Package: RWNN (via r-universe)

October 6, 2026

Type Package

Title Random Weight Neural Networks

Version 0.4

Date 2024-08-29

Description Creation, estimation, and prediction of random weight neural networks (RWNN), Schmidt et al. (1992) <doi:10.1109/ICPR.1992.201708>, including popular variants like extreme learning machines, Huang et al. (2006) <doi:10.1016/j.neucom.2005.12.126>, sparse RWNN, Zhang et al. (2019) <doi:10.1016/j.neunet.2019.01.007>, and deep RWNN, Henríquez et al. (2018) <doi:10.1109/IJCNN.2018.8489703>. It further allows for the creation of ensemble RWNNs like bagging RWNN, Sui et al. (2021) <doi:10.1109/ECCE47101.2021.9595113>, boosting RWNN, stacking RWNN, and ensemble deep RWNN, Shi et al. (2021) <doi:10.1016/j.patcog.2021.107978>.

License MIT + file LICENSE

Imports methods, quadprog, randtoolbox, Rcpp (>= 1.0.4.6), stats, utils

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.3.1

Suggests tinytest

Depends R (>= 4.1.0)

LazyData true

NeedsCompilation yes

Author Søren B. Vilsen [aut, cre]

Maintainer Søren B. Vilsen <svilsen@math.aau.dk>

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

Date/Publication 2024-09-04 02:51:50 UTC

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

RemoteRef HEAD

RemoteSha 731fbfcfd64167ab79879a705899501fd5c92526

Contents

ae_rwnn 2
bag_rwnn 3
boost_rwnn 5
classify 7
control_rwnn 7
ed_rwnn 9
ERWNN-object 11
example_data 11
predict.ERWNN 12
predict.RWNN 13
reduce_network 13
rwnn 16
RWNN-object 18
stack_rwnn 18

Index **21**

`ae_rwnn` *Auto-encoder pre-trained random weight neural networks*

Description

Set-up and estimate weights of a random weight neural network using an auto-encoder for unsupervised pre-training of the hidden weights.

Usage

```
ae_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  method = "l1",
  type = NULL,
  control = list()
)

## S3 method for class 'formula'
ae_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  method = "l1",
  type = NULL,
  control = list()
)
```

Arguments

formula	A formula specifying features and targets used to estimate the parameters of the output-layer.
data	A data-set (either a data.frame or a tibble) used to estimate the parameters of the output-layer.
n_hidden	A vector of integers designating the number of neurons in each of the hidden-layers (the length of the list is taken as the number of hidden-layers).
lambda	A vector of two penalisation constants used when encoding the hidden-weights and training the output-weights, respectively.
method	The penalisation type used for the auto-encoder (either "l1" or "l2").
type	A string indicating whether this is a regression or classification problem.
control	A list of additional arguments passed to the control_rwnn function.

Value

An RWNN-object.

References

Zhang Y., Wu J., Cai Z., Du B., Yu P.S. (2019) "An unsupervised parameter learning model for RVFL neural network." *Neural Networks*, 112, 85-97.

Examples

```
n_hidden <- c(20, 15, 10, 5)
lambda <- c(2, 0.01)

## Using L1-norm in the auto-encoder (sparse solution)
m <- ae_rwnn(y ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda, method = "l1")

## Using L2-norm in the auto-encoder (dense solution)
m <- ae_rwnn(y ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda, method = "l2")
```

---

bag_rwnn *Bagging random weight neural networks*

Description

Use bootstrap aggregation to reduce the variance of random weight neural network models.

Usage

```
bag_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  B = 100,
  method = NULL,
  type = NULL,
  control = list()
)

## S3 method for class 'formula'
bag_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  B = 100,
  method = NULL,
  type = NULL,
  control = list()
)
```

Arguments

formula	A formula specifying features and targets used to estimate the parameters of the output layer.
data	A data-set (either a data.frame or a tibble) used to estimate the parameters of the output layer.
n_hidden	A vector of integers designating the number of neurons in each of the hidden layers (the length of the list is taken as the number of hidden layers).
lambda	The penalisation constant(s) passed to either rwnn or ae_rwnn (see method argument).
B	The number of bootstrap samples.
method	The penalisation type passed to ae_rwnn. Set to NULL (default), "l1", or "l2". If NULL, rwnn is used as the base learner.
type	A string indicating whether this is a regression or classification problem.
control	A list of additional arguments passed to the control_rwnn function.

Value

An ERWNN-object.

References

Breiman L. (1996) "Bagging Predictors." *Machine Learning*, 24, 123-140.

Breiman L. (2001) "Random Forests." *Machine Learning*, 45, 5-32.

Sui X, He S, Vilsen SB, Teodorescu R, Stroe DI (2021) "Fast and Robust Estimation of Lithium-ion Batteries State of Health Using Ensemble Learning." *In 2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, 1-8.

Examples

```
n_hidden <- 50

B <- 100
lambda <- 0.01

m <- bag_rwnn(y ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda, B = B)
```

`boost_rwnn` *Boosting random weight neural networks*

Description

Use gradient boosting to create ensemble random weight neural network models.

Usage

```
boost_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  B = 100,
  epsilon = 0.1,
  method = NULL,
  type = NULL,
  control = list()
)

## S3 method for class 'formula'
boost_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  B = 100,
  epsilon = 0.1,
  method = NULL,
```

```
  type = NULL,
  control = list()
)
```

Arguments

formula	A formula specifying features and targets used to estimate the parameters of the output layer.
data	A data-set (either a data.frame or a tibble) used to estimate the parameters of the output layer.
n_hidden	A vector of integers designating the number of neurons in each of the hidden layers (the length of the list is taken as the number of hidden layers).
lambda	The penalisation constant(s) passed to either rwnn or ae_rwnn (see method argument).
B	The number of levels used in the boosting tree.
epsilon	The learning rate.
method	The penalisation type passed to ae_rwnn. Set to NULL (default), "l1", or "l2". If NULL, rwnn is used as the base learner.
type	A string indicating whether this is a regression or classification problem.
control	A list of additional arguments passed to the control_rwnn function.

Value

An ERWNN-object.

References

Friedman J.H. (2001) "Greedy function approximation: A gradrient boosting machine." *The Annals of Statistics*, 29, 1189-1232.

Examples

```
n_hidden <- 10

B <- 100
epsilon <- 0.1
lambda <- 0.01

m <- boost_rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
                lambda = lambda, B = B, epsilon = epsilon)
```

classify *Classifier*

Description

Function classifying an observation.

Usage

```
classify(y, C, t = NULL, b = NULL)
```

Arguments

y	A matrix of predicted classes.
C	A vector of class names corresponding to the columns of y.
t	The decision threshold which the predictions have to exceed (defaults to '0').
b	A buffer which the largest prediction has to exceed when compared to the second largest prediction (defaults to '0').

Value

A vector of class predictions.

control_rwnn *rwnn control function*

Description

A function used to create a control-object for the rwnn function.

Usage

```
control_rwnn(
  n_hidden = NULL,
  n_features = NULL,
  lnorm = NULL,
  bias_hidden = TRUE,
  bias_output = TRUE,
  activation = NULL,
  combine_input = FALSE,
  combine_hidden = TRUE,
  include_data = TRUE,
  include_estimate = TRUE,
  rng = runif,
  rng_pars = list(min = -1, max = 1)
)
```

Arguments

n_hidden	A vector of integers designating the number of neurons in each of the hidden layers (the length of the list is taken as the number of hidden layers).
n_features	The number of randomly chosen features in the RWNN model. Note: This is meant for use in bag_rwnn, and it is not recommended outside of that function.
lnorm	A string indicating the type of regularisation used when estimating the weights in the output layer, "l1" or "l2" (default).
bias_hidden	A vector of TRUE/FALSE values. The vector should have length 1, or be equal to the number of hidden layers.
bias_output	TRUE/FALSE: Should a bias be added to the output layer?
activation	A vector of strings corresponding to activation functions (see details). The vector should have length 1, or be equal to the number of hidden layers.
combine_input	TRUE/FALSE: Should the input be included to predict the output?
combine_hidden	TRUE/FALSE: Should all hidden layers be combined to predict the output?
include_data	TRUE/FALSE: Should the original data be included in the returned object? Note: this should almost always be set to 'TRUE', but using 'FALSE' is more memory efficient in ERWNN-object's.
include_estimate	TRUE/FALSE: Should the rwnn-function estimate the output parameters? Note: this should almost always be set to 'TRUE', but using 'FALSE'is more memory efficient in ERWNN-object's.
rng	A string indicating the sampling distribution used for generating the weights of the hidden layer (defaults to runif).
rng_pars	A list of parameters passed to the rng function (defaults to list(min = -1, max = 1)).

Details

The possible activation functions supplied to 'activation' are:

"identity"

$$f(x) = x$$

"bentidentity"

$$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$$

"sigmoid"

$$f(x) = \frac{1}{1+\exp(-x)}$$

"tanh"

$$f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$$

"relu"

$$f(x) = \max\{0, x\}$$

`"silu"` **(default)**

$$f(x) = \frac{x}{1 + \exp(-x)}$$

`"softplus"`

$$f(x) = \ln(1 + \exp(x))$$

`"softsign"`

$$f(x) = \frac{x}{1 + |x|}$$

`"sqnl"`

$$f(x) = -1, \text{ if } x < -2,\ f(x) = x + \frac{x^2}{4}, \text{ if } -2 \leq x < 0,\ f(x) = x - \frac{x^2}{4}, \text{ if } 0 \leq x \leq 2, \text{ and } f(x) = 2, \text{ if } x > 2$$

`"gaussian"`

$$f(x) = \exp(-x^2)$$

`"sqrbf"`

$$f(x) = 1 - \frac{x^2}{2}, \text{ if } |x| \leq 1,\ f(x) = \frac{(2 - |x|)^2}{2}, \text{ if } 1 < |x| < 2, \text{ and } f(x) = 0, \text{ if } |x| \geq 2$$

The 'rng' argument can also be set to `"orthogonal"`, `"torus"`, `"halton"`, or `"sobol"` for added stability. The `"torus"`, `"halton"`, and `"sobol"` methods relay on the torus, halton, and sobol functions. NB: this is not recommended when creating ensembles.

Value

A list of control variables.

References

Wang W., Liu X. (2017) "The selection of input weights of extreme learning machine: A sample structure preserving point of view." *Neurocomputing*, 261, 28-36.

---

ed_rwnn *Ensemble deep random weight neural networks*

Description

Use multiple layers to create deep ensemble random weight neural network models.

Usage

```
ed_rwnn(
  formula,
  data = NULL,
  n_hidden,
  lambda = 0,
  method = NULL,
  type = NULL,
  control = list()
)

## S3 method for class 'formula'
ed_rwnn(
  formula,
  data = NULL,
  n_hidden,
  lambda = 0,
  method = NULL,
  type = NULL,
  control = list()
)
```

Arguments

formula	A formula specifying features and targets used to estimate the parameters of the output layer.
data	A data-set (either a data.frame or a tibble) used to estimate the parameters of the output layer.
n_hidden	A vector of integers designating the number of neurons in each of the hidden layers (the length of the list is taken as the number of hidden layers).
lambda	The penalisation constant(s) passed to either rwnn or ae_rwnn (see method argument).
method	The penalisation type passed to ae_rwnn. Set to NULL (default), "l1", or "l2". If NULL, rwnn is used as the base learner.
type	A string indicating whether this is a regression or classification problem.
control	A list of additional arguments passed to the control_rwnn function.

Value

An ERWNN-object.

References

Shi Q., Katuwal R., Suganthan P., Tanveer M. (2021) "Random vector functional link neural network based ensemble deep learning." *Pattern Recognition*, 117, 107978.

Examples

```
n_hidden <- c(20, 15, 10, 5)
lambda <- 0.01

#
m <- ed_rwnn(y ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda)
```

`ERWNN-object` *An ERWNN-object*

Description

An ERWNN-object is a list containing the following:

`data` The original data used to estimate the weights.

`models` A list with each element being an RWNN-object.

`weights` A vector of ensemble weights.

`method` A string indicating the method.

`example_data` *Example data*

Description

A data-set of 2000 observations were sampled independently according to the function:

$$y_n = \frac{1}{1 + \exp(-x_n^T \beta + \varepsilon_n)},$$

where x_n^T is a vector containing an intercept and five input features, β is a vector containing the parameters, $(-1 \;\; 2 \;\; 1 \;\; 2 \;\; 0.5 \;\; 3)^T$, and ε_n is normally distributed noise with mean 0 and variance 0.1. Furthermore, the five features were generated as $x_1 \sim \mathcal{U}nif(-5, 5)$, $x_2 \sim \mathcal{U}nif(0, 2)$, $x_3 \sim \mathcal{N}(2, 4)$, $x_4 \sim \mathcal{G}amma(2, 4)$, and $x_5 \sim \mathcal{B}eta(10, 4)$, respectively.

Usage

```
example_data
```

Format

An object of class `data.frame` with 2000 rows and 6 columns.

predict.ERWNN *Predicting targets of an ERWNN-object*

Description

Predicting targets of an ERWNN-object

Usage

```
## S3 method for class 'ERWNN'
predict(object, ...)
```

Arguments

object	An ERWNN-object.
...	Additional arguments.

Details

The additional arguments 'newdata', 'type', and 'class' can be specified as follows:

newdata Expects a matrix or data.frame with the same features (columns) as in the original data.

type A string taking the following values:

- "mean" (default) Returns the average prediction across all ensemble models.
- "std" Returns the standard deviation of the predictions across all ensemble models.
- "all" Returns all predictions for each ensemble models.

class A string taking the following values:

- "classify" Returns the predicted class of the ensemble. If used together with type = "mean", the average prediction across the ensemble models are used to create the classification. However, if used with type = "all", every ensemble is classified and returned.
- "voting" Returns the predicted class of the ensemble by classifying each ensemble and using majority voting to make the final prediction. NB: the type argument is overruled.

Furthermore, if 'class' is set to either "classify" or "voting", additional arguments 't' and 'b' can be passed to the classify-function.

NB: if the ensemble is created using the boost_rwnn-function, then type should always be set to "mean".

Value

An list, matrix, or vector of predicted values depended on the arguments 'method', 'type', and 'class'.

predict.RWNN *Predicting targets of an RWNN-object*

Description

Predicting targets of an RWNN-object

Usage

```
## S3 method for class 'RWNN'
predict(object, ...)
```

Arguments

object	An RWNN-object.
...	Additional arguments.

Details

The additional arguments used by the function are 'newdata' and 'class'. The argument 'newdata' expects a matrix or data.frame with the same features (columns) as in the original data. While the 'class' argument can be set to "classify". If class == "classify" additional arguments 't' and 'b' can be passed to the classify-function.

Value

A vector of predicted values.

reduce_network *Reduce the weights of a random weight neural network.*

Description

Methods for weight and neuron pruning in random weight neural networks.

Usage

```
reduce_network(object, method, retrain = TRUE, ...)

## S3 method for class 'RWNN'
reduce_network(object, method, retrain = TRUE, ...)

## S3 method for class 'ERWNN'
reduce_network(object, method, retrain = TRUE, ...)
```

Arguments

`object`	An RWNN-object or ERWNN-object.
`method`	A string, or a function, setting the method used to reduce the network (see details).
`retrain`	TRUE/FALSE: Should the output weights be retrained after reduction (defaults to TRUE)?
`...`	Additional arguments passed to the reduction method (see details).

Details

The 'method' and additional arguments required by the method are:

`"global"` **(or** `"glbl"`**)** `p`**: The proportion of weights to remove globally based on magnitude.**

`"uniform"` **(or** `"unif"`**)** `p`**: The proportion of weights to remove uniformly layer-by-layer based on magnitude.**

`"lamp"` `p`**: The proportion of weights to remove based on LAMP scores.**

`"apoz"` `p`**: The proportion of neurons to remove based on proportion of zeroes produced.**

`tolerance`**: The tolerance used when identifying zeroes.**

`type`**: A string indicating whether weights should be removed globally (**`'global'`**) or uniformly (**`'uniform'`**).**

`"correlation"` **(or** `"cor"`**)** `type`**: The type of correlation (argument passed to cor function).**

`rho`**: The correlation threshold used to remove neurons.**

`"correlationtest"` **(or** `"cortest"`**)** `type`**: The type of correlation (argument passed to cor function).**

`rho`**: The correlation threshold used to remove neurons.**

`alpha`**: The significance levels used to test whether the observed correlation between two neurons is small than r**

`"relief"` `p`**: The proportion of neurons or weights to remove based on relief scores.**

`type`**: A string indicating whether neurons (**`'neuron'`**) or weights (**`'weight'`**) should be removed.**

`"output"` `tolerance`**: The tolerance used when removing zeroes from the output layer.**

If the object is an ERWNN-object, the reduction is applied to all RWNN-object's in the ERWNN-object. Furthermore, when the ERWNN-object is created as a stack and the weights of the stack is trained, then 'method' can be set to:

`"stack"` `tolerance`**: The tolerance used when removing elements from the stack.**

Lastly, 'method' can also be passed as a function, with additional arguments passed through the ... argument. NB: features and target are passed using the names `X` and `y`, respectively.

Value

A reduced RWNN-object or ERWNN-object.

References

Han S., Mao H., Dally W.J. (2016) "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding." arXiv: 1510.00149.

Hu H., Peng R., Tai Y.W., Tang C.K. (2016) "Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures." arXiv: 1607.03250.

Morcos A.S., Yu H., Paganini M., Tian Y. (2019) "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers." arXiv: 1906.02773.

Lee J., Park S., Mo S., Ahn S., Shin J. (2021) "Layer-adaptive sparsity for the Magnitude-based Pruning." arXiv: 2010.07611.

Dekhovich A., Tax D.M., Sluiter M.H., Bessa M.A. (2024) "Neural network relief: a pruning algorithm based on neural activity." *Machine Learning*, 113, 2597-2618.

Examples

```
## RWNN-object
n_hidden <- c(10, 15)
lambda <- 2

m <- rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
          lambda = lambda, control = list(lnorm = "l2"))

m |>
    reduce_network(method = "relief", p = 0.2, type = "neuron") |>
    (\(x) x$weights)()

m |>
    reduce_network(method = "relief", p = 0.2, type = "neuron") |>
    reduce_network(method = "correlationtest", rho = 0.995, alpha = 0.05) |>
    (\(x) x$weights)()


m |>
    reduce_network(method = "relief", p = 0.2, type = "neuron") |>
    reduce_network(method = "correlationtest", rho = 0.995, alpha = 0.05) |>
    reduce_network(method = "lamp", p = 0.2) |>
    (\(x) x$weights)()

m |>
    reduce_network(method = "relief", p = 0.4, type = "neuron") |>
    reduce_network(method = "relief", p = 0.4, type = "weight") |>
    reduce_network(method = "output") |>
    (\(x) x$weights)()

## ERWNN-object (reduction is performed element-wise on each RWNN)
n_hidden <- c(10, 15)
lambda <- 2
B <- 100


m <- bag_rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
```

```
                lambda = lambda, B = B, control = list(lnorm = "l2"))

m |>
    reduce_network(method = "relief", p = 0.2, type = "neuron") |>
    reduce_network(method = "relief", p = 0.2, type = "weight") |>
    reduce_network(method = "output")



m <- stack_rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
                lambda = lambda, B = B, optimise = TRUE)

# Number of models in stack
length(m$weights)
# Number of models in stack with weights > .Machine$double.eps
length(m$weights[m$weights > .Machine$double.eps])

m |>
    reduce_network(method = "stack", tolerance = 1e-8) |>
    (\(x) x$weights)()
```

---

rwnn *Random weight neural networks*

Description

Set-up and estimate weights of a random weight neural network.

Usage

```
rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = 0,
  type = NULL,
  control = list()
)

## S3 method for class 'formula'
rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = 0,
  type = NULL,
  control = list()
)
```

Arguments

formula	A formula specifying features and targets used to estimate the parameters of the output layer.
data	A data-set (either a data.frame or a tibble) used to estimate the parameters of the output layer.
n_hidden	A vector of integers designating the number of neurons in each of the hidden layers (the length of the list is taken as the number of hidden layers).
lambda	The penalisation constant used when training the output layer.
type	A string indicating whether this is a regression or classification problem.
control	A list of additional arguments passed to the control_rwnn function.

Details

A deep RWNN is constructed by increasing the number of elements in the vector n_hidden. Furthermore, if type is null, then the function tries to deduce it from class of target.

Value

An RWNN-object.

References

Schmidt W., Kraaijveld M., Duin R. (1992) "Feedforward neural networks with random weights." *In Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, 1–4.

Pao Y., Park G., Sobajic D. (1992) "Learning and generalization characteristics of random vector Functional-link net." *Neurocomputing*, 6, 163–180.

Huang G.B., Zhu Q.Y., Siew C.K. (2006) "Extreme learning machine: Theory and applications." *Neurocomputing*, 70(1), 489–501.

Henríquez P.A., Ruz G.A. (2018) "Twitter Sentiment Classification Based on Deep Random Vector Functional Link." *In 2018 International Joint Conference on Neural Networks (IJCNN)*, 1–6.

Examples

```
## Models with a single hidden layer
n_hidden <- 50
lambda <- 0.01

# Regression
m <- rwnn(y ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda)

# Classification
m <- rwnn(I(y > median(y)) ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda)

## Model with multiple hidden layers
n_hidden <- c(20, 15, 10, 5)
lambda <- 0.01
```

```
# Combining outputs from all hidden layers (default)
m <- rwnn(y ~ ., data = example_data, n_hidden = n_hidden, lambda = lambda)

# Using only the output of the last hidden layer
m <- rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
          lambda = lambda, control = list(combine_hidden = FALSE))
```

`RWNN-object` *An RWNN-object*

Description

An RWNN-object is a list containing the following:

`data` The original data used to estimate the weights.

`n_hidden` The vector of neurons in each layer.

`activation` The vector of the activation functions used in each layer.

`lnorm` The norm used when estimating the output weights.

`lambda` The penalisation constant used when estimating the output weights.

`bias` The `TRUE/FALSE` bias vectors set by the control function for both hidden layers, and the output layer.

`weights` The weigths of the neural network, split into random (stored in hidden) and estimated (stored in output) weights.

`sigma` The standard deviation of the corresponding linear model.

`type` A string indicating the type of modelling problem.

`combined` A list of two `TRUE/FALSE` values stating whether the direct links were made to the input, and whether the output of each hidden layer was combined to make the prediction.

`stack_rwnn` *Stacking random weight neural networks*

Description

Use stacking to create ensemble random weight neural networks.

Usage

```
stack_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  B = 100,
  optimise = FALSE,
  folds = 10,
  method = NULL,
  type = NULL,
  control = list()
)

## S3 method for class 'formula'
stack_rwnn(
  formula,
  data = NULL,
  n_hidden = c(),
  lambda = NULL,
  B = 100,
  optimise = FALSE,
  folds = 10,
  method = NULL,
  type = NULL,
  control = list()
)
```

Arguments

formula	A formula specifying features and targets used to estimate the parameters of the output layer.
data	A data-set (either a data.frame or a tibble) used to estimate the parameters of the output layer.
n_hidden	A vector of integers designating the number of neurons in each of the hidden layers (the length of the list is taken as the number of hidden layers).
lambda	The penalisation constant(s) passed to either rwnn or ae_rwnn (see method argument).
B	The number of models in the stack.
optimise	TRUE/FALSE: Should the stacking weights be optimised (or should the stack just predict the average)?
folds	The number of folds used when optimising the stacking weights (see optimise argument).
method	The penalisation type passed to ae_rwnn. Set to NULL (default), "l1", or "l2". If NULL, rwnn is used as the base learner.
type	A string indicating whether this is a regression or classification problem.
control	A list of additional arguments passed to the control_rwnn function.

Value

An ERWNN-object.

References

Wolpert D. (1992) "Stacked generalization." *Neural Networks*, 5, 241-259.

Breiman L. (1996) "Stacked regressions." *Machine Learning*, 24, 49-64.

Examples

```
n_hidden <- c(20, 15, 10, 5)
lambda <- 0.01
B <- 100

## Using the average of the stack to predict new targets

m <- stack_rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
                lambda = lambda, B = B)


## Using the optimised weighting of the stack to predict new targets

m <- stack_rwnn(y ~ ., data = example_data, n_hidden = n_hidden,
                lambda = lambda, B = B, optimise = TRUE)
```

Index

∗ **datasets**
example_data, 11

ae_rwnn, 2, *4*, *6*, *10*, *19*

bag_rwnn, 3, *8*
boost_rwnn, 5, *12*

classify, 7, *12, 13*
control_rwnn, *3, 4*, *6*, 7, *10*, *17*, *19*
cor, *14*

data.frame, *3, 4*, *6*, *10*, *12, 13*, *17*, *19*

ed_rwnn, 9
ERWNN-object, *4*, *6*, *8*, *10*, 11, *12*, *14*, *20*
example_data, 11

formula, *3, 4*, *6*, *10*, *17*, *19*

halton, *9*

matrix, *12, 13*

predict.ERWNN, 12
predict.RWNN, 13

reduce_network, 13
rwnn, *4*, *6, 7*, *10*, 16, *19*
RWNN-object, *3*, *11*, *13, 14*, *17*, 18

sobol, *9*
stack_rwnn, 18

tibble, *3, 4*, *6*, *10*, *17*, *19*
torus, *9*