActiveColumnNames()
dsActivateColumns(c(5))
dsGetActiveColumnNames()
```

---

dsCreateWithDataFrame *Create a data source with passed data frame*

Description

Create a data source with passed data frame.

Usage

```
dsCreateWithDataFrame(inDataFrame)
```

Arguments

inDataFrame	Name of data frame

Value

None

Examples

```
# Create a data source and with built in iris data frame.
dsCreateWithDataFrame(inDataFrame = iris)
```

dsDeactivateColumns *Deactivate columns*

Description

Deactivate columns of a data source in order to exclude them in generation of generative data. Note that in this version only columns with values of type double or float can be used in generation of generative data. All columns with values of other type have to be deactivated.

Usage

```
dsDeactivateColumns(columnVector)
```

Arguments

columnVector	Vector of column indices

Value

None

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsDeactivateColumns(c(5))
dsGetInactiveColumnNames()
```

dsGetActiveColumnNames *Get active column names*

Description

Get active column names of a data source

Usage

```
dsGetActiveColumnNames()
```

Value

Names of active columns

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsGetActiveColumnNames()
```

dsGetInactiveColumnNames *Get inactive column names*

Description

Get inactive column names of a data source

Usage

```
dsGetInactiveColumnNames()
```

Value

Names of inactive columns

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsGetInactiveColumnNames()
```

dsGetNumberOfRows *Get number of rows*

Description

Get number of rows in a data source

Usage

```
dsGetNumberOfRows()
```

Value

Number of rows

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsGetNumberOfRows()
```

dsGetRow *Get a row in a data source*

Description

Get a row in a data source containing values of columns for a row index

Usage

```
dsGetRow(index)
```

Arguments

index	Index of row

Value

Row in data source

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsGetRow(1)
```

dsRead *Read a data source from file*

Description

Read a data source from a file in binary format

Usage

```
dsRead(inFileName)
```

Arguments

inFileName	Name of data source file

Value

None

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsDeactivateColumns(c(5))
dsWrite("iris4d.bin")
dsRead("iris4d.bin")
```

dsWrite *Write a data source to file*

Description

Write a data source to a file in binary format

Usage

```
dsWrite(outFileName)
```

Arguments

outFileName	Name of data source file

Value

None

Examples

```
dsCreateWithDataFrame(inDataFrame = iris)
dsDeactivateColumns(c(5))
dsWrite("iris4d.bin")
```

gdGenerate *Generate generative data for a data source*

Description

Read a data source from a file, generate generative data for the data source in iterative training steps and write generated generative data to a file in binary format. When a higher number of iterations is used the distribution of generated generative data gets closer to that of the data source.

Usage

```
gdGenerate(
  inDataSourceFileName,
  outGenerativeDataFileName,
  numberOfIterations,
  keepProbability,
  columnIndices
)
```

Arguments

`inDataSourceFileName`
: Name of data source file

`outGenerativeDataFileName`
: Name of generative data file

`numberOfIterations`
: Number of iterations. In this version the limit of number of iterations is set to 25000.

`keepProbability`
: Value in the range of 0 to 1 which is used in training of neural networks to train generalized networks.

`columnIndices`
: Vector of two column indices that are used to show two-dimensional projections of normalized generated generative data and data source for a training step in RStudio Plots pane. Indices refer to indices of active columns of data source.

Value

None

Examples

```
gdGenerate("iris4d.bin", "gd.bin", 2500, 0.95, c(1, 2))
```

---

gdGetNumberOfRows *Get number of rows*

Description

Get number of rows in generative data

Usage

```
gdGetNumberOfRows()
```

Value

Number of rows

Examples

```
gdRead("gd.bin")
gdGetNumberOfRows()
```

gdGetRow *Get a row in generative data*

Description

Get a row in generative data containing values of active columns for a row index

Usage

```
gdGetRow(index)
```

Arguments

index	Index of row

Value

Row in generative data

Examples

```
gdRead("gd.bin")
gdGetRow(1)
```

gdPlot2dProjection *Create an image file for generative data and data source*

Description

Create an image file containing a two-dimensional projection for randomly selected rows in generative data and data source. Data points of data source are drawn above data points of generative data.

Usage

```
gdPlot2dProjection(
  outImageFileName,
  columnIndices,
  numberOfRandomGdPoints,
  numberOfRandomDsPoints
)
```

Arguments

`outImageFileName`
: Name of image file

`columnIndices`
: Vector of two column indices that are used for the two-dimensional projection. The indices refer to indices of active columns of data source.

`numberOfRandomGdPoints`
: Number of randomly selected rows in generative data

`numberOfRandomDsPoints`
: Number of randomly selected rows in data source

Value

None

Examples

```
gdRead("gd.bin", "iris4d.bin")
gdPlot2dProjection("gd34d.png", c(3, 4), 55000, 2500)
gdPlot2dProjection("gd12d.png", c(1, 2), 55000, 2500)
```

---

gdRead *Read generative data and data source*

Description

Read generative data and data source from corresponding files. Read in generative data and data source are accessed in gdPlot2dProjection(), generative data in gdGetRow().

Usage

```
gdRead(inGenerativeDataFileName, inDataSourceFileName = "")
```

Arguments

`inGenerativeDataFileName`
: Name of generative data file

`inDataSourceFileName`
: Name of data source file

Value

None

Examples

```
gdRead("gd.bin", "iris4d.bin")
```

Index

∗ **package**
ganGenerativeData-package, 2

dsActivateColumns, 5
dsCreateWithDataFrame, 6
dsDeactivateColumns, 7
dsGetActiveColumnNames, 7
dsGetInactiveColumnNames, 8
dsGetNumberOfRows, 8
dsGetRow, 9
dsRead, 9
dsWrite, 10

ganGenerativeData
*(*ganGenerativeData-package*)*, 2
ganGenerativeData-package, 2
gdGenerate, 10
gdGetNumberOfRows, 11
gdGetRow, 12
gdPlot2dProjection, 12
gdRead, 13