Package ‘LSTMfactors’

July 7, 2025

Type Package

Title Determining the Number of Factors in Exploratory Factor Analysis by LSTM

Version 1.0.0

Date 2025-06-25

Author Haijiang Qin [aut, cre, cph] (ORCID: <https://orcid.org/0009-0000-6721-5653>), Lei Guo [aut, cph] (ORCID: <https://orcid.org/0000-0002-8273-3587>)

Maintainer Haijiang Qin <haijiang133@outlook.com>

Description
A method for factor retention using a pre-trained Long Short Term Memory (LSTM) Network, which is originally developed by
Hochreiter and Schmidhuber (1997) <doi:10.1162/neco.1997.9.8.1735>, is provided.
The sample size of the dataset used to train the LSTM model is 1,000,000.
Each sample is a batch of simulated response data with a specific latent factor structure.
The eigenvalues of these response data will be used as sequential data to train the LSTM.
The pre-trained LSTM is capable of factor retention for real response data with a
true latent factor number ranging from 1 to 10, that is, determining the number of factors.

License GPL-3

Depends R (>= 4.3.0)

Imports reticulate, EFAfactors

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Collate 'af.softmax.R' 'check_python_libraries.R' 'data.DAPCS.R' 'data.datasets.LSTM.R' 'data.scaler.LSTM.R' 'LSTM.R' 'extractor.feature.R' 'load.R' 'normalizor.R' 'plot.R' 'print.R' 'utils.R' 'zzz.R'

Repository CRAN

URL https://haijiangqin.com/LSTMfactors/

Date/Publication 2025-07-07 12:50:12 UTC

Contents

`af.softmax` *An Activation Function: Softmax*

Description

This function computes the softmax of a numeric vector. The softmax function transforms a vector of real values into a probability distribution, where each element is between 0 and 1 and the sum of all elements is 1. @seealso `LSTM`

Usage

```
af.softmax(x)
```

Arguments

`x`	A numeric vector for which the softmax transformation is to be computed.

Details

The softmax function is calculated as:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

In the case of overflow (i.e., when $exp(x_i)$ is too large), this function handles `Inf` values by assigning `1` to the corresponding positions and `0` to the others before Softmax. @seealso LSTM

Value

A numeric vector representing the softmax-transformed values of `x`.

Examples

```
x <- c(1, 2, 3)
af.softmax(x)
```

`check_python_libraries` *Check and Install Python Libraries (numpy and onnxruntime)*

Description

This function checks whether the Python (suggested >= 3.11) libraries ‘numpy‘ and ‘onnxruntime‘ are installed. If not, it will prompt the user to decide whether to install them. If the user chooses ’y’, the required library will be installed using the ‘reticulate‘ package. If the user chooses ’n’, the installation will be skipped. @seealso LSTM

Usage

```
check_python_libraries()
```

Value

A list indicating whether ‘numpy‘ and ‘onnxruntime‘ are installed. The list contains the following logical elements:

`numpy_installed`	TRUE if ‘numpy‘ is installed, FALSE otherwise.
`onnxruntime_installed`	TRUE if ‘onnxruntime‘ is installed, FALSE otherwise.

`data.DAPCS` *20-item Dependency-Oriented and Achievement-Oriented Psychological Control Scale (DAPCS)*

Description

This dataset contains responses to a 20-item Dependency-Oriented and Achievement-Oriented Psychological Control Scale (DAPCS), measuring four distinct factors of psychological control perceived by adolescents from their parents.

Details

The data were collected in 2022 from a sample of 987 general high school students in China. Among the participants, 406 were male and 581 were female, with a mean age of 15.823 years (SD = 0.793).

The DAPCS scale was developed by Soenens and Vansteenkiste (2010). It consists of 20 items that are grouped into four distinct dimensions, each with demonstrated internal consistency:

- **Autonomy – Negative Reaction**: Measures the extent of negative parental responses to adolescents' autonomy. *Reliability:* Cronbach's $\alpha = 0.857$
- **Dependence – Positive Reaction**: Measures the extent of positive parental responses to adolescents' dependence. *Reliability:* Cronbach's $\alpha = 0.817$
- **Low Achievement – Negative Reaction**: Measures the extent of negative parental responses to adolescents' low academic achievement. *Reliability:* Cronbach's $\alpha = 0.885$
- **High Achievement – Positive Reaction**: Measures the extent of positive parental responses to adolescents' high academic achievement. *Reliability:* Cronbach's $\alpha = 0.889$

The scale contains 20 items rated on a 5-point Likert scale, ranging from 1 = strongly disagree to 5 = strongly agree. In the dataset in this `EFAfactors` package, the total scale demonstrated a Cronbach's α of 0.923, and the four subscales showed Cronbach's α ranging from 0.817 to 0.889, indicating good reliability.

References

Soenens, B., & Vansteenkiste, M. (2010). A theoretical upgrade of the concept of parental psychological control: Proposing new insights on the basis of self-determination theory. Developmental Review, 30(1), 74–99.

Examples

```
data(data.DAPCS)
response <- data.DAPCS[, -c(1, 2)]
head(response)
```

`data.datasets.LSTM` *Subset Dataset for Training the Pre-Trained Long Short Term Memory (LSTM) Network*

Description

This dataset is a subset of the full datasets, consisting of 1,000 samples from the original 1,000,000-sample datasets.

Format

A 1,000×21 matrix, where the firt column represents the labels and the last 20 columns represent feature values, which correspond to the number of factors associated with the features.

Note

Methods for generating and extracting features from the dataset can be found in LSTM.

See Also

LSTM, load.scaler, data.scaler.LSTM, normalizor

Examples

```
data(data.datasets.LSTM)
head(data.datasets.LSTM)
```

`data.scaler.LSTM` *the Scaler for the Pre-Trained Long Short Term Memory (LSTM) Network*

Description

This dataset contains the means and standard deviations of the 1,000,000 datasets for training the Long Short Term Memory (LSTM) Network, which can be used to determine the number of factors.

Format

A `list` containing two `vectors`, each of length 20:

means A numeric vector representing the means of the 20 features extracted from the 1,000,000 datasets.

sds A numeric vector representing the standard deviations of the 20 features extracted from the 1,000,000 datasets.

See Also

LSTM, load.scaler, data.datasets.LSTM, normalizor

Examples

```
data(data.scaler.LSTM)
print(data.scaler.LSTM)

data.scaler <- load.scaler()
print(data.scaler)
```

`extractor.feature` *Extracting features for the pre-trained Long Short Term Memory (LSTM) Network*

Description

This function is used to extract the features required by the pre-trained Long Short Term Memory (LSTM) Network. @seealso LSTM

Usage

```
extractor.feature(
  response,
  cor.type = "pearson",
  use = "pairwise.complete.obs"
)
```

Arguments

`response`	A required `N` × `I` matrix or data.frame consisting of the responses of `N` individuals to `I` items.
`cor.type`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
`use`	An optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.

Details

For `"LSTM"`, a total of 2 types of features. These features are as follows:

(1) The top 10 largest eigenvalues.

(2) The difference of the top 10 largest eigenvalues to the corresponding reference eigenvalues from arallel Analysis (PA). @seealso PA

Value

A matrix (1×20) containing all the features for the LSTM.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

See Also

LSTM

Examples

```
library(LSTMfactors)
set.seed(123)

##Take the data.DAPCS dataset as an example.
data(data.DAPCS)

response <- as.matrix(data.DAPCS[, 3:22]) ## loading data


## Run extractor.feature function
features <- extractor.feature(response)

print(features)
```

`load.LSTM` *Load the pre-trained Long Short Term Memory (LSTM) Network*

Description

Loads the pre-trained Long Short Term Memory (LSTM) Network form `LSTM.onnx`. The function uses the `reticulate` package to import the `onnxruntime` Python library and create an inference session for the model.

Usage

```
load.LSTM()
```

Value

An ONNX runtime inference session object for the LSTM model.

Note

Note that Python (suggested >= 3.11) and the libraries `numpy` and `onnxruntime` are required.

First, please ensure that Python is installed on your computer and that Python is included in the system's PATH environment variable. If not, please download and install it from the official website (https://www.python.org/).

If you encounter an error when running this function stating that the `numpy` and `onnxruntime` modules are missing:

`Error in py_module_import(module, convert = convert) :`

`ModuleNotFoundError: No module named 'numpy'` or

`Error in py_module_import(module, convert = convert) :`

`ModuleNotFoundError: No module named 'onnxruntime'`

this means that the `numpy` or `onnxruntime` library is missing from your Python environment. If you are using Windows or macOS, please run the command `pip install numpy` or `pip install onnxruntime` in Command Prompt or Windows PowerShell (Windows), or Terminal (macOS). If you are using Linux, please ensure that `pip` is installed and use the command `pip install numpy` or `pip install onnxruntime` to install the missing libraries.

See Also

`LSTM`

`load.scaler` *Load the Scaler for the pre-trained Long Short Term Memory (LSTM) Network*

Description

Loads the scaler object within the `LSTMfactors` package. This object is a `list` containing a mean vector and a standard deviation vector, which were computed from the 1,000,000 datasets `data.datasets.LSTM` training the Long Short Term Memory (LSTM) Network. It serves as a tool for normalizing features in `LSTM`.

Usage

```
load.scaler()
```

Value

scaler objective.

See Also

`LSTM`, `normalizor`, `data.scaler.LSTM`

Examples

```
library(LSTMfactors)

scaler <- load.scaler()
print(scaler)
```

LSTM — *A pre-trained Long Short Term Memory (LSTM) Network for Determining the Number of Factors*

Description

This function will invoke a pre-trained Long Short Term Memory (LSTM) Network that can reliably perform the task of determining the number of factors. The maximum number of factors that the network can discuss is 10. The LSTM model is implemented in Python and trained on PyTorch (https://pytorch.org/) with CUDA 12.6 for acceleration. After training, the LSTM were saved as `LSTM.onnx` file. The `LSTM` function performs inference by loading the `LSTM.onnx` file in both Python and R environments. Therefore, please note that Python (suggested >= 3.11) and the libraries `numpy` and `onnxruntime` are required. @seealso `check_python_libraries`

To run this function, Python (suggested >= 3.11) is required, along with the installation of `numpy` and `onnxruntime`. See more in Details and Note.

Usage

```
LSTM(
  response,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

`response`	A required $N \times I$ matrix or data.frame consisting of the responses of N individuals to I items.
`cor.type`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso `cor`.
`use`	An optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso `cor`.
`vis`	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
`plot`	A Boolean variable that will print the NN plot when set to TRUE, and will not print it when set to FALSE. (Default = TRUE)

Details

A total of 1,000,000 datasets (`data.datasets.LSTM`) were simulated to extract features for training LSTM. Each dataset was generated following the methods described by Auerswald & Moshagen (2019) and Goretzko & Buhner (2020), with the following specifications:

- Factor number: *F* ~ U[1,10]
- Sample size: *N* ~ U[100,1000]
- Number of variables per factor: *vpf* ~ [3,10]
- Factor correlation: *fc* ~ U[0.0,0.5]
- Primary loadings: *pl* ~ U[0.35,0.80]
- Cross-loadings: *cl* ~ U[-0.2,0.2]

A population correlation matrix was created for each data set based on the following decomposition:

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T + \mathbf{\Delta}$$

where $\mathbf{\Lambda}$ is the loading matrix, $\mathbf{\Phi}$ is the factor correlation matrix, and $\mathbf{\Delta}$ is a diagonal matrix, with $\mathbf{\Delta} = 1 - \text{diag}(\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T)$. The purpose of $\mathbf{\Delta}$ is to ensure that the diagonal elements of $\mathbf{\Sigma}$ are 1.

The response data for each subject was simulated using the following formula:

$$X_i = L_i + \epsilon_i, \quad 1 \leq i \leq I$$

where L_i follows a normal distribution $N(0, \sigma)$, representing the contribution of latent factors, and ϵ_i is the residual term following a standard normal distribution. L_i and ϵ_i are uncorrelated, and ϵ_i and ϵ_j are also uncorrelated.

For each simulated dataset, a total of 2 types of features (@seealso `extractor.feature`). These features are as follows:

(1) The top 10 largest eigenvalues.

(2) The difference of the top 10 largest eigenvalues to the corresponding reference eigenvalues from arallel Analysis (PA). @seealso PA

The two types of features above were treated as sequence data with a time step of 10 to train the LSTM model, resulting in a final classification accuracy of 0.847.

The LSTM model is implemented in Python and trained on PyTorch (https://download.pytorch.org/whl/cu126) with CUDA 12.6 for acceleration. After training, the LSTM was saved as a `LSTM.onnx` file. The `NN` function performs inference by loading the `LSTM.onnx` file in both Python and R environments.

Value

An object of class `LSTM` is a `list` containing the following components:

`nfact`	The number of factors to be retained.
`features`	A matrix (1×20) containing all the features for determining the number of factors by the LSTM.
`probability`	A matrix containing the probabilities for factor numbers ranging from 1 to 10 (1x10), where the number in the f-th column represents the probability that the number of factors for the response is f.

Note

Note that Python (suggested >= 3.11) and the libraries `numpy` and `onnxruntime` are required.

First, please ensure that Python is installed on your computer and that Python is included in the system's PATH environment variable. If not, please download and install it from the official website (https://www.python.org/).

If you encounter an error when running this function stating that the `numpy` and `onnxruntime` modules are missing:

`Error in py_module_import(module, convert = convert) :`

`ModuleNotFoundError: No module named 'numpy'`

or

`Error in py_module_import(module, convert = convert) :`

`ModuleNotFoundError: No module named 'onnxruntime'`

this means that the `numpy` or `onnxruntime` library is missing from your Python environment. The `check_python_libraries` function can help you install these two dependency libraries.

Of course, you can also choose not to use the `check_python_libraries` function. You can directly install the `numpy` or `onnxruntime` library using the appropriate commands. If you are using Windows or macOS, please run the command `pip install numpy` or `pip install onnxruntime` in Command Prompt or Windows PowerShell (Windows), or Terminal (macOS). If you are using Linux, please ensure that `pip` is installed and use the command `pip install numpy` or `pip install onnxruntime` to install the missing libraries.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/10.1037/met0000200.

Goretzko, D., & Buhner, M. (2020). One model to rule them all? Using machine learning algorithms to determine the number of factors in exploratory factor analysis. Psychol Methods, 25(6), 776-786. https://doi.org/10.1037/met0000262.

`normalizor` *Feature Normalization*

Description

This function normalizes a matrix of features using precomputed means and standard deviations. The function automatically runs `load.scaler` to read the standard deviations and means of the features, which are organized into a `list` object named `data.scaler.LSTM`. These means and standard deviations are computed from the 1,000,000 datasets `data.datasets.LSTM` for training the pre-trained Long Short Term Memory (LSTM) Network.

Usage

```
normalizor(features)
```

Arguments

features	A numeric matrix where each row represents an observation and each column represents a feature.

Details

The function applies z-score normalization to each element in the `features` matrix. It uses the `scaler` object, which is expected to contain precomputed means and standard deviations for each feature. The normalized value for each element is computed as:

$$z = \frac{x - \mu}{\sigma}$$

where x is the original value, μ is the mean, and σ is the standard deviation.

Value

A matrix of the same dimensions as `features`, where each feature has been normalized.

See Also

`LSTM`, `load.scaler`, `data.datasets.LSTM`, `data.scaler.LSTM`

Examples

```
library(LSTMfactors)
set.seed(123)

##Take the data.DAPCS dataset as an example.
data(data.DAPCS)

response <- as.matrix(data.DAPCS[, 3:22]) ## loading data


## Run extractor.feature function
features <- extractor.feature(response)

features.nor <- normalizor(features)
print(features.nor)
```

`plot.LSTM` *Plot LSTM Classification Probability Distribution*

Description

This function generates a bar plot of the classification probabilities predicted by the pre-trained LSTM for determining the number of factors. The plot displays the probability distribution across different numbers of factors, with each bar representing the probability for a specific number of factors. The maximum number of factors that the network can evaluate is 10. The function also annotates each bar with its probability value.

Usage

```
## S3 method for class 'LSTM'
plot(x, ...)
```

Arguments

`x`	An object of class `LSTM`, representing the results to be plotted.
`...`	Additional arguments to be passed to the plotting function.

Value

None. This function is used for side effects (plotting).

See Also

`LSTM`

`print.LSTM` *Print LSTM Results*

Description

This function prints the number of factors suggested by the LSTM.

Usage

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

Arguments

`x`	An object of class `LSTM`, representing the results to be printed.
`...`	Additional arguments to be passed to the plotting function.

Value

None. This function is used for side effects (printing).

See Also

`LSTM`

Index