Package ‘yap’

October 25, 2020

Title Yet Another Probabilistic Neural Network
Version 0.1.1
Author WenSui Liu
Maintainer WenSui Liu <liuwensui@gmail.com>
Description Another implementation of probabilistic neural network in R based on Specht (1990) <DOI:10.1016/0893-6080(90)90049-Q>. It is applicable to the pattern recognition with a N-level response, where N > 2.

URL https://github.com/statcompute/yap
Depends R (>= 3.6.0)
Imports stats, randtoolbox, lhs, parallel, datasets
License GPL (>= 2)
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2020-10-25 20:30:02 UTC

R topics documented:

dummies ................................................................. 2
folds ................................................................. 2
gen_latin ........................................................... 3
gen_sobol ........................................................... 4
gen_unifm ........................................................... 4
logl ................................................................. 5
pnn.fit ............................................................ 6
pnn.imp ............................................................ 6
pnn.optimiz_logl .................................................. 7
pnn.parpred ....................................................... 8
pnn.pfi ............................................................ 9
dummies

Convert a N-category vector to a N-dimension matrix

Description

The function dummies converts a N-category vector to a N-dimension matrix

Usage

dummies(x)

Arguments

x A N-category vector

Value

A N-dimension matrix with 0/1 values

Examples

data(iris, package = "datasets")
dummies(iris[, 5])

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**

```r
tails(seq(10), 3, 2020)
```

---

**Description**

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

**Usage**

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

**Arguments**

- `min` The minimum value of random numbers
- `max` The maximum value of random numbers
- `n` The number of random numbers to generate
- `seed` The seed value of random number generation

**Value**

A vector of random numbers bounded by the `min` and `max`.

**Examples**

```r
gen_latin(0, 1, 10, 2020)
```
**gen_sobol**  
*Generate sobol sequence*

**Description**  
The function `gen_sobol` generates a vector of scrambled sobol sequence

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

**Arguments**
- `min`: The minimum value of random numbers
- `max`: The maximum value of random numbers
- `n`: The number of random numbers to generate
- `seed`: The seed value of random number generation

**Value**
A vector of sobol sequence bounded by the min and max

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

**Arguments**
- `min`: The minimum value of random numbers
- `max`: The maximum value of random numbers
- `n`: The number of random numbers to generate
- `seed`: The seed value of random number generation
**Value**

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

**Examples**

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

---

**log1**

*Calculate the multiclass cross-entropy*

**Description**

The function `log1` calculates the multiclass cross entropy

**Usage**

```r
log1(y_true, y_pred)
```

**Arguments**

- `y_true`: A matrix of multiclass 0/1 indicators
- `y_pred`: A matrix of predicted probability of each class

**Value**

The value of multiclass cross entropy

**Examples**

```r
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
log1(y_true = pnet$y.ind, y_pred = pnn.predict(pnet, X))
```
pnn.fit  

Create a probabilistic neural network

Description
The function pnn.fit creates a probabilistic neural network (PNN)

Usage
pnn.fit(x, y, sigma = 1)

Arguments
x  A matrix of predictors
y  A vector of N-category factors
sigma  A scalar with the positive value

Value
A PNN object

References

Examples
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)

pnn.imp  Derive the importance rank of all predictors used in the PNN

Description
The function pnn.imp derives the importance rank of all predictors used in the PNN It essentially is a wrapper around the function pnn.x_imp.

Usage
pnn.imp(net)

Arguments
net  A PNN object generated by pnn.fit()
pnn.optmiz_logl

Value
A dataframe with important values of all predictors in the PNN

See Also
pnn.x_imp

Examples

data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
pnn.imp(pnet)

pnn.optmiz_logl Optimize the optimal value of PNN smoothing parameter based on the cross entropy

Description
The function pnn.optmiz_logl optimize the optimal value of PNN smoothing parameter by cross-validation.

Usage
pnn.optmiz_logl(net, lower = 0, upper, nfolds = 4, seed = 1, method = 1)

Arguments
net A PNN object generated by pnn.fit()
lower A scalar for the lower bound of the smoothing parameter, 0 by default
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 search and 2 for Brent’s method

Value
The best outcome

See Also
pnn.search_logl
Examples

data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)

pnn.optmiz_logl(pnet, upper = 1)

pnn.parpred

Calculate predicted probabilities of PNN by using parallelism

Description

The function `pnn.parpred` calculates a matrix of PNN predicted probabilities based on an input matrix.

Usage

`pnn.parpred(net, x)`

Arguments

- `net`: A PNN object generated by `pnn.fit()`
- `x`: A matrix of input predictors

Value

A matrix of predicted probabilities

See Also

`pnn.predict`

Examples

data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
pnn.parpred(pnet, X[seq(5), ])

pnn.pfi

Derive the PFI rank of all predictors used in the PNN

Description

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

Usage

pnn.pfi(net, ntry = 1000, seed = 1)

Arguments

net A PNN object generated by pnn.fit()
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 PNN

See Also

pnn.x_pfi

Examples

data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)

pnn.pfi(pnet)

-------------------

pnn.predict

Calculate a matrix of predicted probabilities

Description

The function pnn.predict calculates a matrix of predicted probabilities based on a matrix of predictors

Usage

pnn.predict(net, x)
Arguments
net  The PNN object generated by pnn.fit()
x    The matrix of input predictors

Value
A matrix of predicted probabilities for all categories

See Also
pnn.predone

Examples
```r
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
pnn.predict(pnet, X[seq(5), ])
```

pnn.predone  Calculate the predicted probability for each category of PNN

Description
The function pnn.predone calculates the predicted probability for each category of PNN

Usage
```r
pnn.predone(net, x)
pnn.predone(net, x)
```

Arguments
net  A PNN object created by pnn.fit()
x    A vector of input predictors

Value
A one-row matrix of predicted probabilities

See Also
pnn.fit	pnn.fit
Examples

```r
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
for (i in seq(5)) print(pnn.predone(pnet, X[i, ]))
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
for (i in seq(5)) print(pnn.predone(pnet, X[i, ]))
```

---

**pnn.search_logl**

Search for the optimal value of PNN smoothing parameter based on the cross entropy

**Description**

The function `pnn.search_logl` searches for the optimal value of PNN smoothing parameter by cross-validation.

**Usage**

```r
pnn.search_logl(net, sigmas, nfolds = 4, seed = 1)
```

**Arguments**

- `net` A PNN object generated by `pnn.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**

```r
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
pnn.search_logl(pnet, c(0.5, 1), nfolds = 2)
```
Derive the importance of a predictor used in the PNN

Description

The function `pnn.x_imp` derives the importance of a predictor used in the PNN, where the "importance" is measured by the increase in cross entropy after eliminating the impact of the predictor in interest.

Usage

```r
pnn.x_imp(net, i)
```

Arguments

- `net`: A PNN object generated by `pnn.fit()`
- `i`: The ith predictor in the PNN

Value

A vector with the variable name and two values of importance measurements, namely "imp1" and "imp2". The "imp1" measures the increase in cross entropy after replacing all values of the predictor with its mean. The "imp2" measures the increase in cross entropy after dropping the predictor from the PNN.

See Also

`pnn.x_pfi`

Examples

```r
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
pnn.x_imp(pnet, 1)
```

Derive the permutation feature importance of a predictor used in the PNN

Description

The function `pnn.x_pfi` derives the permutation feature importance (PFI) of a predictor used in the PNN, where the "importance" is defined by the increase in cross entropy after the predictor is randomly permuted.
Usage

\texttt{pnn.x_pfi(net, i, ntry = 1000, seed = 1)}

Arguments

\begin{itemize}
  \item \texttt{net} \hspace{1cm} A PNN object generated by \texttt{pnn.fit()}
  \item \texttt{i} \hspace{1cm} The \(i\)th predictor in the PNN
  \item \texttt{ntry} \hspace{1cm} The number of random permutations to try, 1e3 times by default
  \item \texttt{seed} \hspace{1cm} The seed value for the random permutation
\end{itemize}

Value

A vector with the variable name and the PFI value.

See Also

\texttt{pnn.x_imp}

Examples

\begin{verbatim}
data(iris, package = "datasets")
Y <- iris[, 5]
X <- scale(iris[, 1:4])
pnet <- pnn.fit(x = X, y = Y)
pnn.x_pfi(pnet, 1)
\end{verbatim}
Index

dummies, 2
folds, 2
gen_latin, 3
gen_sobol, 4
gen_unifm, 4
logl, 5
pnn.fit, 6, 10
pnn.imp, 6
pnn.optimiz_logl, 7
pnn.parpred, 8
pnn.pfi, 9
pnn.predict, 8, 9
pnn.predone, 10, 10
pnn.search_logl, 7, 11
pnn.x_imp, 7, 12, 13
pnn.x_pfi, 9, 12, 12