Package 'divnn'

March 4, 2021

Type Package

Title An implementation of the DeepInsight Visible Neural Network

Version 0.1.3

Date 2020-11-10

Depends R (>= 4.0.2)

Description This package facilitates application of DeepInsight (DI) and Visible Neural Network (VNN) algorithms from Alok Sharma and Michael Ku Yu, respectively. The application is intended for supervised machine learning by convolutional neural network (CNN). DeepInsight converts non-image data into image-like data by dimensionality reduction algorithms. This package maps the data into a multi-dimensional array. Meanwhile, VNN determines a neural network architecture by hierarchical clustering algorithms, particularly for data-driven ontology. This package generate a CNN model based on the ontology using the DeepInsight array as the input. However, this package includes neither dimensionality reduction nor data-driven ontology inference. A comprehensive guide to orchestrate this package and other packages to develop the DI-VNN model is described in this package vignette. The inputs are instance-feature value data frame, outcome vector, feature similarity matrix, feature three-dimensional mapping matrix, and ontology source-target-similarity-relation data frame. The outputs are tidy (expression) set, training array, and Keras CNN model.

License GPL-3

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports tidyverse,
BiocGenerics,
Biobase,
pbapply,
matrixStats,
tensorflow,
keras,
igraph

Suggests BiocStyle,
knitr,
rmarkdown,
kableExtra,

magick,
BiocManager,
doParallel,
WGCNA,
preprocessCore,
limma,
Rtsne,
reticulate,
zeallot,
igraph,
ParBayesianOptimization,
testthat

URL https://github.com/herdiantrisufriyana/divnn

BugReports https://github.com/herdiantrisufriyana/divnn/issues

VignetteBuilder knitr

R topics documented:

`generator.ontoarray` *Make an ontoarray generator for visible neural network (VNN) modeling*

Description

This function creates a function that generate a batch of ontoarray for training or testing a Keras Convolutional Neural Network (CNN) model using `fit_generator`, `evaluate_generator`, or `predict_generator` function from Keras R package.

Usage

```
generator.ontoarray(tidy_set, index, batch_size, sample_weights = NULL)
```

Arguments

`tidy_set`	TidySet, an ExpressionSet with three tables.
`index`	An integer vector of index to select which ontoarray will be used for training or testing.

batch_size	An integer of how much samples are generated everytime this function runs. If all samples are generated,this function will loop over the samples.
sample_weights	An numeric vector for weighting the loss function. This is useful to pay more losses for minority class.

Value

output sample generator, a function for argument of generator in fit_generator, evaluate_generator, or predict_generator function from Keras R package.

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )

## Create ontonet (Keras model object) generator function
ontonet=generator.ontonet(tidy_set)

## Randomize sample and split indices for train and test set
set.seed(33)
index=sample(1:dim(tidy_set)[2],dim(tidy_set)[2],F)
test_i=1:round(0.2*length(index))
train_i=!index %in% index[test_i]

## Fit the model
history=
  ontonet %>%
  compile(
    loss='mean_squared_error'
    ,loss_weights=c(rep(0.3,length(.$outputs)-1),1)
    ,metrics='accuracy'
  ) %>%
  fit_generator(
    generator=
      generator.ontoarray(
        tidy_set
        ,index=index[train_i]
        ,batch_size=4
      )
    ,steps_per_epoch=24
    ,validation_data=
      generator.ontoarray(
        tidy_set
        ,index=index[test_i]
        ,batch_size=4
```

```
    )
  ,validation_steps=6
  ,epochs=30
  ,verbose=1
)
```

`generator.ontonet` *Make an ontonet generator for visible neural network (VNN) modeling*

Description

This function creates a function that generate a Keras Convolutional Neural Network (CNN) model with a specific layer architecture for each path in the hierarchy of the given ontology.

Usage

```
generator.ontonet(
  tidy_set,
  path = NULL,
  init_seed = 888,
  init2_seed = 9999,
  l2_norm = 0,
  output_unit = 1,
  output_activation = "sigmoid"
)
```

Arguments

`tidy_set`	TidySet, an ExpressionSet with three tables.
`path`	A character of file path if the model json file is saved.
`init_seed`	An integer of random seed for ReLU initializer.
`init2_seed`	An integer of random seed for tanh initializer.
`l2_norm`	A numeric of L2-norm regularization factor.
`output_unit`	An integer of how many node for every output layer.
`output_activation`	A character of activation function name for all nodes in every output layer, i.e. sigmoid, softmax, tanh, relu, exponential, softplus, softsign, selu, elu. If linear, set this value as NULL.

Value

output Keras model object, a pointer to Keras model object in python environment, which will be an input to train VNN model using Keras R package.

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )

## Create ontonet (Keras model object) generator function
ontonet=generator.ontonet(tidy_set)
```

`TidySet.compile` *Make a TidySet for visible neural network (VNN) modeling*

Description

This function create a TidySet, an ExpressionSet class to orchestrate five data into a single set of three tables.

Usage

```
TidySet.compile(
  value,
  outcome,
  similarity,
  mapping,
  ontology,
  ranked = T,
  dims = 7,
  decreasing = F,
  seed_num = 33
)
```

Arguments

`value`	Instance-feature value, a data frame with rows for instances and columns for features. All rows in `value` should have names. All values should be numerics.
`outcome`	Outcome, a vector of binary integers with the same length as the instances. The length and the order of `outcome` should be the same with those of `value`. Value of 0 and 1 should refer to non-event and event outcome, respectively.
`similarity`	Feature similarity, a square matrix of numerics containing feature-feature similarity measures.
`mapping`	Feature three-dimensional mapping, a matrix of integers with rows for features and three columns for three dimensions where the features are mapped onto.

`ontology` Ontology, a data frame with rows for ontologies and four columns for source, target, similarity, and relation. Feature (source)- ontology (target) relation should be annotated as 'feature', while ontology- ontology relation should be annotated as 'is_a'. To differentiate between feature and ontology names, a prefix of 'ONT:' precedes an ontology name. All columns except similarity in `ontology` should be characters. Similarity (a numeric) is a minimum threshold by which either features or ontologies (source) belong to an ontology (target).

Value

output TidySet, an ExpressionSet with three tables. Instance-feature value data frame and outcome vector are compiled as a phenotype data frame with rows for instances and columns for features and outcome. Instance- feature value data frame and feature three-dimensional mapping matrix are compiled as an expression matrix with rows for positions of features and columns for instances. The mapping and similarity matrices and ontology data frame are compiled as a feature data frame with rows for positions of features and columns for feature names and ontological relations. For easier, access the similarity matrix, ontomap four-dimensional array, ontotype list of two-dimensional matrices, and ontology data frame are included in experiment notes that can be called using Biobase function `notes`.

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )

## The TidySet
tidy_set

## The phenotype data frame
pData(tidy_set)

## The feature data frame
fData(tidy_set)

## The expression data frame
exprs(tidy_set)

## Recall a similarity matrix
notes(tidy_set)$similarity

## Recall an ontomap four-dimensional array
notes(tidy_set)$ontomap

## Recall an ontotype list of two-dimensional matrices
notes(tidy_set)$ontotype
```

```
## Recall an ontology data frame
notes(tidy_set)$ontology
```

`TidySet.read` *Read a .ts.tar.gz file to a TidySet*

Description

This function read multiple files archived by tar with gzip compression to a TidySet.

Usage

```
TidySet.read(path)
```

Arguments

`path`	A character of .ts.tar.gz file path (include file extension).

Value

output A TidySet, an ExpressionSet with three tables. Function of `TidySet.write` can write this file from the TidySet.

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )

## Write a .ts.tar.gz file from a TidySet
TidySet.write(tidy_set,'example')

## Read a .ts.tar.gz file to a TidySet
TidySet.read('example.ts.tar.gz')
```

TidySet.write *Write a .ts.tar.gz file from a TidySet*

Description

This function write multiple files archived by tar with gzip compression from a TidySet.

Usage

```
TidySet.write(tidy_set, path)
```

Arguments

tidy_set	TidySet, an ExpressionSet with three tables.
path	A character of .ts.tar.gz file path (do not include file extension).

Value

output A .ts.tar.gz file containing exprs.csv, pData.csv, fData.csv, similarity.csv, ontology.csv, and others.txt. Function of `TidySet.read` can read this file back to a TidySet.

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )

## Write a .ts.tar.gz file from a TidySet
write_tidy_set(tidy_set,'example')
```

utils.example *Make an input example for divnn package*

Description

This function create an input example for several function in divnn package.

Usage

```
utils.example()
```

Value

output A list of inputs: 1) value, a data frame with rows for instances and columns for features; 2) outcome, a vector of binary integers with the same length as the instances; 3) similarity, a square matrix of numerics containing feature-feature similarity measures; 4) mapping, a matrix of numerics with rows for features and three columns for three dimensions where the features are mapped onto; and 5) ontology, a data frame with rows for ontologies and four columns for source, target, similarity, and relation. In addition, a result of hierarchical clustering is also included for visualization purpose.

Examples

```
## Create input example
input=utils.example()

## Show output and visualize the ontology by hierarchical clustering
input
plot(input$hierarchy)
```

---

viz.ontoarray *Visualize an ontoarray for visible neural network (VNN) modeling*

Description

This function visualizes an ontoarray using the average values representing all ontoarrays. The average value of event is substracted by that of non-event. The visualization demonstrates a representation layer per ontology as relative values between event and non-event.

Usage

```
viz.ontoarray(tidy_set, ontonet, batch_size = 32, verbose = T)
```

Arguments

tidy_set	TidySet, an ExpressionSet with three tables.
ontonet	A Keras model object, a pointer to Keras model object in python environment, which will be an input to train VNN model using Keras R package.
batch_size	An integer of how much samples are generated everytime this function runs to get the output of the representation layers. If all samples are generated, this function will loop over the samples. But, only the same sample size will be generated.
verbose	Verbosity, a logical indicating whether progress should be shown.
pal	A vector of two characters for colors representing the minimum and maximum values of the metric.
label	A logical whether feature label are shown.
label.family	A character of font family for feature label.
label.size	A numeric of font size for feature label.
label.color	A character of font color for feature label.
grid_col	A integer of the column number of grid for showing for ontoarrays for all ontologies per channel (z).

Value

viz.ontoarray object, a list of ontoarrays containing differences of summarized values between event and non-event. For each ontology, there are an output of a layer representing the ontology, and an ontotype information about features grouped within the ontology. Visualization are shown by `plot()`.

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )
## Create ontonet (Keras model object) generator function
ontonet=generator.ontonet(tidy_set)

## Visualize ontoarray
viz.ontoarray(tidy_set,ontonet)
```

`viz.ontonet` *Visualize an ontonet for visible neural network (VNN) modeling*

Description

This function visualizes an ontonet, including the evaluation results for each ontology.

Usage

```
viz.ontonet(
  tidy_set,
  feature = F,
  eval.results = NULL,
  eval.metric = "loss",
  eval.pal = c("darkred", "darkgreen"),
  eval.gradient = 100
)
```

Arguments

`tidy_set`	TidySet, an ExpressionSet with three tables.
`feature`	A logical whether feature nodes are shown; otherwise, only ontology nodes are shown.
`eval.results`	A list of resampled results or a single result from `evaluate_generator()` of `keras` package.

eval.metric	A character of metric name to visualize.
eval.pal	A vector of two characters for colors representing the minimum and maximum values of the metric.
eval.gradient	An integer of the number of possible colors between ones for the minimum and maximum values of the metric.
node.color	A character of color for bordering node.
node.fill	A character of color for filling node.
node.shape	A character of node shape. Run vertex.shapes() from igraph package to find the options.
node.size	A numeric of node size.
label	A logical whether the nodes are labeled.
label.family	A character of font family for node label.
label.cex	A numeric of font size for node label.
label.color	A character of font color for node label.
edge.color	A character of color for edge.
seed_num	A integer of random seed for network construction algorithm.

Value

viz.ontonet object, a list of node and edge data frames with the visualization parameters. Visualization are shown by plot().

Examples

```
## Create input example
input=utils.example()

## Compile input to a TidySet
tidy_set=
  TidySet.compile(
    value=input$value
    ,outcome=input$outcome
    ,similarity=input$similarity
    ,mapping=input$mapping
    ,ontology=input$ontology
  )

## Visualize ontonet
viz.ontonet(tidy_set)
```

Index