Package: yager (via r-universe)

October 11, 2024

Title Yet Another General Regression Neural Network

Version 0.1.1

Author WenSui Liu

Maintainer WenSui Liu <liuwensui@gmail.com>

Description Another implementation of general regression neural network in R based on Specht (1991) <DOI:10.1109/72.97934>. It is applicable to the functional approximation or the classification.

URL https://github.com/statcompute/yager

Depends R (>= 3.6.0)

Imports datasets, stats, randtoolbox, lhs, MLmetrics, graphics, parallel

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Date/Publication 2020-10-25 18:30:02 UTC

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

RemoteUrl https://github.com/cran/yager

RemoteRef HEAD

RemoteSha d9dbdd097273d4da0104192f8cc4473073fe8500

Contents

`folds` *Generate a list of index for the n-fold cross-validation*

Description

The function `folds` generates a list of index for the n-fold cross-validation

Usage

```
folds(idx, n, seed = 1)
```

Arguments

`idx`	A vector of index list
`n`	The number of n folds
`seed`	The seed value to generate random n-fold index

Value

A list of n-fold index

Examples

```
folds(seq(10), n = 3, seed = 2020)
```

gen_latin *Generate random numbers of latin hypercube sampling*

Description

The function gen_latin generates a vector of random numbers by latin hypercube sampling

Usage

```
gen_latin(min = 0, max = 1, n, seed = 1)
```

Arguments

min	The minimum value of random numbers
max	The maxinum value of random numbers
n	The number of random numbers to gernate
seed	The seed value of random number generation

Value

A vector of random numbers bounded by the min and max

Examples

```
gen_latin(0, 1, 10, 2020)
```

gen_sobol *Generate sobol sequence*

Description

The function gen_sobol generates a vector of scrambled sobol sequence

Usage

```
gen_sobol(min = 0, max = 1, n, seed = 1)
```

Arguments

min	The minimum value of random numbers
max	The maxinum value of random numbers
n	The number of random numbers to gernate
seed	The seed value of random number generation

Value

A vector of sobol sequence bounded by the min and max

Examples

```
gen_sobol(0, 1, 10, 2020)
```

`gen_unifm` *Generate Uniform random numbers*

Description

The function `gen_unifm` generates a vector of uniform random numbers

Usage

```
gen_unifm(min = 0, max = 1, n, seed = 1)
```

Arguments

`min`	The minimum value of random numbers
`max`	The maxinum value of random numbers
`n`	The number of random numbers to gernate
`seed`	The seed value of random number generation

Value

A vector of uniform random numbers bounded by the min and max

Examples

```
gen_unifm(0, 1, 10, 2020)
```

grnn.fit *Create a general regression neural network*

Description

The function `grnn.fit` creates a general regression neural network (GRNN)

Usage

```
grnn.fit(x, y, sigma = 1, w = rep(1, length(y)))
```

Arguments

`x`	The matrix of predictors
`y`	The vector of response variable
`sigma`	The scalar of smoothing parameter
`w`	The vector of weights with default = 1 for each record

Value

A general regression neural network object

References

Donald Specht. (1991). A General Regression Neural Network.

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
```

grnn.imp *Derive the importance rank of all predictors used in the GRNN*

Description

The function `grnn.imp` derives the importance rank of all predictors used in the GRNN It essentially is a wrapper around the function `grnn.x_imp`.

Usage

```
grnn.imp(net, class = FALSE)
```

Arguments

net	The GRNN object generated by grnn.fit()
class	TRUE or FALSE, whether it is for the classification or not

Value

A dataframe with important values of all predictors in the GRNN

See Also

grnn.x_imp

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:3])
gnet <- grnn.fit(x = X, y = Y)
## Not run:
grnn.imp(net = gnet, class = TRUE)

## End(Not run)
```

---

grnn.margin *Derive the marginal effect of a predictor used in a GRNN*

Description

The function grnn.margin derives the marginal effect of a predictor used in a GRNN by assuming mean values for the rest predictors

Usage

```
grnn.margin(net, i, plot = TRUE)
```

Arguments

net	The GRNN object generated by grnn.fit()
i	The ith predictor in the GRNN
plot	TRUE or FALSE to plot the marginal effect

Value

A plot of the marginal effect or a dataframe of the marginal effect

See Also

grnn.partial

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.margin(gnet, 1, plot = FALSE)
```

`grnn.optmiz_auc` *Optimize the optimal value of GRNN smoothing parameter based on AUC*

Description

The function `grnn.optmiz_auc` optimize the optimal value of GRNN smoothing parameter by cross-validation. It is applicable to the classification.

Usage

```
grnn.optmiz_auc(net, lower = 0, upper, nfolds = 4, seed = 1, method = 1)
```

Arguments

`net`	A GRNN object generated by grnn.fit()
`lower`	A scalar for the lower bound of the smoothing parameter
`upper`	A scalar for the upper bound of the smoothing parameter
`nfolds`	A scalar for the number of n-fold, 4 by default
`seed`	The seed value for the n-fold cross-validation, 1 by default
`method`	A scalar referring to the optimization method, 1 for Golden section searc and 2 for Brent's method

Value

The best outcome

See Also

`grnn.search_auc`

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
## Not run:
grnn.optmiz_auc(net = gnet, lower = 3, upper = 7, nfolds = 2)

## End(Not run)
```

grnn.parpred *Calculate predicted values of GRNN by using parallelism*

Description

The function grnn.parpred calculates a vector of GRNN predicted values based on an input matrix

Usage

```
grnn.parpred(net, x)
```

Arguments

net	The GRNN object generated by grnn.fit()
x	The matrix of input predictors

Value

A vector of predicted values

See Also

grnn.predict

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.parpred(gnet, X[seq(5), ])
```

grnn.partial *Derive the partial effect of a predictor used in a GRNN*

Description

The function grnn.partial derives the partial effect of a predictor used in a GRNN by average-out values of the rest predictors.

Usage

```
grnn.partial(net, i, plot = TRUE)
```

Arguments

net	The GRNN object generated by grnn.fit()
i	The ith predictor in the GRNN
plot	TRUE or FALSE to plot the partial effect

Value

A plot of the partial effect or a dataframe of the partial effect

See Also

grnn.margin

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
## Not run:
grnn.partial(gnet, 1, plot = FALSE)

## End(Not run)
```

---

grnn.pfi *Derive the PFI rank of all predictors used in the GRNN*

Description

The function grnn.pfi derives the PFI rank of all predictors used in the GRNN It essentially is a wrapper around the function grnn.x_pfi.

Usage

```
grnn.pfi(net, class = FALSE, ntry = 1000, seed = 1)
```

Arguments

net	The GRNN object generated by grnn.fit()
class	TRUE or FALSE, whether it is for the classification or not
ntry	The number of random permutations to try, 1e3 times by default
seed	The seed value for the random permutation

Value

A dataframe with PFI values of all predictors in the GRNN

See Also

grnn.x_pfi

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:3])
gnet <- grnn.fit(x = X, y = Y)
## Not run:
grnn.pfi(net = gnet, class = TRUE)

## End(Not run)
```

---

grnn.predict *Calculate predicted values of GRNN*

Description

The function grnn.predict calculates a vector of GRNN predicted values based on an input matrix

Usage

```
grnn.predict(net, x)
```

Arguments

net	The GRNN object generated by grnn.fit()
x	The matrix of input predictors

Value

A vector of predicted values

See Also

grnn.predone

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.predict(gnet, X[seq(5), ])
```

grnn.predone *Calculate a predicted value of GRNN*

Description

The function grnn.predone calculates a predicted value of GRNN based on an input vector

The function grnn.predone calculates a predicted value of GRNN based on an input vector

Usage

```
grnn.predone(net, x, type = 1)
```


```
grnn.predone(net, x, type = 1)
```


Arguments

net	The GRNN object generated by grnn.fit()
x	The vector of input predictors
type	A scalar, 1 for euclidean distance and 2 for manhattan distance

Value

A scalar of the predicted value

A scalar of the predicted value

References

Donald Specht. (1991). A General Regression Neural Network.

Donald Specht. (1991). A General Regression Neural Network.

See Also

grnn.fit

grnn.fit

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
for (i in seq(5)) print(grnn.predone(gnet, X[i, ]))
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
for (i in seq(5)) print(grnn.predone(gnet, X[i, ]))
```

grnn.search_auc — *Search for the optimal value of GRNN smoothing parameter based on AUC*

Description

The function grnn.search_auc searches for the optimal value of GRNN smoothing parameter by cross-validation. It is applicable to the classification.

Usage

```
grnn.search_auc(net, sigmas, nfolds = 4, seed = 1)
```

Arguments

net	A GRNN object generated by grnn.fit()
sigmas	A numeric vector to search for the best smoothing parameter
nfolds	A scalar for the number of n-fold, 4 by default
seed	The seed value for the n-fold cross-validation, 1 by default

Value

The list of all searching outcomes and the best outcome

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.search_auc(net = gnet, sigmas = c(3, 5, 7), nfolds = 2)
```

grnn.search_rsq — *Search for the optimal value of GRNN smoothing parameter based on r-square*

Description

The function grnn.search_rsq searches for the optimal value of GRNN smoothing parameter by cross-validation. It is applicable to the functional approximation

Usage

```
grnn.search_rsq(net, sigmas, nfolds = 4, seed = 1)
```

Arguments

net	A GRNN object generated by grnn.fit()
sigmas	A numeric vector to search for the best smoothing parameter
nfolds	A scalar for the number of n-fold, 4 by default
seed	The seed value for the n-fold cross-validation, 1 by default

Value

The list of all searching outcomes and the best outcome

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.search_rsq(net = gnet, sigmas = seq(3), nfolds = 2)
```

---

grnn.x_imp *Derive the importance of a predictor used in the GRNN*

Description

The function grnn.x_imp derives the importance of a predictor used in the GRNN by using the loss of predictability after eliminating the impact of the predictor in interest.

Usage

```
grnn.x_imp(net, i, class = FALSE)
```

Arguments

net	The GRNN object generated by grnn.fit()
i	The ith predictor in the GRNN
class	TRUE or FALSE, whether it is for the classification or not

Value

A vector with the variable name and two values of importance measurements, namely "imp1" and "imp2". The "imp1" measures the loss of predictability after replacing all values of the predictor with its mean. The "imp2" measures the loss of predictability after dropping the predictor from the GRNN.

See Also

grnn.x_pfi

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.x_imp(net = gnet, 1)
```

`grnn.x_pfi` *Derive the permutation feature importance of a predictor used in the GRNN*

Description

The function `grnn.x_pfi` derives the permutation feature importance (PFI) of a predictor used in the GRNN

Usage

```
grnn.x_pfi(net, i, class = FALSE, ntry = 1000, seed = 1)
```

Arguments

`net`	The GRNN object generated by grnn.fit()
`i`	The ith predictor in the GRNN
`class`	TRUE or FALSE, whether it is for the classification or not
`ntry`	The number of random permutations to try, 1e3 times by default
`seed`	The seed value for the random permutation

Value

A vector with the variable name and the PFI value.

See Also

`grnn.x_imp`

Examples

```
data(iris, package = "datasets")
Y <- ifelse(iris[, 5] == "setosa", 1, 0)
X <- scale(iris[, 1:4])
gnet <- grnn.fit(x = X, y = Y)
grnn.x_pfi(net = gnet, 1)
```

Index