

Package ‘AMORE’

April 17, 2009

Encoding UTF-8

Version 0.2-11

Date 2007-10-05

Title A MORE flexible neural network package

Author Manuel Castejón Limas, Joaquín B. Ordieres Meré, Eliseo P. Vergara González, Francisco Javier Martínez de Pisón Ascacibar, Alpha V. Pernía Espinoza, Fernando Alba Elías

Maintainer Manuel Castejón Limas <manuel.castejon@unileon.es>

Description This package was born to release the TAO robust neural network algorithm to the R users. It has grown and I think it can be of interest for the users wanting to implement their own training algorithms as well as for those others whose needs lye only in the “user space”.

License GPL (>= 2)

URL <http://wiki.r-project.org/rwiki/doku.php?id=packages:cran:amore>

LazyLoad yes

Repository CRAN

Date/Publication 2009-02-19 13:26:02

R topics documented:

ADAPTgd.MLPnet	2
ADAPTgdwm.MLPnet	3
BATCHgd.MLPnet	4
BATCHgdwm.MLPnet	5
error.TAO	6
error.TAO	7
graphviz.MLPnet	8
init.MLPneuron	9
newff	11
random.init.MLPnet	13

random.init.MLPneuron	14
select.activation.function	15
sim.MLPnet	16
train	17
training.report	18

Index	20
--------------	-----------

ADAPTgd.MLPnet	<i>Adaptative gradient descent training</i>
----------------	---

Description

Adaptative gradient descent training method.

Usage

ADAPTgd.MLPnet (*net*, *P*, *T*, *n.epochs*)

Arguments

<i>net</i>	Neural Network to train.
<i>P</i>	Input data set.
<i>T</i>	Target output data set.
<i>n.epochs</i>	Number of epochs to train

Value

This function returns a neural network object modified according to the input and target data set.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
 Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
 Ana González Marcos. <ana.gonzalez@unileon.es>
 Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
 Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
 Francisco Javier Martinez de Pisón. <francisco.martinez@dim.unirioja.es>
 Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Simon Haykin. Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[newff](#), [newff](#), [newff](#)

ADAPTgdwm.MLPnet *Adaptative gradient descent with momentum training*

Description

Adaptative gradient descent with momentum training method.

Usage

```
ADAPTgdwm.MLPnet (net, P, T, n.epochs)
```

Arguments

net	Neural Network to train.
P	Input data set.
T	Target output data set.
n.epochs	Number of epochs to train

Value

This function returns a neural network object modified according to the input and target data set.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martinez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Simon Haykin. Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[newff](#), [newff](#), [newff](#)

BATCHgd.MLPnet *Batch gradient descent training*

Description

Modifies the neural network weights and biases according to the training set.

Usage

```
BATCHgd.MLPnet (net, P, T, n.epochs)
```

Arguments

<code>net</code>	Neural Network to train.
<code>P</code>	Input data set.
<code>T</code>	Target output data set.
<code>n.epochs</code>	Number of epochs to train

Value

This function returns a neural network object modified according to the chosen data.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martinez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Simon Haykin. Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[newff](#), [newff](#), [newff](#)

BATCHgdm.MLPnet *Batch gradient descent with momentum training*

Description

Modifies the neural network weights and biases according to the training set.

Usage

```
BATCHgdm.MLPnet (net, P, T, n.epochs)
```

Arguments

<code>net</code>	Neural Network to train.
<code>P</code>	Input data set.
<code>T</code>	Target output data set.
<code>n.epochs</code>	Number of epochs to train

Value

This functions returns a neural network object modified according to the chosen data.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Simon Haykin. Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[newff](#), [newff](#), [newff](#)

`error.TAO`*Neural network training error criteria.*

Description

The error functions calculate the goodness of fit of a neural network according to certain criterium:

LMS: Least Mean Squares Error.

LMLS: Least Mean Log Squares minimization.

TAO: TAO error minimization. The deltaE functions calculate the influence functions of their error criteria.

Usage

```
error.LMS(arguments)
error.LMLS(arguments)
error.TAO(arguments)
deltaE.LMS(arguments)
deltaE.LMLS(arguments)
deltaE.TAO(arguments)
```

Arguments

`arguments` List of arguments to pass to the functions.

The first element is the prediction of the neuron.

The second element is the corresponding component of the target vector.

The third element is the whole net. This allows the TAO criterium to know the value of the S parameter and eventually (next minor update) will allow the user to apply regularization criteria.

Value

This functions return the error and influence function criteria.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Pernía Espinoza, A.V., Ordieres Meré, J.B., Martínez de Pisón, F.J., González Marcos, A. TAO-robust backpropagation learning algorithm. *Neural Networks*. Vol. 18, Issue 2, pp. 191–204, 2005.

Simon Haykin. *Neural Networks – a Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[train](#)

error.TAO

TAO robust error criterium auxiliar functions.

Description

Auxiliar functions. Not meant to be called from the user but from the [error.TAO](#) and the [deltaE.TAO](#) functions.

Usage

```
hfun(v, k)
phifun(v, k)
dphifun(v, k)
```

Arguments

v	Input value.
k	Threshold limit.

Value

These functions return a numeric array with dimension equal to the dimension of v.

Author(s)

Manuel Castejón Limas. [⟨manuel.castejon@unileon.es⟩](mailto:manuel.castejon@unileon.es)
Joaquín Ordieres Meré. [⟨joaquin.ordieres@dim.unirioja.es⟩](mailto:joaquin.ordieres@dim.unirioja.es)
Ana González Marcos. [⟨ana.gonzalez@unileon.es⟩](mailto:ana.gonzalez@unileon.es)
Alpha V. Pernía Espinoza. [⟨alpha.pernia@alum.unirioja.es⟩](mailto:alpha.pernia@alum.unirioja.es)
Eliseo P. Vergara Gonzalez. [⟨eliseo.vergara@dim.unirioja.es⟩](mailto:eliseo.vergara@dim.unirioja.es)
Francisco Javier Martínez de Pisón. [⟨francisco.martinez@dim.unirioja.es⟩](mailto:francisco.martinez@dim.unirioja.es)
Fernando Alba Elías. [⟨fernando.alba@unavarra.es⟩](mailto:fernando.alba@unavarra.es)

References

Pernía Espinoza, A.V., Ordieres Meré, J.B., Martínez de Pisón, F.J., González Marcos, A. TAO-robust backpropagation learning algorithm. *Neural Networks*. Vol. 18, Issue 2, pp. 191–204, 2005.

Simon Haykin. *Neural Networks – a Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[train](#)

graphviz.MLPnet *Neural network graphic representation*

Description

Creates a dot file, suitable to be processed with graphviz, containing a graphical representation of the network topology and some numerical information about the network parameters.

Usage

```
graphviz.MLPnet(net, filename, digits)
```

Arguments

net	Neural Network.
filename	Name of the dot file to be written.
digits	Number of digits used to round the parameters.

Value

This function writes a file suitable to be postprocessed with the graphviz package. Thus, multiple formats can be obtained: ps, pdf, ...

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquín Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

<http://www.graphviz.org>

init.MLPneuron *Neuron constructor.*

Description

Creates a neuron according to the structure established by the AMORE package standard.

Usage

```
init.MLPneuron(id, type, activation.function, output.links, output.aims, input.links)
```

Arguments

id	Numerical index of the neuron (so as to be referred in a network operation).
type	Either hidden or output, according to the layer the neuron belongs to.
activation.function	The name of the characteristic function of the neuron. It can be "pureline", "tansig", "sigmoid" or even "custom" in case that the user wants to configure its own activation function accordingly defining f0 and f1.
output.links	The id's of the neurons that accept the output value of this neuron as an input.
output.aims	The location of the output of the neuron in the input set of the addressed neuron. Gives answer to: Is this output the first, the second, the third, ..., input at the addressed neuron?. Similarly for an output neuron: Is this output the first, the second, the third, ..., element of the output vector?
input.links	The id's of the neurons whose outputs work as inputs for this neuron. Positive values represent that we take the outputs of other neurons as inputs. Negative values represent the coordinates of the input vector to be considered as inputs.
weights	The multiplying factors of the input values.
bias	The bias summed to the weighted sum of the inputs.
method	Preferred training method. Currently it can be: <ul style="list-style-type: none"> • "ADAPTgd": Adaptive gradient descend. • "ADAPTgdwm": Adaptive gradient descend with momentum. • "BATCHgd": BATCH gradient descend. • "BATCHgdwm": BATCH gradient descend with momentum.
method.dep.variables	Variables used by the training methods: <ul style="list-style-type: none"> • ADAPTgd method: <ul style="list-style-type: none"> – delta: Used in the backpropagation method.

- learning.rate: Learning rate parameter. Notice that we can use a different rate for each neuron.
- ADAPTgdwm method:
 - delta: Used in the backpropagation method.
 - learning.rate: Learning rate parameter. Notice that we can use a different rate for each neuron.
 - momentum: Momentum constant used in the backpropagation with momentum learning criterium.
 - former.weight.change: Last increment in the weight parameters. Used by the momentum training technique.
 - former.bias.change: Last increment in the bias parameter. Used by the momentum training technique.
- BATCHgd method:
 - delta: Used in the backpropagation method.
 - learning.rate: Learning rate parameter. Notice that we can use a different rate for each neuron.
 - sum.delta.x: Used as an acumulator of the changes to apply to the weight parameters in the batch training.
 - sum.delta.bias: Used as an acumulator of the changes to apply to the bias parameters in the batch training.
- BATCHgdwm method:
 - delta: Used in the backpropagation method.
 - learning.rate: Learning rate parameter. Notice that we can use a different rate for each neuron.
 - sum.delta.x: Used as an acumulator of the changes to apply to the weight parameters in the batch training.
 - sum.delta.bias: Used as an acumulator of the changes to apply to the bias parameters in the batch training.
 - momentum: Momentum constant used in the backpropagation with momentum learning criterium.
 - former.weight.change: Last increment in the weight parameters. Used by the momentum training technique.
 - former.bias.change: Last increment in the bias parameter. Used by the momentum training technique.

Value

init.MLPneuron returns a single neuron. Mainly used to create a neural network object.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
 Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
 Ana González Marcos. <ana.gonzalez@unileon.es>
 Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
 Eliseo P. Vergara González. <eliseo.vergara@dim.unirioja.es>

Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
 Fernando Alba Elías. <fernando.alba@unavarra.es>

See Also

`newff`, `random.init.MLPnet`, `random.init.MLPneuron`, `select.activation.function`, `init.MLPneuron`

`newff`

Create a Multilayer Feedforward Neural Network

Description

Creates a feedforward artificial neural network according to the structure established by the AMORE package standard.

Usage

```
newff(n.neurons, learning.rate.global, momentum.global, error.criterium, Stao, hidden.layer)
```

Arguments

`n.neurons` Numeric vector containing the number of neurons of each layer. The first element of the vector is the number of input neurons, the last is the number of output neurons and the rest are the number of neuron of the different hidden layers.

`learning.rate.global` Learning rate at which every neuron is trained.

`momentum.global` Momentum for every neuron. Needed by several training methods.

`error.criterium` Criterium used to measure to proximity of the neural network prediction to its target. Currently we can choose amongst:

- "LMS": Least Mean Squares.
- "LMLS": Least Mean Logarithm Squared (Liano 1996).
- "TAO": TAO Error (Pernia, 2004).

`Stao` Stao parameter for the TAO error criterium. Unused by the rest of criteria.

`hidden.layer` Activation function of the hidden layer neurons. Available functions are:

- "purelin".
- "tansig".
- "sigmoid".
- "hardlim".
- "custom": The user must manually define the f0 and f1 elements of the neurons.

`output.layer` Activation function of the hidden layer neurons according to the former list shown above.

`method` Preferred training method. Currently it can be:

- "ADAPTgd": Adaptive gradient descend.
- "ADAPTgdwm": Adaptive gradient descend with momentum.
- "BATCHgd": BATCH gradient descend.
- "BATCHgdwm": BATCH gradient descend with momentum.

Value

`newff` returns a multilayer feedforward neural network object.

Author(s)

Manuel Castejón Limas. manuel.castejon@unileon.es
 Joaquin Ordieres Meré. Ana González Marcos. Alpha V. Pernía Espinoza. Eliseo P. Vergara Gonzalez. Francisco Javier Martinez de Pisón. Fernando Alba Elías.

References

Pernía Espinoza, A.V., Ordieres Meré, J.B., Martínez de Pisón, F.J., González Marcos, A. TAO-robust backpropagation learning algorithm. *Neural Networks*. Vol. 18, Issue 2, pp. 191–204, 2005.

Simon Haykin. *Neural Networks – a Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[init.MLPneuron](#), [random.init.MLPnet](#), [random.init.MLPneuron](#), [select.activation.function](#)

Examples

```
#Example 1

library(AMORE)
# P is the input vector
P <- matrix(sample(seq(-1,1,length=1000), 1000, replace=FALSE), ncol=1)
# The network will try to approximate the target P^2
target <- P^2
# We create a feedforward network, with two hidden layers.
# The first hidden layer has three neurons and the second has two neurons.
# The hidden layers have got Tansig activation functions and the output layer is Purelin.
net <- newff(n.neurons=c(1,3,2,1), learning.rate.global=1e-2, momentum.global=0.5,
            error.criterium="LMS", Stao=NA, hidden.layer="tansig",
            output.layer="purelin", method="ADAPTgdwm")
result <- train(net, P, target, error.criterium="LMS", report=TRUE, show.step=100, n.shows=5)
y <- sim(result$net, P)
plot(P,y, col="blue", pch="+")
```

```
points(P,target, col="red", pch="x")
```

random.init.MLPnet *Initialize the network with random weights and biases.*

Description

Provides random values to the network weights and biases so as to start with. Basically it applies the random.init.MLPneuron function to every neuron in the network.

Usage

```
random.init.MLPnet(net)
```

Arguments

net The neural network object

Value

random.init.MLPnet returns the input network with weights and biases changed randomly.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara González. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

See Also

[random.init.MLPneuron](#), [init.MLPneuron](#), [newff](#)

```
random.init.MLPneuron
```

Initialize the neuron with random weights and bias.

Description

Provides random values to the neuron weights and bias so as to start with. It is usually called by the random.init.NeuralNet function during the construction of the neural object by the *newff* function.

Usage

```
random.init.MLPneuron(net.number.weights, neuron)
```

Arguments

<code>net.number.weights</code>	Number of bias and weight parameters of the neural network the neuron belongs to.
<code>neuron</code>	The neuron object.

Details

The values are assigned according to the suggestions of *Haykin*.

Value

random.init.MLPneuron returns the input neuron with bias and weights changed randomly.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquín Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara González. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Simon Haykin. *Neural Networks – a Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

`random.init.MLPnet`, `init.MLPneuron`, `newff`

```
select.activation.function
```

Provides R code of the selected activation function.

Description

Provides random values to the neuron weights and bias so as to start with. It is usually called by the `random.init.NeuralNet` function during the construction of the neural object by the `newff` function.

Usage

```
select.activation.function(activation.function)
```

Arguments

```
activation.function
```

Activation function name. Currently the user may choose amongst *purelin*, *tansig*, *sigmoid*, *hardlim* and *custom*. If *custom* is chosen the the user must manually assign the neuron *f0* and *f1* functions.

Value

`select.activation.function` returns a list with two elements. The first, *f0* is the R code selected to serve as the neuron activation function. The second, *f1* is the R code of the activation function derivative.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara González. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

See Also

`init.MLPneuron`, `newff`

sim.MLPnet

Performs the simulation of a neural network from an input data set.

Description

This function calculates the output values of the neural network for a given data set. Various versions are provided according to different degrees of C code conversion. The *sim.MLPnet* function is the latest and quickest.

Usage

```
sim(net, P, ...)  
sim.MLPnet(net, P, ...)
```

Arguments

...	Currently, the parameters below are accepted.
net	Neural Network to simulate.
P	Data Set input values.

Value

This function returns a matrix containing the output values of the neural network for the given data set.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Simon Haykin. Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[newff](#), [newff](#)

train	<i>Neural network training function.</i>
-------	--

Description

For a given data set (training set), this function modifies the neural network weights and biases to approximate the relationships amongst variables present in the training set. These may serve to satisfy several needs, i.e. fitting non-linear functions.

Usage

```
train(net, P, T, Pval=NULL, Tval=NULL, error.criterium="LMS", report=TRUE, n.shows,
```

Arguments

net	Neural Network to train.
P	Training set input values.
T	Training set output values
Pval	Validation set input values for optional early stopping.
Tval	Validation set output values for optional early stopping.
error.criterium	Criterium used to measure the goodness of fit: "LMS", "LMLS", "TAO".
Stao	Initial value of the S parameter used by the TAO algorithm.
report	Logical value indicating whether the training function should keep quiet or should provide graphical/written information during the training process instead.
n.shows	Number of times to report (if report is TRUE). The total number of training epochs is n.shows times show.step.
show.step	Number of epochs to train non-stop until the training function is allow to report.
prob	Vector with the probabilities of each sample so as to apply resampling training.

Value

This function returns a list with two elements: the trained Neural Network object with weights and biases adjusted by the adaptative backpropagation with momentum method and a matrix with the errors obtained during the training. If the validation set is provided, the early stopping technique is applied.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
 Joaquin Ordieres Meré <joaquin.ordieres@dim.unirioja.es>
 Ana González Marcos. <ana.gonzalez@unileon.es>
 Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>

Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elías. <fernando.alba@unavarra.es>

References

Pernía Espinoza, A.V., Ordieres Meré, J.B., Martínez de Pisón, F.J., González Marcos, A. TAO-robust backpropagation learning algorithm. *Neural Networks*. Vol. 18, Issue 2, pp. 191–204, 2005.

Simon Haykin. *Neural Networks – a Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[newff](#)

`training.report` *Neural network training report generator function.*

Description

Function in charge of reporting the behavior of the network training. The users should modify this function according to their needs.

Usage

```
training.report(net,P,T, idx.show, error.criterium)
```

Arguments

<code>net</code>	Neural Network to train.
<code>P</code>	Training set input values.
<code>T</code>	Training set output values
<code>idx.show</code>	Current show index.
<code>error.criterium</code>	Criterium used to measure the goodness of fit.

Value

This function does not return any value. Just useful for printing and plotting.

Author(s)

Manuel Castejón Limas. <manuel.castejon@unileon.es>
Joaquin Ordieres Meré. <joaquin.ordieres@dim.unirioja.es>
Ana González Marcos. <ana.gonzalez@unileon.es>
Alpha V. Pernía Espinoza. <alpha.pernia@alum.unirioja.es>
Eliseo P. Vergara Gonzalez. <eliseo.vergara@dim.unirioja.es>
Francisco Javier Martínez de Pisón. <francisco.martinez@dim.unirioja.es>
Fernando Alba Elfas. <fernando.alba@unavarra.es>

References

Simon Haykin. Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999. ISBN 0-13-273350-1.

See Also

[train](#)

Index

*Topic **neural**

- ADAPTgd.MLPnet, 2
 - ADAPTgdwm.MLPnet, 3
 - BATCHgd.MLPnet, 4
 - BATCHgdwm.MLPnet, 5
 - error.TAO, 6, 7
 - graphviz.MLPnet, 8
 - init.MLPneuron, 9
 - newff, 11
 - random.init.MLPnet, 13
 - random.init.MLPneuron, 14
 - select.activation.function, 15
 - sim.MLPnet, 16
 - train, 17
 - training.report, 18
-
- ADAPTgd.MLPnet, 2
 - ADAPTgdwm.MLPnet, 3
-
- BATCHgd.MLPnet, 4
 - BATCHgdwm.MLPnet, 5
-
- deltaE.LMLS (error.TAO), 6
 - deltaE.LMS (error.TAO), 6
 - deltaE.TAO, 7
 - deltaE.TAO (error.TAO), 6
 - dphifun (error.TAO), 7
-
- error.LMLS (error.TAO), 6
 - error.LMS (error.TAO), 6
 - error.TAO, 6, 7, 7
-
- graphviz.MLPnet, 8
-
- hfun (error.TAO), 7
-
- init.MLPneuron, 9, 11–13, 15
-
- newff, 2–5, 11, 11, 13, 15, 16, 18
-
- phifun (error.TAO), 7
-
- random.init.MLPnet, 11, 12, 13, 15
 - random.init.MLPneuron, 11–13, 14
-
- select.activation.function, 11, 12, 15
-
- sim (sim.MLPnet), 16
 - sim.MLPnet, 16
-
- train, 7, 8, 17, 19
 - training.report, 18