Package 'criticality'

October 12, 2022

Title Modeling Fissile Material Operations in Nuclear Facilities

Version 0.9.2

Depends R (>= 3.5.0)

Description A collection of functions for modeling fissile material operations in nuclear facilities, based on Zywiec et al (2021) <doi:10.1016/j.ress.2020.107322>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

Imports bnlearn, caret, dplyr, evd, fitdistrplus, ggplot2, keras, magrittr, parallel, reticulate, scales

NeedsCompilation no

Author Will Zywiec [aut, cre] (<https://orcid.org/0000-0002-1842-9599>)

Maintainer Will Zywiec <willzywiec@gmail.com>

Repository CRAN

Date/Publication 2022-04-20 17:52:29 UTC

R topics documented:

BN *BN Function*

Description

This function creates a Bayesian network from pre-formatted nuclear facility data.

Usage

```
BN(dist = "gamma", facility.data, ext.dir)
```

Arguments

dist	Truncated probability distribution (e.g., "gamma", "normal")
facility.data	.csv file name
ext.dir	External directory (full path)

Value

A Bayesian network that models fissile material operations (op), controls (ctrl), and parameters that affect nuclear criticality safety

Examples

```
ext.dir <- paste0(tempdir(), "/criticality/extdata")
dir.create(ext.dir, recursive = TRUE, showWarnings = FALSE)

extdata <- paste0(.libPaths()[1], "/criticality/extdata")
file.copy(paste0(extdata, "/facility.csv"), ext.dir, recursive = TRUE)
file.copy(paste0(extdata, "/mcnp-dataset.RData"), ext.dir, recursive = TRUE)

BN(
  dist = "gamma",
  facility.data = "facility.csv",
  ext.dir = ext.dir
)
```

Model *Model Function*

Description

This function builds the deep neural network metamodel architecture.

Usage

```
Model(
  dataset,
  layers = "8192-256-256-256-256-16",
  loss = "sse",
  opt.alg = "adamax",
  learning.rate = 0.00075,
  ext.dir
)
```

Arguments

dataset	Training and test data
layers	String that defines the deep neural network architecture (e.g., "64-64")
loss	Loss function
opt.alg	Optimization algorithm
learning.rate	Learning rate
ext.dir	External directory (full path)

Value

A deep neural network metamodel of Monte Carlo radiation transport code simulation data

NN *NN Function*

Description

This function imports the Tabulate, Scale, Model, Fit, Plot, and Test functions to train an ensemble of deep neural networks to predict keff values.

Usage

```
NN(
  batch.size = 8192,
  code = "mcnp",
  dataset,
  ensemble.size = 5,
  epochs = 1500,
  layers = "8192-256-256-256-256-16",
  loss = "sse",
  opt.alg = "adamax",
  learning.rate = 0.00075,
  val.split = 0.2,
  overwrite = FALSE,
  replot = TRUE,
  verbose = FALSE,
  ext.dir,
  training.dir = NULL
)
```

Arguments

batch.size	Batch size
code	Monte Carlo radiation transport code (e.g., "cog", "mcnp")
dataset	Training and test data
ensemble.size	Number of deep neural networks in the ensemble
epochs	Number of training epochs
layers	String that defines the deep neural network architecture (e.g., "64-64")
loss	Loss function
opt.alg	Optimization algorithm
learning.rate	Learning rate
val.split	Validation split
overwrite	Boolean (TRUE/FALSE) that determines if files should be overwritten
replot	Boolean (TRUE/FALSE) that determines if .png files should be replotted
verbose	Boolean (TRUE/FALSE) that determines if TensorFlow and Test function output should be displayed
ext.dir	External directory (full path)
training.dir	Training directory (full path)

Value

A list of lists containing an ensemble of deep neural networks and weights

Examples

```
ext.dir <- paste0(tempdir(), "/criticality/extdata")
dir.create(ext.dir, recursive = TRUE, showWarnings = FALSE)

extdata <- paste0(.libPaths()[1], "/criticality/extdata")
file.copy(paste0(extdata, "/facility.csv"), ext.dir, recursive = TRUE)
file.copy(paste0(extdata, "/mcnp-dataset.RData"), ext.dir, recursive = TRUE)

config <- FALSE
try(config <- reticulate::py_config()$available)
try(if (config == TRUE) {
  NN(
    batch.size = 128,
    code = "mcnp",
    ensemble.size = 1,
    epochs = 10,
    layers = "256-256-16",
    loss = "sse",
    opt.alg = "adamax",
    learning.rate = 0.00075,
    val.split = 0.2,
    replot = FALSE,
    verbose = FALSE,
    ext.dir = ext.dir
  )
})
```

Plot *Plot Function*

Description

This function generates and saves plots and data.

Usage

```
Plot(i, history = NULL, plot.dir)
```

Arguments

i	Model number
history	Training history
plot.dir	Plot directory (full path)

Value

No output (generates and saves ggplot2 files and training histories)

`Risk` *Risk Function*

Description

This function imports the Sample function and estimates process criticality accident risk.

Usage

```
Risk(
  bn,
  code = "mcnp",
  cores = parallel::detectCores()/2,
  dist = "gamma",
  facility.data,
  keff.cutoff = 0.9,
  metamodel,
  risk.pool = 100,
  sample.size = 1e+09,
  usl = 0.95,
  ext.dir
)
```

Arguments

`bn`	Bayesian network
`code`	Monte Carlo radiation transport code (e.g., "cog", "mcnp")
`cores`	Number of CPU cores to use for generating Bayesian network samples
`dist`	Truncated probability distribution (e.g., "gamma", "normal")
`facility.data`	.csv file name
`keff.cutoff`	keff cutoff value (e.g., keff >= 0.9)
`metamodel`	List of deep neural network metamodels and weights
`risk.pool`	Number of times risk is calculated
`sample.size`	Number of samples used to calculate risk
`usl`	Upper subcritical limit (e.g., keff >= 0.95)
`ext.dir`	External directory (full path)

Value

A list of lists containing process criticality accident risk estimates and Bayesian network samples

Examples

```
ext.dir <- paste0(tempdir(), "/criticality/extdata")
dir.create(ext.dir, recursive = TRUE, showWarnings = FALSE)

extdata <- paste0(.libPaths()[1], "/criticality/extdata")
file.copy(paste0(extdata, "/facility.csv"), ext.dir, recursive = TRUE)
file.copy(paste0(extdata, "/mcnp-dataset.RData"), ext.dir, recursive = TRUE)

config <- FALSE
try(config <- reticulate::py_config()$available)
try(if (config == TRUE) {
  Risk(
    bn = BN(
      facility.data = "facility.csv",
      dist = "gamma",
      ext.dir = ext.dir),
    code = "mcnp",
    cores = 1,
    dist = "gamma",
    facility.data = "facility.csv",
    keff.cutoff = 0.5,
    metamodel = NN(
      batch.size = 128,
      code = "mcnp",
      ensemble.size = 1,
      epochs = 10,
      layers = "8192-256-256-256-256-16",
      loss = "sse",
      opt.alg = "adamax",
      learning.rate = 0.00075,
      val.split = 0.2,
      replot = TRUE,
      verbose = TRUE,
      ext.dir = ext.dir),
    risk.pool = 10,
    sample.size = 1e+04,
    usl = 0.95,
    ext.dir = ext.dir
  )
})
```

`Sample` *Sample Function*

Description

This function samples the Bayesian network and generates keff predictions using a deep neural network metamodel.

Usage

```
Sample(
  bn,
  code = "mcnp",
  cores = parallel::detectCores()/2,
  keff.cutoff = 0.9,
  metamodel,
  sample.size = 1e+09,
  ext.dir,
  risk.dir = NULL
)
```

Arguments

bn	Bayesian network object
code	Monte Carlo radiation transport code (e.g., "cog", "mcnp")
cores	Number of CPU cores to use for generating Bayesian network samples
keff.cutoff	keff cutoff value (e.g., 0.9)
metamodel	List of deep neural network metamodels and weights
sample.size	Number of samples used to calculate risk
ext.dir	External directory (full path)
risk.dir	Risk directory

Value

A list of Bayesian network samples with predicted keff values

---

Scale *Scale Function*

Description

This function centers, scales, and one-hot encodes variables.

Usage

```
Scale(code = "mcnp", dataset = NULL, output, ext.dir)
```

Arguments

code	Monte Carlo radiation transport code (e.g., "cog", "mcnp")
dataset	Training and test data
output	Processed output from Monte Carlo radiation transport code simulations
ext.dir	External directory (full path)

Value

A list of centered, scaled, and one-hot-encoded training and test data

---

`Tabulate` *Tabulate Function*

Description

This function imports the Scale function and loads/saves training and test data.

Usage

```
Tabulate(code = "mcnp", ext.dir)
```

Arguments

`code`	Monte Carlo radiation transport code (e.g., "cog", "mcnp")
`ext.dir`	External directory (full path)

Value

A list of centered, scaled, and one-hot-encoded training and test data

Examples

```
ext.dir <- paste0(tempdir(), "/criticality/extdata")
dir.create(ext.dir, recursive = TRUE, showWarnings = FALSE)

extdata <- paste0(.libPaths()[1], "/criticality/extdata")
file.copy(paste0(extdata, "/facility.csv"), ext.dir, recursive = TRUE)
file.copy(paste0(extdata, "/mcnp-dataset.RData"), ext.dir, recursive = TRUE)

Tabulate(
  code = "mcnp",
  ext.dir = ext.dir
)
```

`Test` *Test Function*

Description

This function calculates deep neural network metamodel weights and generates keff predictions for all training and test data.

Usage

```
Test(
  dataset,
  ensemble.size = 5,
  loss = "sse",
  verbose = FALSE,
  ext.dir,
  training.dir
)
```

Arguments

`dataset`	Training and test data
`ensemble.size`	Number of deep neural networks in the ensemble
`loss`	Loss function
`verbose`	Boolean (TRUE/FALSE) that determines if Test function output should be displayed
`ext.dir`	External directory (full path)
`training.dir`	Training directory (full path)

Value

A list of deep neural network weights

Index