Package ‘neuralnet’

February 20, 2015

Type Package
Title Training of neural networks
Version 1.32
Date 2012-09-19
Author Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>, following earlier work by Marc Suling
Maintainer Frauke Guenther <guenther@bips.uni-bremen.de>
Depends R (>= 2.9.0), grid, MASS
Description Training of neural networks using backpropagation, resilient backpropagation with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version by Anastasiadis et al. (2005). The package allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights (Intrator O & Intrator N, 1993) is implemented.
License GPL (>= 2)
Repository CRAN
Date/Publication 2012-09-20 09:35:16
NeedsCompilation no

R topics documented:

neuralnet-package ........................................... 2
compute ......................................................... 3
confidence.interval .......................................... 4
gwplot ......................................................... 6
neuralnet ....................................................... 7
plot.nn ......................................................... 10
prediction .................................................... 12

Index 13
neuralnet-package  

**Training of neural networks**

**Description**

Training of neural networks using the backpropagation, resilient backpropagation with (Riedmiller, 1994) or without weight backtracking (Riedmiller, 1993) or the modified globally convergent version by Anastasiadis et al. (2005). The package allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights (Intrator O & Intrator N, 1993) is implemented.

**Details**

<table>
<thead>
<tr>
<th>Package:</th>
<th>neuralnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.32</td>
</tr>
<tr>
<td>Date:</td>
<td>2012-09-19</td>
</tr>
<tr>
<td>License:</td>
<td>GPL (&gt;=2)</td>
</tr>
</tbody>
</table>

**Note**

This work has been supported by the German Research Foundation (DFG: http://www.dfg.de) under grant scheme PI 345/3-1.

**Author(s)**

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>, following earlier work by Marc Suling

Maintainer: Frauke Guenther <guenther@bips.uni-bremen.de>

**References**


Anastasiadis A. et. al. (2005) *New globally convergent training scheme based on the resilient propagation algorithm*. Neurocomputing 64, pages 253-270.

compute

See Also

plot.nn for plotting of the neural network.
gwplot for plotting of the generalized weights.
compute for computation of the calculated network.
confidence.interval for calculation of a confidence interval for the weights.
prediction for calculation of a prediction.

Examples

```
AND <- c(rep(0,7),1)
OR <- c(rep(0,7),1)
binary.data <- data.frame(expand.grid(c(0,1), c(0,1), c(0,1)), AND, OR)
print(net <- neuralnet(AND+OR~Var1+Var2+Var3, binary.data, hidden=0,
rep=10, err.fct="ce", linear.output=FALSE))

XOR <- c(0,1,1,0)
xor.data <- data.frame(expand.grid(c(0,1), c(0,1)), XOR)
print(net.xor <- neuralnet(XOR~Var1+Var2, xor.data, hidden=2, rep=5))
plot(net.xor, rep="best")

data(infert, package="datasets")
print(net.infert <- neuralnet(case~parity+induced+spontaneous, infert,
erfct="ce", linear.output=FALSE, likelihood=TRUE))
gwplot(net.infert, selected.covariate="parity")
gwplot(net.infert, selected.covariate="induced")
gwplot(net.infert, selected.covariate="spontaneous")
confidence.interval(net.infert)

var1 <- runif(10, 0, 100)
sqrt.data <- data.frame(var1, sqrt=sqrt(var1))
print(net.sqrt <- neuralnet(sqrt~var1, sqrt.data, hidden=10,
threshhold=0.01))
compute(net.sqrt, (1:10)^2)

Var1 <- rpois(100,0.5)
Var2 <- rbinom(100,2,0.6)
Var3 <- rbinom(100,1,0.5)
SUM <- as.integer(abs(Var1+Var2+Var3+(rnorm(100))))
sum.data <- data.frame(Var1+Var2+Var3, SUM)
print(net.sum <- neuralnet(SUM~Var1+Var2+Var3, sum.data, hidden=1,
act.fct="tanh")
prediction(net.sum)
```

compute

Computation of a given neural network for given covariate vectors
compute, a method for objects of class \texttt{nn}, typically produced by \texttt{neuralnet}. Computes the outputs of all neurons for specific arbitrary covariate vectors given a trained neural network. Please make sure that the order of the covariates is the same in the new matrix or dataframe as in the original neural network.

Usage

\begin{verbatim}
compute(x, covariate, rep = 1)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{x} an object of class \texttt{nn}.
\item \texttt{covariate} a dataframe or matrix containing the variables that had been used to train the neural network.
\item \texttt{rep} an integer indicating the neural network's repetition which should be used.
\end{itemize}

Value

\texttt{compute} returns a list containing the following components:

\begin{itemize}
\item \texttt{neurons} a list of the neurons' output for each layer of the neural network.
\item \texttt{net.result} a matrix containing the overall result of the neural network.
\end{itemize}

Author(s)

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>

Examples

\begin{verbatim}
Var1 <- runif(50, 0, 100)
sqrt.data <- data.frame(Var1, Sqrt=sqrt(Var1))
print(net.sqrt <- neuralnet(Sqrt~Var1, sqrt.data, hidden=10,
threshold=0.01))
compute(net.sqrt, (1:10)*2)$net.result
\end{verbatim}

confidence.interval

Calculates confidence intervals of the weights

Description

\texttt{confidence.interval}, a method for objects of class \texttt{nn}, typically produced by \texttt{neuralnet}. Calculates confidence intervals of the weights (White, 1989) and the network information criteria NIC (Murata et al. 1994). All confidence intervals are calculated under the assumption of a local identification of the given neural network. If this assumption is violated, the results will not be reasonable. Please make also sure that the chosen error function equals the negative log-likelihood function, otherwise the results are not meaningful, too.
Usage

confidence.interval(x, alpha = 0.05)

Arguments

x neural network
alpha numerical. Sets the confidence level to (1-alpha).

Value

confidence.interval returns a list containing the following components:

lower.ci a list containing the lower confidence bounds of all weights of the neural network differentiated by the repetitions.
upper.ci a list containing the upper confidence bounds of all weights of the neural network differentiated by the repetitions.
nic a vector containing the information criteria NIC for every repetition.

Author(s)

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>

References


Murata et al. (1994) *Network information criterion - determining the number of hidden units for an artificial neural network model*. IEEE Transactions on Neural Networks 5 (6), pages 865-871

See Also

neuralnet

Examples

data(infert, package="datasets")
print(net.infert <- neuralnet(case~parity+induced+spontaneous,
infert, err.fct="ce", linear.output=FALSE))
confidence.interval(net.infert)
gwplot, a method for objects of class nn, typically produced by neuralnet. Plots the generalized weights (Intrator and Intrator, 1993) for one specific covariate and one response variable.

Usage

```
 gwplot(x, rep = NULL, max = NULL, min = NULL, file = NULL, 
        selected.covariate = 1, selected.response = 1, 
        highlight = FALSE, type = "p", col = "black", ...) 
```

Arguments

- `x`: an object of class `nn`
- `rep`: an integer indicating the repetition to plot. If `rep="best"`, the repetition with the smallest error will be plotted. If not stated all repetitions will be plotted.
- `max`: maximum of the y axis. In default, max is set to the highest y-value.
- `min`: minimum of the y axis. In default, min is set to the smallest y-value.
- `file`: a character string naming the plot to write to. If not stated, the plot will not be saved.
- `selected.covariate`: either a string of the covariate’s name or an integer of the ordered covariates, indicating the reference covariate in the generalized weights plot. Defaulting to the first covariate.
- `selected.response`: either a string of the response variable’s name or an integer of the ordered response variables, indicating the reference response in the generalized weights plot. Defaulting to the first response variable.
- `highlight`: a logical value, indicating whether to highlight (red color) the best repetition (smallest error). Only reasonable if `rep=NULL`. Default is `FALSE`.
- `type`: a character indicating the type of plotting; actually any of the types as in `plot.default`.
- `col`: a color of the generalized weights.
- `...`: Arguments to be passed to methods, such as graphical parameters (see `par`).

Author(s)

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>

References

neuralnet

See Also

neuralnet

Examples

data(infert, package="datasets")
print(net.infert <- neuralnet(case~parity+induced+spontaneous, infert,
  err.fct="ce", linear.output=FALSE, likelihood=TRUE))
gwplot(net.infert, selected.covariate="parity")
gwplot(net.infert, selected.covariate="induced")
gwplot(net.infert, selected.covariate="spontaneous")

neuralnet

Training of neural networks

Description

neuralnet is used to train neural networks using backpropagation, resilient backpropagation (RPROP) with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version (GRPROP) by Anastasiadis et al. (2005). The function allows flexible settings through custom-choice of error and activation function. Furthermore the calculation of generalized weights (Intrator O. and Intrator N., 1993) is implemented.

Usage

neuralnet(formula, data, hidden = 1, threshold = 0.01,
  stepmax = 1e+05, rep = 1, startweights = NULL,
  learningrate.limit = NULL,
  learningrate.factor = list(minus = 0.5, plus = 1.2),
  learningrate=NULL, lifesign = "none",
  lifesign.step = 1000, algorithm = "rprop+",
  err.fct = "sse", act.fct = "logistic",
  linear.output = TRUE, exclude = NULL,
  constant.weights = NULL, likelihood = FALSE)

Arguments

formula a symbolic description of the model to be fitted.
data a data frame containing the variables specified in formula.
hidden a vector of integers specifying the number of hidden neurons (vertices) in each layer.
threshold a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria.
stepmax the maximum steps for the training of the neural network. Reaching this maximum leads to a stop of the neural network’s training process.
rep the number of repetitions for the neural network’s training.
**startweights**

A vector containing starting values for the weights. The weights will not be randomly initialized.

**learningrate.limit**

A vector or a list containing the lowest and highest limit for the learning rate. Used only for RPROP and GRPROP.

**learningrate.factor**

A vector or a list containing the multiplication factors for the upper and lower learning rate. Used only for RPROP and GRPROP.

**learningrate**

A numeric value specifying the learning rate used by traditional backpropagation. Used only for traditional backpropagation.

**lifesign**

A string specifying how much the function will print during the calculation of the neural network. 'none', 'minimal' or 'full'.

**lifesign.step**

An integer specifying the stepsize to print the minimal threshold in full lifesign mode.

**algorithm**

A string containing the algorithm type to calculate the neural network. The following types are possible: 'backprop', 'rprop+', 'rprop-', 'sag', or 'slr'. 'backprop' refers to backpropagation, 'rprop+' and 'rprop-' refer to the resilient backpropagation with and without weight backtracking, while 'sag' and 'slr' induce the usage of the modified globally convergent algorithm (grprop). See Details for more information.

**err.fct**

A differentiable function that is used for the calculation of the error. Alternatively, the strings 'sse' and 'ce' which stand for the sum of squared errors and the cross-entropy can be used.

**act.fct**

A differentiable function that is used for smoothing the result of the cross product of the covariate or neurons and the weights. Additionally the strings, 'logistic' and 'tanh' are possible for the logistic function and tangent hyperbolicus.

**linear.output**

Logical. If act.fct should not be applied to the output neurons set linear output to TRUE, otherwise to FALSE.

**exclude**

A vector or a matrix specifying the weights, that are excluded from the calculation. If given as a vector, the exact positions of the weights must be known. A matrix with n-rows and 3 columns will exclude n weights, where the first column stands for the layer, the second column for the input neuron and the third column for the output neuron of the weight.

**constant.weights**

A vector specifying the values of the weights that are excluded from the training process and treated as fix.

**likelihood**

Logical. If the error function is equal to the negative log-likelihood function, the information criteria AIC and BIC will be calculated. Furthermore the usage of confidence.interval is meaningful.

**Details**

The globally convergent algorithm is based on the resilient backpropagation without weight backtracking and additionally modifies one learning rate, either the learningrate associated with the smallest absolute gradient (sag) or the smallest learningrate (slr) itself. The learning rates in the grprop algorithm are limited to the boundaries defined in learningrate.limit.
**Value**

`neuralnet` returns an object of class `nn`. An object of class `nn` is a list containing at most the following components:

- **call**: the matched call.
- **response**: extracted from the `data` argument.
- **covariate**: the variables extracted from the `data` argument.
- **model.list**: a list containing the covariates and the response variables extracted from the `formula` argument.
- **err.fct**: the error function.
- **act.fct**: the activation function.
- **data**: the `data` argument.
- **net.result**: a list containing the overall result of the neural network for every repetition.
- **weights**: a list containing the fitted weights of the neural network for every repetition.
- **generalized.weights**: a list containing the generalized weights of the neural network for every repetition.
- **result.matrix**: a matrix containing the reached threshold, needed steps, error, AIC and BIC (if computed) and weights for every repetition. Each column represents one repetition.
- **startweights**: a list containing the startweights of the neural network for every repetition.

**Author(s)**

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>

**References**


Anastasiadis A. et. al. (2005) *New globally convergent training scheme based on the resilient propagation algorithm*. Neurocomputing 64, pages 253-270.

plot.nn is a method for the plot generic. It is designed for an inspection of the weights for objects of class nn, typically produced by neuralnet.

### Examples

```r
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)
print(net <- neuralnet(AND+OR~Var1+Var2+Var3, binary.data, hidden=0,
  rep=10, err.fct="ce", linear.output=FALSE))

data(infert, package="datasets")
print(net.infert <- neuralnet(case~parity+induced+spontaneous, infert,
  err.fct="ce", linear.output=FALSE, likelihood=TRUE))
```

### Arguments

- **x**
  - an object of class nn

- **rep**
  - repetition of the neural network. If rep="best", the repetition with the smallest error will be plotted. If not stated all repetitions will be plotted, each in a separate window.
plot.nn

x.entry  x-coordinate of the entry layer. Depends on the arrow.length in default.
x.out    x-coordinate of the output layer.
radius   radius of the neurons.
arrows.length length of the entry and out arrows.
intercept a logical value indicating whether to plot the intercept.
intercept.factor  x-position factor of the intercept. The closer the factor is to 0, the closer the intercept is to its left neuron.
information       a logical value indicating whether to add the error and steps to the plot.
information.pos   y-position of the information.
col.entry.synapse  color of the synapses leading to the input neurons.
col.entry         color of the input neurons.
col.hidden        color of the neurons in the hidden layer.
col.hidden.synapse color of the weighted synapses.
col.out           color of the output neurons.
col.out.synapse   color of the synapses leading away from the output neurons.
col.intercept     color of the intercept.
fontsize          fontsize of the text.
dimension         size of the plot in inches.
show.weights      a logical value indicating whether to print the calculated weights above the synapses.
file               a character string naming the plot to write to. If not stated, the plot will not be saved.
...                arguments to be passed to methods, such as graphical parameters (see par).

Author(s)

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>

See Also

neuralnet

Examples

XOR <- c(0,1,1,0)
xor.data <- data.frame(expand.grid(c(0,1), c(0,1)), XOR)
print(net.xor <- neuralnet( XOR~Var1+Var2, xor.data, hidden=2, rep=5))
plot(net.xor, rep="best")
prediction

Summarizes the output of the neural network, the data and the fitted values of glm objects (if available)

Description

prediction, a method for objects of class nn, typically produced by neuralnet. In a first step, the dataframe will be amended by a mean response, the mean of all responses corresponding to the same covariate-vector. The calculated data.error is the error function between the original response and the new mean response. In a second step, all duplicate rows will be erased to get a quick overview of the data. To obtain an overview of the results of the neural network and the glm objects, the covariate matrix will be bound to the output of the neural network and the fitted values of the glm object(if available) and will be reduced by all duplicate rows.

Usage

prediction(x, list.glm = NULL)

Arguments

x  
neural network

list.glm  
an optional list of glm objects

Value

a list of the summaries of the repetitions of the neural networks, the data and the glm objects (if available).

Author(s)

Stefan Fritsch, Frauke Guenther <guenther@bips.uni-bremen.de>

See Also

neuralnet

Examples

Var1 <- rpois(100,0.5)
Var2 <- rbinom(100,2,0.6)
Var3 <- rbinom(100,1,0.5)
SUM <- as.integer(abs(Var1+Var2+Var3+(rnorm(100))))
sum.data <- data.frame(Var1+Var2+Var3, SUM)
print(net.sum <- neuralnet( SUM~Var1+Var2+Var3, sum.data, hidden=1,
act.fct="tanh")
main <- glm(SUM~Var1+Var2+Var3, sum.data, family=poisson())
full <- glm(SUM~Var1*Var2*Var3, sum.data, family=poisson())
prediction(net.sum, list.glm=list(main=main, full=full))
Index

*Topic neural
  compute, 3
  confidence.interval, 4
  gwplot, 6
  neuralnet, 7
  neuralnet-package, 2
  plot.nn, 10
  prediction, 12
  compute, 3, 3, 10
  confidence.interval, 3, 4, 10
  gwplot, 3, 6, 10
  neuralnet, 5, 7, 7, 11, 12
  neuralnet-package, 2
  par, 6, 11
  plot.default, 6
  plot.nn, 3, 10, 10
  prediction, 3, 10, 12
  print.nn (neuralnet), 7