Package 'ganDataModel'

May 8, 2026

Type Package

Title Build a Metric Subspaces Data Model for a Data Source

Version 2.0.1

Date 2025-12-19

Author Werner Mueller [aut, cre]

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

Description Neural networks are applied to create a density value function which approximates density values for a data source. The trained neural network is analyzed for different levels. For each level metric subspaces with density values above a level are determined. The obtained set of metric subspaces and the trained neural network are assembled into a data model. A prerequisite is the definition of a data source, the generation of generative data and the calculation of density values. These tasks are executed using package 'ganGenerativeData' <https://cran.r-project.org/package=ganGenerativeData>.

License GPL (>= 2)

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

LinkingTo Rcpp

RoxygenNote 7.3.3

SystemRequirements TensorFlow (https://www.tensorflow.org)

NeedsCompilation yes

Encoding UTF-8

Repository CRAN

Date/Publication 2025-12-19 11:40:02 UTC

Contents

`ganDataModel-package` *Build a Metric Subspaces Data Model for a Data Source*

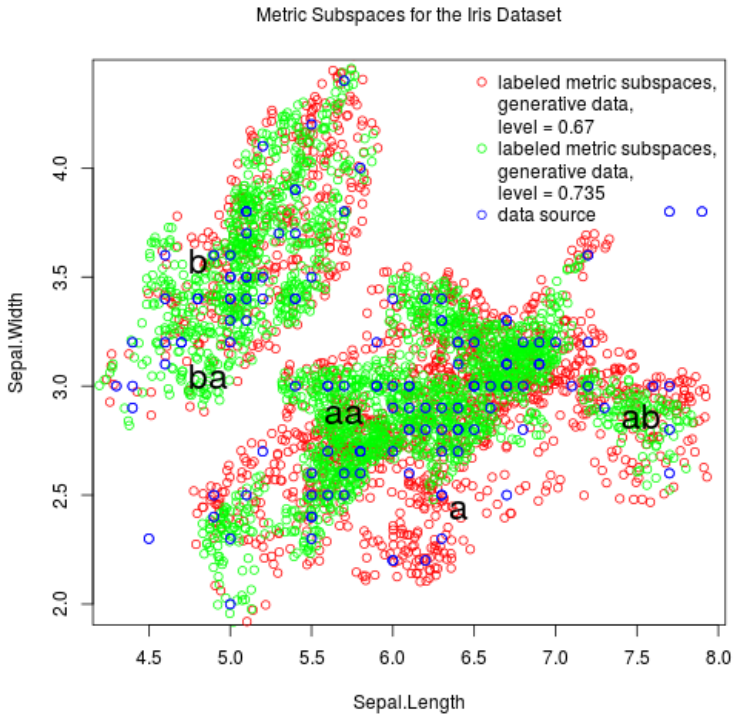
Description

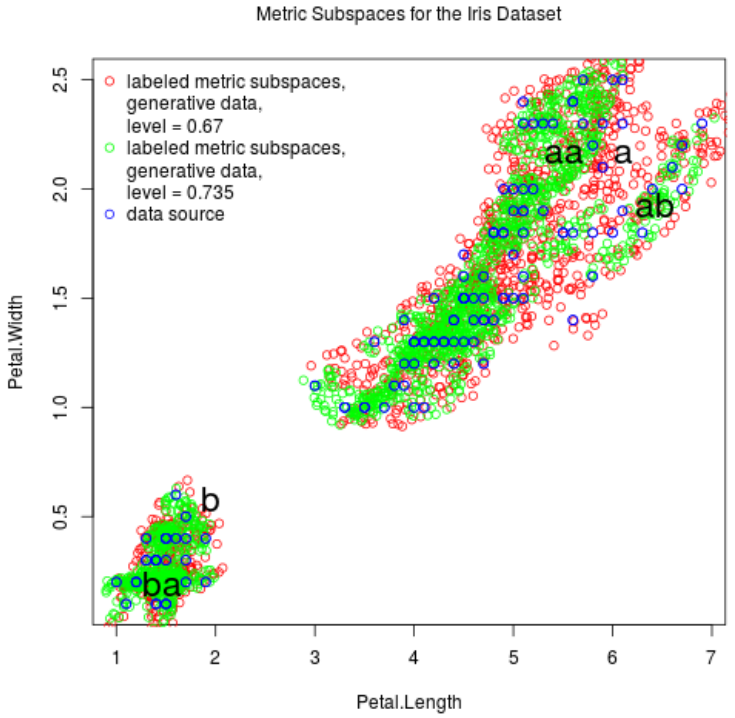
Neural networks are applied to create a density value function which approximates density values for a data source. The trained neural network is analyzed for different levels. For each level metric subspaces with density values above a level are determined. The obtained set of metric subspaces and the trained neural network are assembled into a data model. A prerequisite is the definition of a data source, the generation of generative data and the calculation of density values. These tasks are executed using package 'ganGenerativeData' <https://cran.r-project.org/package=ganGenerativeData>.

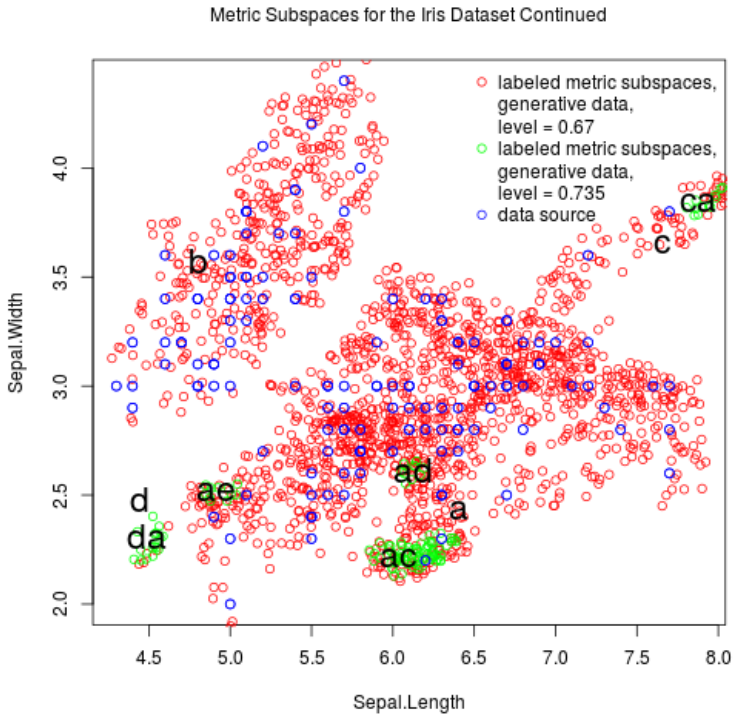
Properties of built metric subspaces:

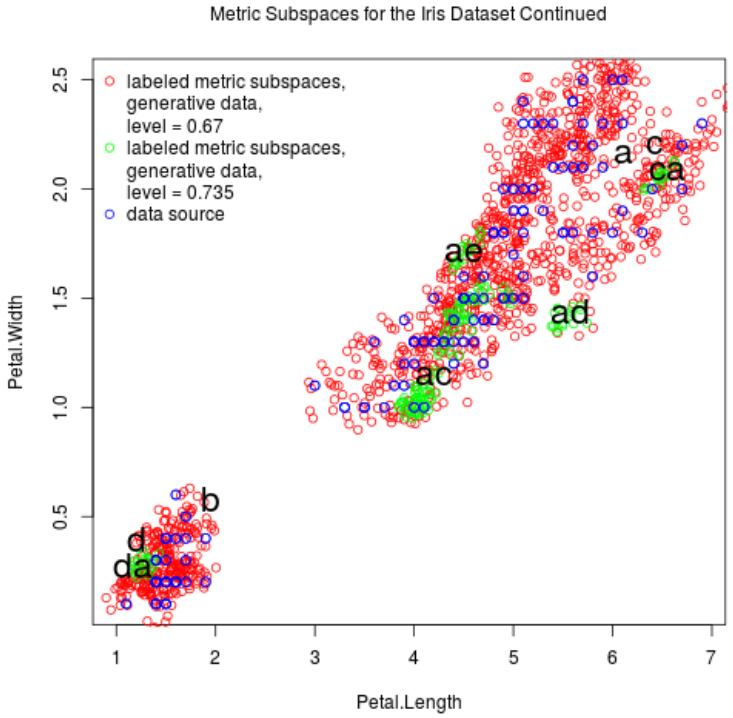
1. They contain data with continuously varying density values above a level.
2. They have the topological property connected. In topology a space is connected when it cannot be represented as the union of disjoint open subspaces.
3. An inclusion relation is defined on them by levels. Higher level metric subspaces are contained in lower level ones.

The inserted images show two-dimensional projections of generative data contained in metric subspaces with assigned labels for the iris dataset.


Metric Subspaces for the Iris Dataset
labeled metric subspaces,
generative data,
level = 0.67
labeled metric subspaces,
generative data,
level = 0.735
data source
b
ba
aa
ab
a
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Metric Subspaces for the Iris Dataset
labeled metric subspaces,
generative data,
level = 0.67
labeled metric subspaces,
generative data,
level = 0.735
data source
b
ba
aa
a
ab
Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


Metric Subspaces for the Iris Dataset Continued
labeled metric subspaces, generative data, level = 0.67
labeled metric subspaces, generative data, level = 0.735
data source
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
b
ca
c
d
ae
ad
a
da
ac


Metric Subspaces for the Iris Dataset Continued
labeled metric subspaces, generative data, level = 0.67
labeled metric subspaces, generative data, level = 0.735
data source
Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
a
c
ca
ae
ad
ac
b
d
da

Details

The API includes main functions `dmTrain()` and `dmBuildMetricSubspaces()`. `dmTrain()` trains a neural network that approximates density values for a data source. `dmBuildMetricSubspaces()` analyzes the trained neural network for a level and determines metric subspaces with density values above a level. The API is used as follows:

1. Prerequisite for building a metric subspaces data model: Create a data source, generate generative data and calculate density values using package ganGenerativeData

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

`dsDeactivateColumns()` Deactivate columns of a data source in order to exclude them in generation of generative data. In current 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.

`dsWrite()` Write created data source including settings of active columns to a file in binary format.

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

`gdCalculateDensityValues()` Read generative data from a file, calculate density values and write generative data with assigned density values to original file.

2. Build a metric subspaces data model

`dmTrain()` Read a data source and generative data from files, train a neural network which approximates density values for a data source in iterative training steps, create a data model containing the trained neural network and write it to a file in binary format.

`dmBuildMetricSubspaces()` Read a data model and generative data from files, analyze the trained neural network in the data model for a level, determine metric subspaces with density values above a level, add obtained metric subspaces to the data model and write it to original file.

`dmRemoveMetricSubspaces()` Remove metric subspaces in a data model for a level.

`dmRead()` Read a data model and generative data from files.

`dmGetLevels()` Get levels for metric subspaces in a data model.

`dmGetMetricSubspacesProperties()` Get metric subspace properties in a data model for a level.

`dmGetContainedInMetricSubspaces()` Get metric subspaces in a data model in which a data record is contained.

`dmPlotMetricSubspaceParameters()` Specify plot parameters for metric subspaces for a level.

`dmPlotEvaluateDataSourceParameters()` Specify plot parameters for evaluated data source.

`dmPlotMetricSubspaces()` Create an image file containing two-dimensional projections of generative data contained in metric subspaces and evaluated data source.

`dmReset()` Reset API.

Author(s)

Werner Mueller

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

References

Package 'ganGenerativeData' <https://cran.r-project.org/package=ganGenerativeData>

Examples

```
# Environment used for execution of examples:

# Operating system: Ubuntu 24.04.3 LTS
# Compiler: g++ 13.3.0 (supports C++17 standard)
# Software: Python 3.12.3, TensorFlow 2.16.2
# R environment: R 4.5.2, RStudio 2024.12.1
# Installed packages: 'Rcpp' 1.1.0, 'tensorflow' 2.20.0,
# 'ganGenerativeData' 2.1.6, 'ganDataModel' 2.0.0

# Package 'tensorflow' provides an interface to machine learning framework
# TensorFlow. To complete the installation function install_tensorflow() has to
# be called.
## Not run:
library(tensorflow)
install_tensorflow()
## End(Not run)

# 1. Prerequisite for building a metric subspaces data model for the iris
# dataset: Create a data source, generate generative data and calculate density
# values for the iris dataset.

# Load library
## Not run:
library(ganGenerativeData)
## End(Not run)

# Create a data source with passed iris data frame.
## Not run:
dsCreateWithDataFrame(iris)
```

```
## End(Not run)

# Deactivate the column with index 5 and name Species in order to exclude it in
# generation of generative data.
## Not run:
dsDeactivateColumns(c(5))
## End(Not run)

# Write the data source including settings of active columns to file "ds.bin" in
# binary format.
## Not run:
dsWrite("ds.bin")
## End(Not run)

# Read data source from file "ds.bin", train a generative model in iterative
# training steps (used number of iterations in tests is in the range of 10000 to
# 50000), write trained generative model and generated data in training steps to
# files "gm.bin" and "gd.bin".
## Not run:
gdTrain("gm.bin", "gd.bin", "ds.bin", c(1, 2), gdTrainParameters(1000))
## End(Not run)
# Read generative data from file "gd.bin", calculate density values and
# write generative data with density values to original file.
## Not run:
gdCalculateDensityValues("gd.bin")
## End(Not run)

# 2. Build a metric subspaces data model for the iris data set

# Load  library
## Not run:
library(ganDataModel)
## End(Not run)

# Read a data source and generative data from files "ds.bin" and "gd.bin",
# train a neural network which approximates density values for a data source
# in iterative training steps (used number of iterations in tests is in the
# range of 250000 to 300000), create a data model containing the trained neural
# network and write it to a file "dm.bin" in binary format.
## Not run:
dmTrain("dm.bin", "ds.bin", "gd.bin", 10000)
## End(Not run)

# Read a data model and generative data from files "dm.bin" and "gd.bin",
# build metric subspaces for level 0.7,
# add obtained metric subspaces to the data model
# and write it to original file.
## Not run:
dmBuildMetricSubspaces("dm.bin", 0.38, "gd.bin")
## End(Not run)

# Read a data model and generative data from files "dm.bin" and "gd,bin".
# Read in data is accessed in function dmPlotMetricSubspaces.
```

```
## Not run:
dmRead("dm.bin", "gd.bin")
## End(Not run)

# Create an image showing a two-dimensional projection of generative data
# contained in metric subspaces fpr level 0.38 for column indices 3, 4 and write
# it to file "ms.png".
## Not run:
dmPlotMetricSubspaces(
  list(dmPlotMetricSubspaceParameters(level = 0.38,
                                      labels = c("*"),
                                      percent = 100,
                                      boundary = TRUE,
                                      color = "red",
                                      backgroundPercent = 0,
                                      backgroundColor = "red",
                                      backgroundReset = TRUE,
                                      plotLabels = TRUE)),
  "msl.png",
  "Metric Subspaces for the Iris Dataset",
  c(3, 4),
  "ds.bin",
  dmPlotEvaluateDataSourceParameters(0.38))
## End(Not run)

# Read a data model and generative data from files "dm.bin" and "gd.bin",
# build metric subspaces for level 0.5,
# add obtained metric subspaces to the data model
# and write it to original file.
## Not run:
dmBuildMetricSubspaces("dm.bin", 0.5, "gd.bin")
## End(Not run)

# Read a data model and generative data from files "dm.bin" and "gd,bin".
# Read in data is accessed in function dmPlotMetricSubspaces.
## Not run:
dmRead("dm.bin", "gd.bin")
## End(Not run)

# Create an image showing a two-dimensional projection of generative data
# contained in metric subspaces for levels 0.38, 0.5 for column indices 3, 4
# and write it to file "msls.png".
## Not run:
dmPlotMetricSubspaces(
  list(dmPlotMetricSubspaceParameters(level = 0.38,
                                      labels = c("*"),
                                      percent = 100,
                                      boundary = TRUE,
                                      color = "red",
                                      backgroundPercent = 0,
                                      backgroundColor = "red",
                                      backgroundReset = TRUE,
                                      plotLabels = TRUE),
```

```
        dmPlotMetricSubspaceParameters(level = 0.5,
                                       labels = c("*"),
                                       percent = 100,
                                       boundary = TRUE,
                                       color = "green",
                                       backgroundPercent = 5,
                                       backgroundColor = "red",
                                       backgroundReset = TRUE,
                                       plotLabels = TRUE)),
    "msls.png",
    "Metric Subspaces for the Iris Dataset",
    c(3, 4),
    "ds.bin",
    dmPlotEvaluateDataSourceParameters(0.38))
  ## End(Not run)
```

`dmBuildMetricSubspaces` *Build metric subspaces for a level*

Description

Read a data model and generative data from files, analyze the contained neural network in the data model for a level, determine metric subspaces with density values above a level, add obtained metric subspaces to the data model and write it to original file.

Usage

```
dmBuildMetricSubspaces(dataModelFileName, level, generativeDataFileName)
```

Arguments

`dataModelFileName`	Name of data model file
`level`	Level
`generativeDataFileName`	Name of generative data file

Value

None

Examples

```
## Not run:
dmBuildMetricSubspaces("dm.bin", 0.5, "gd.bin")
## End(Not run)
```

`dmCalculateDensityValue` *Calculate a density value for a data record*

Description

Calculate a density value for a data record by evaluating the contained neural network in a data model.

Usage

```
dmCalculateDensityValue(dataRecord)
```

Arguments

`dataRecord`	List containing a data record

Value

Normalized density value

Examples

```
## Not run:
dmRead("dm.bin", "gd.bin")
dmCalculateDensityValue(list(4.4, 2.9, 1.4, 0.2))
## End(Not run)
```

`dmGetContainedInMetricSubspaces` *Get metric subspaces in which a data record is contained*

Description

Determine in which metric subspaces in a data model a data record is contained.

Usage

```
dmGetContainedInMetricSubspaces(dataRecord)
```

Arguments

`dataRecord`	List of a data record

Value

List of list containing level and label of metric subspaces

Examples

```
## Not run:
dmRead("dm.bin", "gd.bin")
dmGetContainedInMetricSubspaces(list(4.4, 2.9, 1.4, 0.2))
## End(Not run)
```

`dmGetLevels` *Get levels for metric subspaces*

Description

Get levels for metric subspaces in a data model.

Usage

```
dmGetLevels()
```

Value

Vector of levels

Examples

```
## Not run:
dmRead("dm.bin", "gd.bin")
dmGetLevels()
## End(Not run)
```

`dmGetMetricSubspaceProperties`
Get metric subspace properties for a level

Description

Get properties of metric subspaces in a data model for a level.

Usage

```
dmGetMetricSubspaceProperties(level)
```

Arguments

`level`	Level for metric subspaces

Value

List of list containing label and size of contained generative data for metric subspaces

Examples

```
## Not run:
dmRead("dm.bin", "gd.bin")
dmGetMetricSubspaceProperties(0.73)
## End(Not run)
```

`dmPlotEvaluateDataSourceParameters`
Specify plot parameters for evaluated data source

Description

Specify plot parameters for evaluated data source passed to dmPlotMetricSubspaces().

Usage

```
dmPlotEvaluateDataSourceParameters(level = 0, color = "blue")
```

Arguments

`level`	Level for evaluation
`color`	Color for data points of evaluaded data source

Value

List of plot parameters for evaluated data source

Examples

```
## Not run:
dmPlotEvaluateDataSourceParameters()
## End(Not run)
```

`dmPlotMetricSubspaceParameters`
Specify plot parameters for metric subspaces for a level

Description

Specify plot parameters for metric subspaces in a data model for a level. A list of plot parameters is created for different levels and passed to dmPlotMetricSubspaces().

Usage

```
dmPlotMetricSubspaceParameters(
  level,
  labels = c("*"),
  percent = 10,
  boundary = TRUE,
  color = "red",
  backgroundPercent = 0,
  backgroundColor = "red",
  backgroundReset = TRUE,
  plotLabels = TRUE
)
```

Arguments

level	Level for metric subspaces.
labels	Vector of labels for metric subspaces. The default vector contains the wildcard character * which includes all labels.
percent	Percent of randomly selected data points of generative data contained in metric subspaces
boundary	Boolean value indicating if only data points of metric subspace boundaries should be selected
color	Color for data points of generative data contained in metric subspaces
backgroundPercent	Percent of randomly selected data points of generative data contained in metric subspaces for background
backgroundColor	Color for data points of generative data contained in metric subspaces for background
backgroundReset	Before data points for a metric subspace are drawn reset its background.
plotLabels	Boolean value indicating if labels for metric subspaces for a level should be displayed

Value

List of plot parameters for metric subspaces

Examples

```
## Not run:
dmPlotMetricSubspaceParameters(0.5)
## End(Not run)
```

`dmPlotMetricSubspaces` *Create an image file for metric subspaces*

Description

Create an image file containing two-dimensional projections of generative data contained in metric subspaces in a data model and optionally an evaluated data source. Plot parameters are passed by a list of generated plot parameters for different levels by dmPlotMetricSubspaceParameters() and by dmPlotEvaluateDataSourceParameters(). Data points are drawn in the order generative data contained in metric subspaces by increasing level and evaluated data source.

Usage

```
dmPlotMetricSubspaces(
  plotMetricSubspaceParametersList = list(),
  imageFileName,
  title,
  columnIndices,
  evaluateDataSourceFileName = "",
  plotEvaluateDataSourceParameters = NULL
)
```

Arguments

`plotMetricSubspaceParametersList`
: List of plot parameters for metric subspaces for different levels, see dmPlotMetricSubspaceParameters().

`imageFileName`
: Name of image file

`title`
: Title of image

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

`evaluateDataSourceFileName`
: Name of evaluated data source file

`plotEvaluateDataSourceParameters`
: Plot parameters for evaluated data source, see dmPlotEvaluateDataSourceParameters().

Value

None

Examples

```
## Not run:
dmRead("dm.bin", "gd.bin")
dmPlotMetricSubspaces(
  list(dmPlotMetricSubspaceParameters(level = 0.5,
                                      labels = c("*"),
                                      percent = 100,
                                      boundary = TRUE,
                                      color = "red",
                                      backgroundPercent = 0,
                                      backgroundColor = "red",
                                      backgroundReset = TRUE,
                                      plotLabels = TRUE)),
  "ms.png",
  "Metric Subspaces for the Iris Dataset",
  c(3, 4),
  "ds.bin",
  dmPlotEvaluateDataSourceParameters(0.67))
## End(Not run)
```

---

dmRead *Read a data model and generative data*

Description

Read a data model and generative data from files. This function has to be called before calling API functions when file names for a data model and gernerative data are not passed to functions directly.

Usage

```
dmRead(dataModelFileName, generativeDataFileName)
```

Arguments

dataModelFileName	Name of data model file
generativeDataFileName	Name of generative data file

Value

None

Examples

```
## Not run:
dmRead("dm.bin", "gd.bin")
## End(Not run)
```

dmRemoveMetricSubspaces *Remove metric subspaces for a level*

Description

Read a data model from file, remove metric subspaces in the data model for a level and write it to original file.

Usage

```
dmRemoveMetricSubspaces(dataModelFileName, level)
```

Arguments

dataModelFileName	Name of data model file
level	Level

Value

None

Examples

```
## Not run:
dmRemoveMetricSubspaces("dm.bin", 0.5)
## End(Not run)
```

dmReset *Reset API*

Description

Reset API

Usage

```
dmReset()
```

Value

None

Examples

```
## Not run:
dmReset()
## End(Not run)
```

dmTrain *Train a neural network which approximates density values for a data source*

Description

Read a data source and generative data from files, train a neural network which approximates density values for a data source in iterative training steps, create a data model containing the trained neural network and write it to a file in binary format.

Usage

```
dmTrain(
  dataModelFileName,
  dataSourceFileName,
  generativeDataFileName,
  numberOfIterations,
  numberOfHiddenLayerUnits = 512
)
```

Arguments

dataModelFileName	Name of data model file
dataSourceFileName	Name of data source file
generativeDataFileName	Name of generative data file
numberOfIterations	Number of iterations.
numberOfHiddenLayerUnits	Number of hidden layer units

Value

None

Examples

```
## Not run:
dmTrain("dm.bin", "ds.bin", "gd.bin", 10000)
## End(Not run)
```

Index