Package 'NeuralNetTools'

January 6, 2022

Type Package

Title Visualization and Analysis Tools for Neural Networks

Version 1.5.3

Date 2022-01-06

Author Marcus W. Beck [aut, cre]

Maintainer Marcus W. Beck <mbafs2012@gmail.com>

Description Visualization and analysis tools to aid in the interpretation of neural network models. Functions are available for plotting, quantifying variable importance, conducting a sensitivity analysis, and obtaining a simple list of model weights.

BugReports https://github.com/fawda123/NeuralNetTools/issues

License CC0

LazyData true

Imports ggplot2 (>= 2.0.0), nnet, reshape2, scales, tidyr

Suggests caret, neuralnet, RSNNS, knitr, rmarkdown

Depends R (>= 3.1.1)

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-01-06 15:30:02 UTC

R topics documented:

`bias_lines` *Plot connection weights for bias lines*

Description

Plot connection weights for bias lines in `plotnet`

Usage

```
bias_lines(
  bias_x,
  bias_y,
  mod_in,
  nid,
  rel_rsc,
  all_out,
  pos_col,
  neg_col,
  struct,
  y_names,
  x_range,
  y_range,
  layer_x,
  line_stag,
  max_sp
)
```

Arguments

`bias_x`	numeric vector x axis locations for bias lines
`bias_y`	numeric vector y axis locations for bias lines
`mod_in`	neural network model object
`nid`	logical value indicating if neural interpretation diagram is plotted, default `TRUE`
`rel_rsc`	numeric indicating the scaling range for the width of connection weights in a neural interpretation diagram. Default is `NULL` for no rescaling.

all_out	chr string indicating names of response variables for which connections are plotted, default all
pos_col	chr string indicating color of positive connection weights, default 'black'
neg_col	chr string indicating color of negative connection weights, default 'grey'
struct	numeric vector for network structure
y_names	chr string for names of output variables
x_range	numeric of x axis range for base plot
y_range	numeric of x axis range for base plot
layer_x	numeric indicating locations of layers on x axis
line_stag	numeric value that specifies distance of connection weights from nodes
max_sp	logical indicating if space is maximized in plot

---

bias_points *Plot bias points*

Description

Plot bias points in plotnet

Usage

```
bias_points(
  bias_x,
  bias_y,
  layer_name,
  node_labs,
  x_range,
  y_range,
  circle_cex,
  cex_val,
  bord_col,
  circle_col
)
```

Arguments

bias_x	numeric vector of values for x locations
bias_y	numeric vector for y location
layer_name	string indicating text to put in node
node_labs	logical indicating of node labels are included
x_range	numeric of x axis range for base plot
y_range	numeric of y axis range for base plot
circle_cex	numeric value indicating size of nodes, default 5

cex_val	numeric value indicating size of text labels, default 1
bord_col	chr string indicating border color around nodes, default 'lightblue'
circle_col	chr string indicating color of nodes

---

garson *Variable importance using Garson's algorithm*

Description

Relative importance of input variables in neural networks using Garson's algorithm

Usage

```
garson(mod_in, ...)

## Default S3 method:
garson(
  mod_in,
  x_names,
  y_names,
  bar_plot = TRUE,
  x_lab = NULL,
  y_lab = NULL,
  ...
)

## S3 method for class 'numeric'
garson(mod_in, struct, ...)

## S3 method for class 'nnet'
garson(mod_in, ...)

## S3 method for class 'mlp'
garson(mod_in, ...)

## S3 method for class 'nn'
garson(mod_in, ...)

## S3 method for class 'train'
garson(mod_in, ...)
```

Arguments

mod_in	input model object or a list of model weights as returned from neuralweights if using the default method
...	arguments passed to other methods

x_names	chr string of input variable names, obtained from the model object
y_names	chr string of response variable names, obtained from the model object
bar_plot	logical indicating if a ggplot object is returned (default T), otherwise numeric values are returned
x_lab	chr string of alternative names to be used for explanatory variables in the figure, default is taken from mod_in
y_lab	chr string of alternative name to be used for the y-axis in the figure
struct	numeric vector equal in length to the number of layers in the network. Each number indicates the number of nodes in each layer starting with the input and ending with the output. An arbitrary number of hidden layers can be included.

Details

The weights that connect variables in a neural network are partially analogous to parameter coefficients in a standard regression model and can be used to describe relationships between variables. The weights dictate the relative influence of information that is processed in the network such that input variables that are not relevant in their correlation with a response variable are suppressed by the weights. The opposite effect is seen for weights assigned to explanatory variables that have strong positive or negative associations with a response variable. An obvious difference between a neural network and a regression model is that the number of weights is excessive in the former case. This characteristic is advantageous in that it makes neural networks very flexible for modeling non-linear functions with multiple interactions, although interpretation of the effects of specific variables is of course challenging.

A method described in Garson 1991 (also see Goh 1995) identifies the relative importance of explanatory variables for a single response variables in a supervised neural network by deconstructing the model weights. The relative importance (or strength of association) of a specific explanatory variable for the response variable can be determined by identifying all weighted connections between the nodes of interest. That is, all weights connecting the specific input node that pass through the hidden layer to the response variable are identified. This is repeated for all other explanatory variables until a list of all weights that are specific to each input variable is obtained. The connections are tallied for each input node and scaled relative to all other inputs. A single value is obtained for each explanatory variable that describes the relationship with the response variable in the model (see the appendix in Goh 1995 for a more detailed description). The original algorithm indicates relative importance as the absolute magnitude from zero to one such the direction of the response cannot be determined.

Misleading results may be produced if the neural network was created with a skip-layer using skip = TRUE with the nnet or train functions. Garson's algorithm does not describe the effects of skip layer connections on estimates of variable importance. As such, these values are removed prior to estimating variable importance.

The algorithm currently only works for neural networks with one hidden layer and one response variable.

Value

A ggplot object for plotting if bar_plot = FALSE, otherwise a data.frame of relative importance values for each input variable. The default aesthetics for ggplot can be further modified, as shown with the examples.

References

Beck, M.W. 2018. NeuralNetTools: Visualization and Analysis Tools for Neural Networks. Journal of Statistical Software. 85(11):1-20.

Garson, G.D. 1991. Interpreting neural network connection weights. Artificial Intelligence Expert. 6(4):46-51.

Goh, A.T.C. 1995. Back-propagation neural networks for modeling complex systems. Artificial Intelligence in Engineering. 9(3):143-151.

Olden, J.D., Jackson, D.A. 2002. Illuminating the 'black-box': a randomization approach for understanding variable contributions in artificial neural networks. Ecological Modelling. 154:135-150.

Olden, J.D., Joy, M.K., Death, R.G. 2004. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. Ecological Modelling. 178:389-397.

See Also

olden for a more flexible approach for variable importance

Examples

```
## using numeric input

wts_in <- c(13.12, 1.49, 0.16, -0.11, -0.19, -0.16, 0.56, -0.52, 0.81)
struct <- c(2, 2, 1) #two inputs, two hidden, one output

garson(wts_in, struct)

## using nnet

library(nnet)

data(neuraldat)
set.seed(123)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

garson(mod)

## Not run:
## using RSNNS, no bias layers

library(RSNNS)

x <- neuraldat[, c('X1', 'X2', 'X3')]
y <- neuraldat[, 'Y1']
mod <- mlp(x, y, size = 5)

garson(mod)

## using neuralnet
```

```
library(neuralnet)

mod <- neuralnet(Y1 ~ X1 + X2 + X3, data = neuraldat, hidden = 5)

garson(mod)

## using caret

library(caret)

mod <- train(Y1 ~ X1 + X2 + X3, method = 'nnet', data = neuraldat, linout = TRUE)

garson(mod)

## modify the plot using ggplot2 syntax
library(ggplot2)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

cols <- heat.colors(10)
garson(mod) +
  scale_y_continuous('Rel. Importance', limits = c(-1, 1)) +
  scale_fill_gradientn(colours = cols) +
  scale_colour_gradientn(colours = cols)

## End(Not run)
```

`get_ys` *Get y locations for layers in* `plotnet`

Description

Get y locations for input, hidden, output layers in `plotnet`

Usage

```
get_ys(lyr, max_sp, struct, y_range)
```

Arguments

`lyr`	numeric indicating layer for getting y locations
`max_sp`	logical indicating if space is maximized in plot
`struct`	numeric vector for network structure
`y_range`	numeric vector indicating limits of y axis

layer_lines *Plot connection weights*

Description

Plot connection weights in plotnet

Usage

```
layer_lines(
  mod_in,
  h_layer,
  layer1 = 1,
  layer2 = 2,
  out_layer = FALSE,
  nid,
  rel_rsc,
  all_in,
  pos_col,
  neg_col,
  x_range,
  y_range,
  line_stag,
  x_names,
  layer_x,
  struct,
  max_sp,
  prune_col = NULL,
  prune_lty = "dashed",
  skip
)
```

Arguments

mod_in	neural network model object
h_layer	numeric indicating which connections to plot for the layer
layer1	numeric indicating order of first layer (for multiple hiden layers)
layer2	numeric indicating order of second layer (for multiple hiden layers)
out_layer	logical indicating if the lines are for the output layer
nid	logical value indicating if neural interpretation diagram is plotted, default TRUE
rel_rsc	numeric indicating the scaling range for the width of connection weights in a neural interpretation diagram. Default is NULL for no rescaling.
all_in	chr string indicating names of input variables for which connections are plotted, default all
pos_col	chr string indicating color of positive connection weights, default 'black'

neg_col	chr string indicating color of negative connection weights, default 'grey'
x_range	numeric of x axis range for base plot
y_range	numeric of y axis range for base plot
line_stag	numeric value that specifies distance of connection weights from nodes
x_names	chr string for names of input variables
layer_x	numeric indicating locations of layers on x axis
struct	numeric vector for network structure
max_sp	logical indicating if space is maximized in plot
prune_col	chr string indicating color of pruned connections, otherwise not shown
prune_lty	line type for pruned connections, passed to segments
skip	logical to plot connections for skip layer

---

layer_points *Plot neural network nodes*

Description

Plot neural network nodes in plotnet

Usage

```
layer_points(
  layer,
  x_loc,
  x_range,
  layer_name,
  cex_val,
  circle_cex,
  bord_col,
  in_col,
  node_labs,
  line_stag,
  var_labs,
  x_names,
  y_names,
  ...
)
```

Arguments

layer	specifies which layer, integer from struct
x_loc	indicates x location for layer, integer from layer_x
x_range	numeric for total range of x-axis

layer_name	string indicating text to put in node
cex_val	numeric indicating size of point text
circle_cex	numeric indicating size of circles
bord_col	chr string indicating border color around nodes, default lightblue
in_col	chr string indicating interior color of nodes
node_labs	logical indicating if node labels are to be plotted
line_stag	numeric indicating distance between of text from nodes
var_labs	chr string for variable labels
x_names	chr string for alternative names of input nodes
y_names	chr string for alternative names of output nodes
...	values passed to get_ys

---

lekgrps *Create optional barplot for* lekprofile *groups*

Description

Create optional barplot of constant values of each variable for each group used with lekprofile

Usage

```
lekgrps(grps, position = "dodge", grp_nms = NULL)
```

Arguments

grps	data.frame of values for each variable in each group used to create groups in lekprofile
position	chr string indicating bar position (e.g., 'dodge', 'fill', 'stack'), passed to geom_bar
grp_nms	optional chr string of alternative names for groups in legend

Value

A ggplot object

Examples

```
## enters used with kmeans clustering
x <- neuraldat[, c('X1', 'X2', 'X3')]
grps <- kmeans(x, 6)$center

lekgrps(grps)
```

lekprofile *Sensitivity analysis using Lek's profile method*

Description

Conduct a sensitivity analysis of model responses in a neural network to input variables using Lek's profile method

Usage

```
lekprofile(mod_in, ...)

## Default S3 method:
lekprofile(
  mod_in,
  xvars,
  ysel = NULL,
  xsel = NULL,
  steps = 100,
  group_vals = seq(0, 1, by = 0.2),
  val_out = FALSE,
  group_show = FALSE,
  grp_nms = NULL,
  position = "dodge",
  ...
)

## S3 method for class 'nnet'
lekprofile(mod_in, xsel = NULL, ysel = NULL, ...)

## S3 method for class 'mlp'
lekprofile(mod_in, xvars, yvars, xsel = NULL, ysel = NULL, ...)

## S3 method for class 'train'
lekprofile(mod_in, xsel = NULL, ysel = NULL, ...)

## S3 method for class 'nn'
lekprofile(mod_in, xsel = NULL, ysel = NULL, ...)
```

Arguments

mod_in	input object for which an organized model list is desired. The input can be an object of class nnet or mlp
...	arguments passed to other methods
xvars	data.frame of explanatory variables used to create the input model, only needed for mlp objects

ysel	chr string indicating which response variables to plot if more than one, defaults to all
xsel	chr string of names of explanatory variables to plot, defaults to all
steps	numeric value indicating number of observations to evaluate for each explanatory variable from minimum to maximum value, default 100
group_vals	numeric vector with values from 0-1 indicating quantile values at which to hold other explanatory variables constant or a single value indicating number of clusters to define grouping scheme, see details
val_out	logical value indicating if actual sensitivity values are returned rather than a plot, default FALSE
group_show	logical if a barplot is returned that shows the values at which explanatory variables were held constant while not being evaluated
grp_nms	optional chr string of alternative names for groups in legend
position	chr string indicating bar position (e.g., 'dodge', 'fill', 'stack'), passed to geom_bar, used if group_show = TRUE
yvars	data.frame of explanatory variables used to create the input model, only needed for mlp objects

Details

The Lek profile method is described briefly in Lek et al. 1996 and in more detail in Gevrey et al. 2003. The profile method is fairly generic and can be extended to any statistical model in R with a predict method. However, it is one of few methods used to evaluate sensitivity in neural networks.

The profile method can be used to evaluate the effect of explanatory variables by returning a plot of the predicted response across the range of values for each separate variable. The original profile method evaluated the effects of each variable while holding the remaining explanatory variables at different quantiles (e.g., minimum, 20th percentile, maximum). This is implemented in in the function by creating a matrix of values for explanatory variables where the number of rows is the number of observations and the number of columns is the number of explanatory variables. All explanatory variables are held at their mean (or other constant value) while the variable of interest is sequenced from its minimum to maximum value across the range of observations. This matrix (or data frame) is then used to predict values of the response variable from a fitted model object. This is repeated for each explanatory variable to obtain all response curves. Values passed to group_vals must range from zero to one to define the quantiles for holding unevaluated explanatory variables.

An alternative implementation of the profile method is to group the unevaluated explanatory variables using groupings defined by the statistical properties of the data. Covariance among predictors may present unlikely scenarios if holding all unevaluated variables at the same level. To address this issue, the function provides an option to hold unevaluated variable at mean values defined by natural clusters in the data. kmeans clustering is used on the input data.frame of explanatory variables if the argument passed to group_vals is an integer value greater than one. The centers of the clusters are then used as constant values for the unevaluated variables. An arbitrary grouping scheme can also be passed to group_vals as a data.frame where the user can specify exact values for holding each value constant (see the examples).

For all plots, the legend with the 'Groups' label indicates the colors that correspond to each group. The groups describe the values at which unevaluated explanatory variables were held constant,

either as specific quantiles, group assignments based on clustering, or in the arbitrary grouping defined by the user. The constant values of each explanatory variable for each group can be viewed as a barplot by using `group_show = TRUE`.

Note that there is no predict method for neuralnet objects from the nn package. The lekprofile method for nn objects uses the nnet package to recreate the input model, which is then used for the sensitivity predictions. This approach only works for networks with one hidden layer.

Value

A `ggplot` object for plotting if `val_out = FALSE`, otherwise a two-element `list` is returned with a `data.frame` in long form showing the predicted responses at different values of the explanatory variables and the grouping scheme that was used to hold unevaluated variables constant.

References

Beck, M.W. 2018. NeuralNetTools: Visualization and Analysis Tools for Neural Networks. Journal of Statistical Software. 85(11):1-20.

Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., Aulagnier, S. 1996. Application of neural networks to modelling nonlinear relationships in Ecology. Ecological Modelling. 90:39-52.

Gevrey, M., Dimopoulos, I., Lek, S. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological Modelling. 160:249-264.

Olden, J.D., Joy, M.K., Death, R.G. 2004. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. Ecological Modelling. 178:389-397.

Examples

```
## using nnet

library(nnet)

set.seed(123)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

lekprofile(mod)

## Not run:
## using RSNNS, no bias layers

library(RSNNS)

x <- neuraldat[, c('X1', 'X2', 'X3')]
y <- neuraldat[, 'Y1', drop = FALSE]

mod <- mlp(x, y, size = 5)

lekprofile(mod, xvars = x)
```

```
## using neuralnet

library(neuralnet)

mod <- neuralnet(Y1 ~ X1 + X2 + X3, data = neuraldat, hidden = 5)

lekprofile(mod)

## back to nnet, not using formula to create model
## y variable must have a name attribute

mod <- nnet(x, y, size = 5)

lekprofile(mod)

## using caret

library(caret)

mod <- train(Y1 ~ X1 + X2 + X3, method = 'nnet', data = neuraldat, linout = TRUE)

lekprofile(mod)

## group by clusters instead of sequencing by quantiles

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

lekprofile(mod, group_vals = 6) # six clusters

## enter an arbitrary grouping scheme for the group values
## i.e. hold all values at 0.5
group_vals <- rbind(rep(0.5, length = ncol(x)))
group_vals <- data.frame(group_vals)
names(group_vals) <- names(group_vals)

lekprofile(mod, group_vals = group_vals, xsel = 'X3')

## End(Not run)
```

`neuraldat` *Simulated dataset for function examples*

Description

A simulated dataset of 2000 observations containing two response variables and three explanatory variables. Explanatory variables were sampled from a standard normal distribution. Response variables were linear combinations of the explanatory variables. The response variables Y1 and Y2 are standardized from 0 to 1.

Usage

```
neuraldat
```

Format

A data frame with 2000 rows and 5 variables:

Y1 numeric

Y2 numeric

X1 numeric

X2 numeric

X3 numeric ...

`neuralskips` *Get weights for the skip layer in a neural network*

Description

Get weights for the skip layer in a neural network, only valid for networks created using `skip = TRUE` with the `nnet` function.

Usage

```
neuralskips(mod_in, ...)

## S3 method for class 'nnet'
neuralskips(mod_in, rel_rsc = NULL, ...)
```

Arguments

`mod_in`	input object for which an organized model list is desired.
`...`	arguments passed to other methods
`rel_rsc`	numeric indicating the scaling range for the width of connection weights in a neural interpretation diagram. Default is `NULL` for no rescaling. Scaling is relative to all weights, not just those in the primary network.

Details

This function is similar to `neuralweights` except only the skip layer weights are returned.

Value

Returns a list of connections for each output node, where each element of the list is the connection for each input node in sequential order to the respective output node. The first weight in each element is not the bias connection, unlike the results for `neuralweights`.

Examples

```
data(neuraldat)
set.seed(123)

## using nnet

library(nnet)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5, linout = TRUE,
 skip = TRUE)

neuralskips(mod)
```

`neuralweights` *Get weights for a neural network*

Description

Get weights for a neural network in an organized list by extracting values from a neural network object. This function is generally not called by itself.

Usage

```
neuralweights(mod_in, ...)

## S3 method for class 'numeric'
neuralweights(mod_in, rel_rsc = NULL, struct, ...)

## S3 method for class 'nnet'
neuralweights(mod_in, rel_rsc = NULL, ...)

## S3 method for class 'mlp'
neuralweights(mod_in, rel_rsc = NULL, ...)

## S3 method for class 'nn'
neuralweights(mod_in, rel_rsc = NULL, ...)
```

Arguments

`mod_in`	input object for which an organized model list is desired. The input can be an object of class `numeric`, `nnet`, `mlp`, or `nn`
`...`	arguments passed to other methods
`rel_rsc`	numeric indicating the scaling range for the width of connection weights in a neural interpretation diagram. Default is `NULL` for no rescaling.

`struct` numeric vector equal in length to the number of layers in the network. Each number indicates the number of nodes in each layer starting with the input and ending with the output. An arbitrary number of hidden layers can be included.

Details

Each element of the returned list is named using the construct 'layer node', e.g. 'out 1' is the first node of the output layer. Hidden layers are named using three values for instances with more than one hidden layer, e.g., 'hidden 1 1' is the first node in the first hidden layer, 'hidden 1 2' is the second node in the first hidden layer, 'hidden 2 1' is the first node in the second hidden layer, etc. The values in each element of the list represent the weights entering the specific node from the preceding layer in sequential order, starting with the bias layer if applicable. For example, the elements in a list item for 'hidden 1 1' of a neural network with a 3 5 1 structure (3 inputs, 5 hidden nodes, 1 output) would have four values indicating the weights in sequence from the bias layer, first input layer, second input layer, and third input layer going to the first hidden node.

The function will remove direct weight connections between input and output layers if the neural network was created with a skip-layer using `skip = TRUE` with the `nnet` or `train` functions. This may produce misleading results when evaluating variable performance with the `garson` function.

Value

Returns a two-element list with the first element being a vector indicating the number of nodes in each layer of the neural network and the second element being a named list of weight values for the input model.

Examples

```
data(neuraldat)
set.seed(123)

## using numeric input

wts_in <- c(13.12, 1.49, 0.16, -0.11, -0.19, -0.16, 0.56, -0.52, 0.81)
struct <- c(2, 2, 1) #two inputs, two hidden, one output

neuralweights(wts_in, struct = struct)

## using nnet

library(nnet)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5, linout = TRUE)

neuralweights(mod)

## Not run:
## using RSNNS, no bias layers

library(RSNNS)
```

```
x <- neuraldat[, c('X1', 'X2', 'X3')]
y <- neuraldat[, 'Y1']
mod <- mlp(x, y, size = 5, linOut = TRUE)

neuralweights(mod)

# pruned model using code from RSSNS pruning demo
pruneFuncParams <- list(max_pr_error_increase = 10.0, pr_accepted_error = 1.0,
 no_of_pr_retrain_cycles = 1000, min_error_to_stop = 0.01, init_matrix_value = 1e-6,
 input_pruning = TRUE, hidden_pruning = TRUE)
mod <- mlp(x, y, size = 5, pruneFunc = "OptimalBrainSurgeon",
 pruneFuncParams = pruneFuncParams)

neuralweights(mod)

## using neuralnet

library(neuralnet)

mod <- neuralnet(Y1 ~ X1 + X2 + X3, data = neuraldat, hidden = 5)

neuralweights(mod)

## End(Not run)
```

---

olden *Variable importance using connection weights*

Description

Relative importance of input variables in neural networks as the sum of the product of raw input-hidden, hidden-output connection weights, proposed by Olden et al. 2004.

Usage

```
olden(mod_in, ...)

## Default S3 method:
olden(
  mod_in,
  x_names,
  y_names,
  out_var = NULL,
  bar_plot = TRUE,
  x_lab = NULL,
  y_lab = NULL,
  skip_wts = NULL,
  ...
)
```

```
## S3 method for class 'numeric'
olden(mod_in, struct, ...)

## S3 method for class 'nnet'
olden(mod_in, ...)

## S3 method for class 'mlp'
olden(mod_in, ...)

## S3 method for class 'nn'
olden(mod_in, ...)

## S3 method for class 'train'
olden(mod_in, ...)
```

Arguments

mod_in	input model object or a list of model weights as returned from neuralweights if using the default method
...	arguments passed to or from other methods
x_names	chr string of input variable names, obtained from the model object
y_names	chr string of response variable names, obtained from the model object
out_var	chr string indicating the response variable in the neural network object to be evaluated. Only one input is allowed for models with more than one response. Names must be of the form 'Y1', 'Y2', etc. if using numeric values as weight inputs for mod_in.
bar_plot	logical indicating if a ggplot object is returned (default T), otherwise numeric values are returned
x_lab	chr string of alternative names to be used for explanatory variables in the figure, default is taken from mod_in
y_lab	chr string of alternative names to be used for response variable in the figure, default is taken from out_var
skip_wts	vector from neuralskips for nnet models with skip-layer connections
struct	numeric vector equal in length to the number of layers in the network. Each number indicates the number of nodes in each layer starting with the input and ending with the output. An arbitrary number of hidden layers can be included.

Details

This method is similar to Garson's algorithm (Garson 1991, modified by Goh 1995) in that the connection weights between layers of a neural network form the basis for determining variable importance. However, Olden et al. 2004 describe a connection weights algorithm that consistently out-performed Garson's algorithm in representing the true variable importance in simulated datasets. This 'Olden' method calculates variable importance as the product of the raw input-hidden and hidden-output connection weights between each input and output neuron and sums the product across all hidden neurons. An advantage of this approach is the relative contributions of each

connection weight are maintained in terms of both magnitude and sign as compared to Garson's algorithm which only considers the absolute magnitude. For example, connection weights that change sign (e.g., positive to negative) between the input-hidden to hidden-output layers would have a cancelling effect whereas Garson's algorithm may provide misleading results based on the absolute magnitude. An additional advantage is that Olden's algorithm is capable of evaluating neural networks with multiple hidden layers wheras Garson's was developed for networks with a single hidden layer.

The importance values assigned to each variable are in units that are based directly on the summed product of the connection weights. The actual values should only be interpreted based on relative sign and magnitude between explanatory variables. Comparisons between different models should not be made.

The Olden function also works with networks that have skip layers by adding the input-output connection weights to the final summed product of all input-hidden and hidden-output connections. This was not described in the original method so interpret with caution.

By default, the results are shown only for the first response variable for networks with multiple output nodes. The plotted response variable can be changed with `out_var`.

Value

A `ggplot` object for plotting if `bar_plot = FALSE`, otherwise a `data.frame` of relative importance values for each input variable.

References

Beck, M.W. 2018. NeuralNetTools: Visualization and Analysis Tools for Neural Networks. Journal of Statistical Software. 85(11):1-20.

Garson, G.D. 1991. Interpreting neural network connection weights. Artificial Intelligence Expert. 6(4):46-51.

Goh, A.T.C. 1995. Back-propagation neural networks for modeling complex systems. Artificial Intelligence in Engineering. 9(3):143-151.

Olden, J.D., Jackson, D.A. 2002. Illuminating the 'black-box': a randomization approach for understanding variable contributions in artificial neural networks. Ecological Modelling. 154:135-150.

Olden, J.D., Joy, M.K., Death, R.G. 2004. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. Ecological Modelling. 178:389-397.

Examples

```
## using numeric input

wts_in <- c(13.12, 1.49, 0.16, -0.11, -0.19, -0.16, 0.56, -0.52, 0.81)
struct <- c(2, 2, 1) #two inputs, two hidden, one output

olden(wts_in, struct)

## using nnet
```

```
library(nnet)

data(neuraldat)
set.seed(123)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

olden(mod)

## Not run:
## View the difference for a model w/ skip layers

set.seed(123)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5, skip = TRUE)

olden(mod)

## using RSNNS, no bias layers

library(RSNNS)

x <- neuraldat[, c('X1', 'X2', 'X3')]
y <- neuraldat[, 'Y1']
mod <- mlp(x, y, size = 5)

olden(mod)

## using neuralnet

library(neuralnet)

mod <- neuralnet(Y1 ~ X1 + X2 + X3, data = neuraldat, hidden = 5)

olden(mod)

## using caret

library(caret)

mod <- train(Y1 ~ X1 + X2 + X3, method = 'nnet', data = neuraldat, linout = TRUE)

olden(mod)

## multiple hidden layers

x <- neuraldat[, c('X1', 'X2', 'X3')]
y <- neuraldat[, 'Y1']
mod <- mlp(x, y, size = c(5, 7, 6), linOut = TRUE)

olden(mod)

## End(Not run)
```

plotnet *Plot a neural network model*

Description

Plot a neural interpretation diagram for a neural network object

Usage

```
plotnet(mod_in, ...)

## Default S3 method:
plotnet(
  mod_in,
  x_names,
  y_names,
  struct = NULL,
  nid = TRUE,
  all_out = TRUE,
  all_in = TRUE,
  bias = TRUE,
  bias_y = 0.95,
  rel_rsc = c(1, 7),
  circle_cex = 5,
  node_labs = TRUE,
  var_labs = TRUE,
  line_stag = NULL,
  cex_val = 1,
  alpha_val = 1,
  circle_col = "lightblue",
  pos_col = "black",
  neg_col = "grey",
  bord_col = "lightblue",
  max_sp = FALSE,
  pad_x = 1,
  prune_col = NULL,
  prune_lty = "dashed",
  skip = NULL,
  ...
)

## S3 method for class 'nnet'
plotnet(mod_in, x_names = NULL, y_names = NULL, skip = FALSE, ...)

## S3 method for class 'numeric'
plotnet(mod_in, struct, x_names = NULL, y_names = NULL, ...)
```

```
## S3 method for class 'mlp'
plotnet(
  mod_in,
  x_names = NULL,
  y_names = NULL,
  prune_col = NULL,
  prune_lty = "dashed",
  ...
)

## S3 method for class 'nn'
plotnet(mod_in, x_names = NULL, y_names = NULL, ...)

## S3 method for class 'train'
plotnet(mod_in, x_names = NULL, y_names = NULL, skip = FALSE, ...)
```

Arguments

mod_in	neural network object or numeric vector of weights
...	additional arguments passed to or from other methods
x_names	chr string indicating names for input variables, default from model object
y_names	chr string indicating names for output variables, default from model object
struct	numeric vector equal in length to the number of layers in the network. Each number indicates the number of nodes in each layer starting with the input and ending with the output. An arbitrary number of hidden layers can be included.
nid	logical value indicating if neural interpretation diagram is plotted, default TRUE
all_out	chr string indicating names of response variables for which connections are plotted, default all
all_in	chr string indicating names of input variables for which connections are plotted, default all
bias	logical value indicating if bias nodes and connections are plotted, default TRUE
bias_y	value from 0 to 1 for locattion of bias nodes on y-axis
rel_rsc	numeric indicating the scaling range for the width of connection weights
circle_cex	numeric value indicating size of nodes, default 5
node_labs	logical value indicating if labels are plotted directly on nodes, default TRUE
var_labs	logical value indicating if variable names are plotted next to nodes, default TRUE
line_stag	numeric value that specifies distance of connection weights from nodes
cex_val	numeric value indicating size of text labels, default 1
alpha_val	numeric value (0-1) indicating transparency of connections, default 1
circle_col	chr string indicating color of nodes, default 'lightblue', or two element list with first element indicating color of input nodes and second indicating color of remaining nodes
pos_col	chr string indicating color of positive connection weights, default 'black'

neg_col	chr string indicating color of negative connection weights, default 'grey'
bord_col	chr string indicating border color around nodes, default 'lightblue'
max_sp	logical value indicating if space between nodes in each layer is maximized, default FALSE
pad_x	numeric for increasing or decreasing padding on the x-axis, values less than one will increase padding and values greater than one will decrease padding
prune_col	chr string indicating color of pruned connections, otherwise not shown
prune_lty	line type for pruned connections, passed to segments
skip	logical if skip layer connections are plotted instead of the primary network

Details

This function plots a neural network as a neural interpretation diagram as in Ozesmi and Ozesmi (1999). Options to plot without color-coding or shading of weights are also provided. The default settings plot positive weights between layers as black lines and negative weights as grey lines. Line thickness is in proportion to relative magnitude of each weight. The first layer includes only input variables with nodes labelled arbitrarily as I1 through In for n input variables. One through many hidden layers are plotted with each node in each layer labelled as H1 through Hn. The output layer is plotted last with nodes labeled as O1 through On. Bias nodes connected to the hidden and output layers are also shown. Neural networks created using mlp do not show bias layers.

A primary network and a skip layer network can be plotted for nnet models with a skip layer connection. The default is to plot the primary network, whereas the skip layer network can be viewed with skip = TRUE. If nid = TRUE, the line widths for both the primary and skip layer plots are relative to all weights. Viewing both plots is recommended to see which network has larger relative weights. Plotting a network with only a skip layer (i.e., no hidden layer, size = 0) will include bias connections to the output layer, whereas these are not included in the plot of the skip layer if size is greater than zero.

The numeric method for plotting requires the input weights to be in a specific order given the structure of the network. An additional argument struct (from neuralweights is also required that lists the number of nodes in the input, hidden, and output layers. The example below for the numeric input shows the correct weight vector for a simple neural network model with two input variables, one output variable, and one hidden layer with two nodes. Bias nodes are also connected to the hidden and output layer. Using the plot syntax of I, H, O, and B for input, hidden, output, and bias to indicate weighted connections between layers, the correct weight order for the mod_in vector is B1-H1, I1-H1, I2-H1, B1-H2, I1-H2, I2-H2, B2-O1, H1-O1, H2-O1. For a generic network (three layers) with n input nodes, j hidden nodes, and k output nodes, the weights are ordered as the connections from B1, I1,...,In to H1,...,Hj, then B2, H1,..,Hj to O1,...,Ok.

Value

A graphics object unless wts_only = TRUE, then neural network weights from neuralweights.

References

Beck, M.W. 2018. NeuralNetTools: Visualization and Analysis Tools for Neural Networks. Journal of Statistical Software. 85(11):1-20.

Ozesmi, S.L., Ozesmi, U. 1999. An artificial neural network approach to spatial habitat modeling with interspecific interaction. Ecological Modelling. 116:15-31.

Examples

```
## using numeric input

# B1-H1, I1-H1, I2-H1, B1-H2, I1-H2, I2-H2, B2-O1, H1-O1, H2-O1.
wts_in <- c(13.12, 1.49, 0.16, -0.11, -0.19, -0.16, 0.56, -0.52, 0.81)
struct <- c(2, 2, 1) #two inputs, two hidden, one output

plotnet(wts_in, struct = struct)

# numeric input, two hidden layers

# B1-H11, I1-H11, I2-H11, B1-H12, I1-H12, I2-H12, B2-H21, H11-H21, H12-H21,
# B2-H22, H11-H22, H12-H22, B3-O1, H21-O1, H22-O1
wts_in <- c(1.12, 1.49, 0.16, -0.11, -0.19, -0.16, 0.5, 0.2, -0.12, -0.1,
 0.89, 0.9, 0.56, -0.52, 0.81)
struct <- c(2, 2, 2, 1) # two inputs, two (two nodes each), one output

plotnet(wts_in, struct = struct)

## using nnet

library(nnet)

data(neuraldat)
set.seed(123)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

plotnet(mod)

## plot the skip layer from nnet model

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5, skip = TRUE)

plotnet(mod, skip = TRUE)

## Not run:
## using RSNNS, no bias layers

library(RSNNS)

x <- neuraldat[, c('X1', 'X2', 'X3')]
y <- neuraldat[, 'Y1']
mod <- mlp(x, y, size = 5)

plotnet(mod)

# pruned model using code from RSSNS pruning demo
pruneFuncParams <- list(max_pr_error_increase = 10.0, pr_accepted_error = 1.0,
```

```
 no_of_pr_retrain_cycles = 1000, min_error_to_stop = 0.01, init_matrix_value = 1e-6,
 input_pruning = TRUE, hidden_pruning = TRUE)
mod <- mlp(x, y, size = 5, pruneFunc = "OptimalBrainSurgeon",
 pruneFuncParams = pruneFuncParams)

plotnet(mod)
plotnet(mod, prune_col = 'lightblue')

## using neuralnet

library(neuralnet)

mod <- neuralnet(Y1 ~ X1 + X2 + X3, data = neuraldat, hidden = 5)

plotnet(mod)

## using caret

library(caret)

mod <- train(Y1 ~ X1 + X2 + X3, method = 'nnet', data = neuraldat, linout = TRUE)

plotnet(mod)

## a more complicated network with categorical response
AND <- c(rep(0, 7), 1)
OR <- c(0, rep(1, 7))

binary_data <- data.frame(expand.grid(c(0, 1), c(0, 1), c(0, 1)), AND, OR)

mod <- neuralnet(AND + OR ~ Var1 + Var2 + Var3, binary_data,
 hidden = c(6, 12, 8), rep = 10, err.fct = 'ce', linear.output = FALSE)

plotnet(mod)

## recreate the previous example with numeric inputs

# get the weights and structure in the right format
wts <- neuralweights(mod)
struct <- wts$struct
wts <- unlist(wts$wts)

# plot
plotnet(wts, struct = struct)

## color input nodes by relative importance
mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

rel_imp <- garson(mod, bar_plot = FALSE)$rel_imp
cols <- colorRampPalette(c('lightgreen', 'darkgreen'))(3)[rank(rel_imp)]

plotnet(mod, circle_col = list(cols, 'lightblue'))
```

```
## End(Not run)
```

`pred_sens` *Predicted values for Lek profile method*

Description

Get predicted values for Lek Profile method, used iteratively in `lekprofile`

Usage

```
pred_sens(mat_in, mod_in, var_sel, step_val, grps, ysel)
```

Arguments

`mat_in`	`data.frame` of only the explanatory variables used to create model
`mod_in`	any model object with a predict method
`var_sel`	chr string of explanatory variable to select
`step_val`	number of values to sequence range of selected explanatory variable
`grps`	matrix of values for holding explanatory values constant, one column per variable and one row per group
`ysel`	chr string of response variable names for correct labelling

Details

Gets predicted output for a model's response variable based on matrix of explanatory variables that are restricted following Lek's profile method. The selected explanatory variable is sequenced across a range of values. All other explanatory variables are held constant at the values in `grps`.

Value

A `list` of predictions where each element is a `data.frame` with the predicted value of the response and the values of the explanatory variable defined by `var_sel`. Each element of the list corresponds to a group defined by the rows in `grps` at which the other explanatory variables were held constant.

See Also

lekprofile

Examples

```
## using nnet

library(nnet)

data(neuraldat)
set.seed(123)

mod <- nnet(Y1 ~ X1 + X2 + X3, data = neuraldat, size = 5)

mat_in <- neuraldat[, c('X1', 'X2', 'X3')]
grps <- apply(mat_in, 2, quantile, seq(0, 1, by = 0.2))

pred_sens(mat_in, mod, 'X1', 100, grps, 'Y1')
```

Index