Package: tensorsem (via r-universe)

August 15, 2024

Type Package

Title Estimate structural equation models using computation graphs

Version 2.1.0

Author Erik-Jan van Kesteren

Maintainer Erik-Jan van Kesteren <e.vankesteren1@uu.nl>

Description Use lavaan code to create structural equation models, use torch to estimate them. This package provides the interface between lavaan and torch.

License GPL-3

Encoding UTF-8

LazyData true

Imports R6

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Depends torch, lavaan

Suggests rmarkdown, knitr, ggplot2

VignetteBuilder knitr

Repository https://vankesteren.r-universe.dev

RemoteUrl https://github.com/vankesteren/tensorsem

RemoteRef HEAD

RemoteSha d0471473c1c25759f52f9aad562fb67de2e4f62f

Contents

`df_to_tensor` *Prepare data for tensorsem model*

Description

This function prepares a dataframe for a tensorsem model. It first converts the variables to a design matrix, then centers it, and lastly converts it to a torch_tensor

Usage

```
df_to_tensor(df, dtype = NULL, device = NULL)
```

Arguments

`df`	data frame
`dtype`	data type of the resulting tensor
`device`	device to store the resulting tensor on

Value

Torch tensor of scaled and processed data

See Also

`torch::torch_tensor()`, `stats::model.matrix()`

`lav_mod_to_torch_opts` *Create a torch options list from a lavaan Model class.*

Description

Create a torch options list from a lavaan Model class.

Usage

```
lav_mod_to_torch_opts(lav_mod)
```

Arguments

`lav_mod`	lavaan Model class object

Value

list of tensorsem options

`mvn_negloglik` *Multivariate normal negative log-likelihood loss function for tensorsem nn module.*

Description

Multivariate normal negative log-likelihood loss function for tensorsem nn module.

Usage

```
mvn_negloglik(dat, Sigma)
```

Arguments

`dat`	The centered dataset as a tensor
`Sigma`	The model() implied covariance matrix

Value

torch_tensor: scalar negative log likelihood

`partable_from_torch` *Create a lavaan parameter table from torch free_params output*

Description

See examples in tensorsem for how to save the output.

Usage

```
partable_from_torch(pars, syntax)
```

Arguments

`pars`	data frame of parameter estimates (est) and their standard errors (se)
`syntax`	syntax of the original model

sem_fitfun *SEM fitting function*

Description

SEM fitting function

Usage

```
sem_fitfun(S, Sigma)
```

Arguments

S	The observed covariance matrix
Sigma	The model implied covariance matrix

Value

torch_tensor: scalar loss function

syntax_to_torch_opts *Create a torch options list from lavaan syntax.*

Description

Create a torch options list from lavaan syntax.

Usage

```
syntax_to_torch_opts(syntax)
```

Arguments

syntax	lavaan syntax

Value

list of tensorsem options

torch_jacobian *Compute jacobian of output wrt input tensor*

Description

Compute jacobian of output wrt input tensor

Usage

```
torch_jacobian(output, input)
```

Arguments

output	Tensor vector of size Po
input	Tensor vector of size Pi

Value

jacobian: Tensor of size Pi, Po

torch_sem *Structural equation model with a Torch backend*

Description

Function for creating a structural equation model

Usage

```
torch_sem(syntax, dtype = torch_float32(), device = torch_device("cpu"))
```

Arguments

syntax	lavaan syntax for the SEM model
dtype	(optional) torch dtype for the model (default torch_float32())
device	(optional) device type to put the model on. see torch::torch_device()

Details

This function instantiates a torch object for computing the model-implied covariance matrix based on a structural equation model. Through torch, gradients of this forward model can then be computed using backpropagation, and the parameters can be optimized using gradient-based optimization routines from the torch package.

Because of this, it is easy to add additional penalties to the standard objective function, or to write a new objective function altogether.

Value

A `torch_sem` object, which is an `nn_module` (torch object)

Fields

`free_params` Vector of free parameters

Methods

`$initialize()`:

The initialize method. Don't use this, just use `torch_sem()`

Arguments:

- `syntax` lavaan syntax for the SEM model
- `dtype` (optional) torch dtype for the model (default torch_float32())
- `device` (optional) device type to put the model on. see `torch::torch_device()`

Value:
A `torch_sem` object, which is an `nn_module` (torch object)

`$forward()`:

Compute the model-implied covariance matrix. Don't use this; `nn_modules` are callable, so access this method by calling the object itself as a function, e.g., `my_torch_sem()`. In the forward pass, we apply constraints to the parameter vector, and we create matrix views from it to compute the model-implied covariance matrix.

Value:
A `torch_tensor` of the model-implied covariance matrix

`$inverse_Hessian(loss)`:

Compute and return the asymptotic covariance matrix of the parameters with respect to the loss function

Arguments:

- `loss` torch_tensor of freshly computed loss function (needed by torch for backwards pass)

Value:
A `torch_tensor`, representing the ACOV of the free parameters

`$standard_errors(loss)`:

Compute and return observed information standard errors of the parameters, assuming the loss function is the likelihood and the current estimates are ML estimates.

Arguments:

- `loss` torch_tensor of freshly computed loss function (needed by torch for backwards pass)

Value:
A `numeric vector` of standard errors of the free parameters

`$partable(loss)`:

Create a lavaan-like parameter table from the current parameter estimates in the torch_sem object.

Arguments:

- `loss` (optional) torch_tensor of freshly computed loss function (needed by torch for backwards pass)

Value:
lavaan partable object

`$fit(dat, lrate, maxit, verbose, tol)`:
Fit a torch_sem model using the default maximum likelihood objective. This function uses the Adam optimizer to estimate the parameters of a torch_sem

Arguments:

- `dat` dataset (centered!) as a `torch_tensor`
- `lrate` learning rate of the Adam optimizer.
- `maxit` maximum number of epochs to train the model
- `verbose` whether to print progress to the console
- `tol` parameter change tolerance for stopping training

Value:
Self, i.e., the `torch_sem` object with updated parameters

`$loglik(dat)`:
Multivariate normal log-likelihood of the data.

Arguments:

- `dat` dataset (centered!) as a `torch_tensor`

Value:
Log-likelihood value (torch scalar)

See Also

`df_to_tensor()`

`torch_vech` *Half-vectorization of square matrices*

Description

Half-vectorization of square matrices

Usage

```
torch_vech(x)
```

Arguments

`x`	square (symmetric) matrix tensor

Value

column vector of stacked lower-diagonal elements

vech_dup_idx *Constructs index vector for transforming a vech vector into a vec vector to create an n*n symmetric matrix from the vech vector. tensor$index_select(0, idx)$view(3,3)*

Description

Constructs index vector for transforming a vech vector into a vec vector to create an n*n symmetric matrix from the vech vector. tensor$index_select(0, idx)$view(3,3)

Usage

```
vech_dup_idx(n)
```

Arguments

n	size of the resulting square matrix

Value

array containing the indices

Index