Package 'darch'

February 19, 2015

Type Package

Title Package for deep architectures and Restricted-Bolzmann-Machines

Version 0.9.1

Date 2013-04-17

Author Martin Drees

Maintainer Martin Drees <mdrees@stud.fh-dortmund.de>

Description The darch package is build on the basis of the code from G. E. Hinton and R. R. Salakhutdinov (available under Matlab Code for deep belief nets: last visit: 01.08.2013). This package is for generating neural networks with many layers (deep architectures) and train them with the method introduced by the publications `A fast learning algorithm for deep belief nets" (G. E. Hinton, S. Osindero, Y. W. Teh) and `Reducing the dimensionality of data with neural networks" (G. E. Hinton, R. R. Salakhutdinov). This method includes a pre training with the contrastive divergence method publishing by G.E Hinton (2002) and a fine tuning with common known training algorithms like backpropagation or conjugate gradient.

License GPL-2

URL http://github.com/maddin79/darch

Imports futile.logger, ff, methods

Collate 'net.R' 'darch.R' 'backpropagation.R' 'darch.Add.R'
'darch.Getter.R' 'darch.Learn.R' 'darch.Remove.R'
'darch.Reset.R' 'darch.Setter.R' 'darchUnitFunctions.R'
'errorFunctions.R' 'rbm.R' 'generateRBMs.R' 'generateWeights.R'
'loadDArch.R' 'loadRBM.R' 'loadRBMFFWeights.R'
'makeStartEndPoints.R' 'minimize.R' 'minimizeAutoencoder.R'
'minimizeClassifier.R' 'net.Getter.R' 'net.Setter.R'
'newDArch.R' 'rbm.Setter.R' 'rbm.Reset.R' 'newRBM.R'
'package-darch.R' 'rbm.Getter.R' 'rbm.Learn.R'
'rbmUnitFunctions.R' 'rbmUpdate.R' 'readMNIST.R'
'rpropagation.R' 'runDArch.R' 'saveDArch.R' 'saveRBM.F'
'saveRBMFFWeights.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-16 13:34:52

R topics documented:

addExecOutput
addLayer
addLayerField
backpropagation
binSigmoidUnit
crossEntropyError
DArch
darch 9
fineTuneDArch
generateRBMs
generateWeights
getBatchSize
getCancel
getCancelMessage
getErrorFunction
getExecOutput
getExecOutputs
getExecuteFunction
getFF
getFinalMomentum
getFineTuneFunction
getGenWeightFunction
getHiddenBiases
getHiddenBiasesInc
getHiddenUnitStates
getLayer
getLayerField
getLayerFunction
getLayers
getLayerWeights
getLearnRateBiases
getLearnRateBiasHidden
getLearnRateBiasVisible
getLearnRateWeights
getMomentum
getMomentumSwitch
getNumHidden
getNumVisible
getOutput
getPosPhaseData
getRBMList
getStats

getVisibleBiases		30
getVisibleBiasesInc		30
getVisibleUnitStates		31
getWeightCost		32
getWeightInc		32
getWeights		33
linearUnit		34
linear Unit Derivative		34
linearUnitFunc		35
loadDArch		36
loadRBM		36
oadRBMFFWeights		37
makeStartEndPoints		37
minimize		38
minimizeAutoencoder		39
minimizeClassifier	 	40
mseError		41
Net	 	41
newDArch	 	42
newRBM	 	43
preTrainDArch	 	43
quadraticError		44
RBM		45
rbmUpdate	 	46
readMNIST	 	46
removeLayerField	 	47
resetDArch	 	47
resetExecOutput		48
resetRBM		48
rpropagation		49
runDArch		50
saveDArch		51
saveRBM		52
saveRBMFFWeights		52
setBatchSize<		53
setCancel<		53
setCancelMessage<		54
setErrorFunction<-		54
setExecuteFunction<		55
setFF<		55
setFinalMomentum<		56
setFineTuneFunction<		56
setGenWeightFunction<		57
setHiddenBiases<-		57
setHiddenBiasesInc<		58
setHiddenUnitFunction<		58
setHiddenUnitStates<		59
setLayer<		59
· · · · · · · · · · · · · · · · · · ·	 •	-/

4 addExecOutput

	setLayerField<	60
	setLayerFunction<	60
	setLayers<	61
	setLayerWeights<	61
	setLearnRateBiases<	
	setLearnRateBiasHidden<	62
	setLearnRateBiasVisible<	63
	setLearnRateWeights<	63
	setLogLevel<	63
	setMomentum<	64
	setMomentumSwitch<	65
	setNumHidden<	65
	setNumVisible<	66
	setOutput<	66
	setPosPhaseData<	67
	setRBMList<	67
	setStats<	68
	setUpdateFunction<	68
	setVisibleBiases<	69
	setVisibleBiasesInc<	69
	setVisibleUnitFunction<	70
	setVisibleUnitStates<-	70
	setWeightCost<-	71
	setWeightInc<-	71
	setWeights<	72
	sigmoidUnit	72
	sigmoidUnitDerivative	73
	sigmUnitFunc	73
	sigmUnitFuncSwitch	74
	softmaxUnit	75
	softmaxUnitDerivative	75
	trainRBM	76
	ualikDM	70
Index		77
addEx	xecOutput Adds an execution output for a DArch object	

Description

This method can be used to save the execution outputs of every layer for the DArch object. The outputs are saved in a list and every time this function is called, the list is extended of one field with the new output.

addLayer 5

Usage

```
addExecOutput(darch, output)
## S4 method for signature 'DArch'
addExecOutput(darch, output)
```

Arguments

darch An instance of the class DArch.

output The output of the layer.

See Also

DArch

addLayer

Adds a layer to the DArch object

Description

Adds a layer to the given DArch object. The parameter weights and biases will be put together in one matrix.

Usage

```
addLayer(darch, weights, biases, unitFunction)
## S4 method for signature 'DArch'
addLayer(darch, weights, biases, unitFunction)
```

Arguments

darch An instance of the class DArch.

weights The weights for the layer.

biases The biases for the layer.

unitFunction The functions of the units in the layer.

See Also

 $\label{lem:decomposition} DArch, sigmoidUnit, bin SigmoidUnit, sigmoidUnit Derivative, linear Unit, linear Unit Derivative, softmax Unit, softmax Unit Derivative, linear Unit, linear Unit Derivative, softmax Unit, softmax Unit, softmax Unit, softmax Unit Derivative, linear Unit, linear Un$

6 backpropagation

addLayerField

Adds a field to a layer

Description

Adds a field to the layer given by the index.

Usage

```
addLayerField(darch, index, field)
## S4 method for signature 'DArch'
addLayerField(darch, index, field)
```

Arguments

darch An instance of the class DArch.

index The position of the layer.

field The new field for the layer.

See Also

DArch

backpropagation

Backpropagation learning function

Description

This function provides the backpropagtion algorithm for deep architectures.

Usage

backpropagation(darch, trainData, targetData, epoch)

Arguments

darch An instance of the class DArch.

trainData The data for training.
targetData The targets for the data

epoch Number of epochs for the training

binSigmoidUnit 7

Details

The function is getting the learning parameters from the provided DArch object. It uses the attributes momentum, finalMomentum and momentumSwitch for the calculation of the new weights with momentum. The parameter epoch is provided for the change from momentum to finalMomentum and is compared to momentumSwitch. The attributes learnRateWeights and learnRateBiases will be used for updating the weights. To use the backpropagation function as the fine tuning function the layer functions of the darch DArch object must set to the versions which calculates also the derivatives of the function result.

Value

The trained deep architecture

References

Rumelhart, D., G. E. Hinton, R. J. Williams, Learning representations by backpropagating errors, Nature 323, S. 533-536, DOI: 10.1038/323533a0, 1986.

See Also

DArch rpropagation minimizeAutoencoder minimizeClassifier minimizeClassifier

binSigmoidUnit

Binary sigmoid unit function.

Description

The function calculates the activation and the output from the sigmoid transfer function. It returns a binary matrix where a entry is 1 if the value is bigger than a random number generated with runif.

Usage

binSigmoidUnit(data,weights)

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

Value

A list with the binary activation of the unit in the first entry.

See Also

DArch, sigmoidUnit, sigmoidUnitDerivative, linearUnit, linearUnitDerivative, softmaxUnit, softmaxUnitDerivative

8 DArch

crossEntropyError

Cross entropy error function

Description

The function calculates the cross entropy error from the original and estimate parameters.

Usage

```
crossEntropyError(original, estimate)
```

Arguments

original The original data matrix
estimate The calculated data matrix

Value

A list with the name of the error function in the first entry and the error value in the second entry

See Also

```
quadraticError, mseError
```

DArch

Class for deep architectures

Description

This class implements deep architectures and provides the ability to train them with a pre training using contrastive divergence and fine tuning with backpropagation, resilient backpropagation and conjugate gradients.

Details

The class is inherits all attributes from the class link{Net}. When creating a new instance with the constructor newDArch (recommended), the darch-object contained the number of layers -1 restricted bolzmann machines (RBM), which are used for the unsupervised pre training of the network. The RBMs are saved in the attribute rbmList and can be fetched over the getter method (getRBMList. The two attributes fineTuneFunction and executeFunction containing the functions for the fine tuning (default: backpropagation) and for the execution (default: runDArch. The training of the network is performed by the two learning functions preTrainDArch and fineTuneDArch. The first function trains the network with the unsupervised method contrastive divergence. The second function used the function in the attribute fineTuneFunction for the fine tuning. After an execution of the network, the outputs of every layer can be found in the attribute executeOutput.

darch 9

Slots

rbmList: A list which contains all rbm's for the pre-training.

layers: A list with the layer information. In the first field are the weights and in the second field is the unit function.

learnRateBiases: The learning rate for the bias weights.

fineTuneFunction: Contains the function for the fine tuning.

executeFunction: Contains the function for executing the network.

executeOutput: A list which contains the outputs of every layer after an execution of the network.

cancel: Boolean value which indicates if the network training is canceled.

executeOutput: A string containing the message why the network training is stopped.

cancel: Indicates if the execution is canceled.

cancelMessage: The message when the execution is canceled.

Author(s)

Martin Drees

See Also

Net, RBM

darch

Deep architectures in R

Description

The darch-package implements Deep-Architecture-Networks and Restricted-Bolzmann-Machines.

Details

The creation of this package is motivated by the papers from G. Hinton et. al. from 2006 (see references for details) and from the matlab source code developed in this context. This package provides the possibility to generate deep architecture networks (darch) like the deep belief networks from Hinton et. al.. The deep architectures can then be trained with the contrastive divergence method. After this pre-training it can be fine tuned with several lerning methods like backpropagation, resilient backpropagtion and conjugate gradients.

Package: darch
Type: Package
Version: 0.9.1
Date: 2013-04-17
License: GPL-2
LazyLoad: yes

10 darch

Author(s)

Martin Drees <mdrees@stud.fh-dortmund.de>

Maintainer: Martin Drees <mdrees@stud.fh-dortmund.de>

References

Hinton, G. E., S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18(7), S. 1527-1554, DOI: 10.1162/neco.2006.18.7.1527 2006.

Hinton, G. E., R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313(5786), S. 504-507, DOI: 10.1126/science.1127647, 2006.

Examples

```
## Not run:
# Generating the datasets
inputs <- matrix(c(0,0,0,1,1,0,1,1),ncol=2,byrow=TRUE)
outputs <- matrix(c(0,1,1,0),nrow=4)
# Generating the darch
darch <- newDArch(c(2,4,1),batchSize=2)</pre>
# Pre-Train the darch
darch <- preTrainDArch(darch,inputs,maxEpoch=1000)</pre>
# Prepare the layers for backpropagation training for
# backpropagation training the layer functions must be
# set to the unit functions which calculates the also
# derivatives of the function result.
layers <- getLayers(darch)</pre>
for(i in length(layers):1){
 layers[[i]][[2]] <- sigmoidUnitDerivative</pre>
setLayers(darch) <- layers</pre>
rm(layers)
# Setting and running the Fine-Tune function
setFineTuneFunction(darch) <- backpropagation</pre>
darch <- fineTuneDArch(darch,inputs,outputs,maxEpoch=1000)</pre>
# Running the darch
darch <- darch <- getExecuteFunction(darch)(darch,inputs)</pre>
outputs <- getExecOutputs(darch)</pre>
cat(outputs[[length(outputs)]])
## End(Not run)
```

fineTuneDArch 11

The tuning function for the deep architecture.	fineTuneDArch	Fine tuning function for the deep architecture.	
--	---------------	---	--

Description

The fine tuning function for deep architectures. This function use the function saved in the attribute fineTuneFunction to train the deep architecture.

Usage

```
fineTuneDArch(darch, trainData, targetData, ..., maxEpoch = 1,
  isBin = FALSE, isClass = TRUE, validData = NULL, validTargets = NULL,
  testData = NULL, testTargets = NULL, stopErr = -Inf,
  stopClassErr = 101, stopValidErr = -Inf, stopValidClassErr = 101)
## S4 method for signature 'DArch'
fineTuneDArch(darch, trainData, targetData, ...,
 maxEpoch = 1, isBin = FALSE, isClass = TRUE, validData = NULL,
  validTargets = NULL, testData = NULL, testTargets = NULL,
  stopErr = -Inf, stopClassErr = 101, stopValidErr = -Inf,
  stopValidClassErr = 101)
```

Arguments

darch	A instance of the class DArch.
trainData	The training data matrix
targetData	The expected output matrix for the training data
	Additional parameters for the training function
maxEpoch	The number of training iterations
isBin	Indicates whether the output data must be interpreted as boolean value. Default is FALSE. If it is true, every value over 0.5 is interpreted as 1 and under as 0.
isClass	Indicates whether the training is for a classification net. When TRUE then statistics for classification will be determind. Default is TRUE
validData	Data for validating the network. Default is NULL
validTargets	The expected output for the training data Default is NULL
testData	Data for testing the network. Default is NULL
testTargets	The expected output for the training data Default is NULL
stopErr	Stop criteria for the error on the train data. Default is -Inf
stopClassErr	Stop criteria for the classification error on the train data. Default is 101
stopValidErr	Stop criteria for the error on the validation data. Default is -Inf.
stopValidClassE	irr
	Stop criteria for the classification error on the validation data. Default is 101.

Stop criteria for the classification error on the validation data. Default is 101.

12 generateRBMs

Details

The function trains the given network darch with the function saved in the attribute fineTuneFunction of the DArch-Object. The data (trainData, validData, testData) and belonging classes of the data (targetData, validTargets, testTargets) can be hand over either as matrix or as ff-matrix (see package ff for details). The data and classes for validation and testing are optional. If they are provided the network will be executed with this datasets and statistics will be calculated. This statistics are saved in the stats attribute (see Net). The attribue isBin indicates whether the output data must be interpreted as binary value. If true every value over 0.5 is interpreted as 1 otherwise as 0. Also it is possible to set stop criteria for the training on the error (stopErr, stopValidErr) or the correct classifications (stopClassErr, stopValidClassErr) of the training or validation dataset.

See Also

DArch, Net, backpropagation, rpropagation, minimizeAutoencoder, minimizeClassifier

generateRBMs

Generates the rbm's for the pre-training.

Description

Used the layer sizes from the DArch object to create the RBM objects for the pre-training.

Usage

```
generateRBMs(darch, layers, genWeightFunc = generateWeights)
## S4 method for signature 'DArch'
generateRBMs(darch, layers, genWeightFunc = generateWeights)
```

Arguments

darch A instance of the class DArch.

layers An array with the sizes of the layers

genWeightFunc The function for generating the weight matrices

Value

The DArch object with the generated rbm's

See Also

DArch RBM

generateWeights 13

generateWeights

Generates a weight matrix.

Description

This function is the standard method for generating weights for instances of Net. When using another function to generate weights, the function must be like this one.

Usage

```
generateWeights(numUnits1, numUnits2)
```

Arguments

numUnits1 Number of units in the lower layer. numUnits2 Number of units in the upper layer.

See Also

Net

getBatchSize

Returns the batch size of the Net.

Description

Returns the batch size of the Net.

Usage

```
getBatchSize(net)
## S4 method for signature 'Net'
getBatchSize(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

14 getCancelMessage

getCancel

Returns the cancel value.

Description

Returns the cancel value.

Usage

```
getCancel(darch)
## S4 method for signature 'DArch'
getCancel(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

DArch

getCancelMessage

Returns the cancel message.

Description

Returns the message, why the learing is canceled. If the message is not set, the default value "no reason specified" will be returned.

Usage

```
getCancelMessage(darch)

## S4 method for signature 'DArch'
getCancelMessage(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

getErrorFunction 15

 ${\tt getErrorFunction}$

Returns the error function of the Net.

Description

Returns the error function of the Net.

Usage

```
getErrorFunction(net)
## S4 method for signature 'Net'
getErrorFunction(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

getExecOutput

Returns the execution output of the layer from the DArch object

Description

Returns the execution output of the layer by the given index. If the index is not set, the output of the last layer will be returned.

Usage

```
getExecOutput(darch,index=1)
## S4 method for signature 'DArch'
getExecOutput(darch, index = NULL)
```

Arguments

darch A instance of the class DArch.

index The index of the layer.

See Also

16 getExecuteFunction

getExecOutputs

Returns the execution output list of the DArch object

Description

Returns the execution output of the DArch object. The list contains all outputs of every layer in the network.

Usage

```
getExecOutputs(darch)
## S4 method for signature 'DArch'
getExecOutputs(darch)
```

Arguments

darch

A instance of the class DArch

See Also

DArch

getExecuteFunction

Returns the function for the execution of the DArch network.

Description

Returns the function for the execution of the DArch network.

Usage

```
getExecuteFunction(darch)

## S4 method for signature 'DArch'
getExecuteFunction(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

getFF 17

getFF

Returns if the weights are saved as ff objects

Description

Returns if the weights are saved as ff objects Returns the ff state of the Net.

Usage

```
getFF(net)
## S4 method for signature 'Net'
getFF(net)
getFF(net)
## S4 method for signature 'Net'
getFF(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

Net

 ${\tt getFinalMomentum}$

Returns the final momentum of the Net.

Description

Returns the final momentum of the Net.

Usage

```
getFinalMomentum(net)
## S4 method for signature 'Net'
getFinalMomentum(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

getFineTuneFunction

Returns the fine tune function for the DArch object.

Description

Returns the fine tune function which is executed by the function fineTuneDArch.

Usage

```
getFineTuneFunction(darch)
## S4 method for signature 'DArch'
getFineTuneFunction(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

DArch, backpropagation, rpropagation, minimizeAutoencoder, minimizeClassifier

getGenWeightFunction Returns the function for generating weight matrices.

Description

Returns the function for generating weight matrices.

Usage

```
getGenWeightFunction(net)
## S4 method for signature 'Net'
getGenWeightFunction(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

getHiddenBiases 19

getHiddenBiases

Returns the biases of the hidden units.

Description

Returns the biases of the hidden units.

Usage

```
getHiddenBiases(rbm)
## S4 method for signature 'RBM'
getHiddenBiases(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The biases of the hidden units.

See Also

RBM

RBM

getHiddenBiasesInc

Returns the update value for the biases of the hidden units.

Description

Returns the update value for the biases of the hidden units.

Usage

```
getHiddenBiasesInc(rbm)

## S4 method for signature 'RBM'
getHiddenBiasesInc(rbm)
```

Arguments

rbm

A instance of the class RBM.

20 getHiddenUnitStates

Value

The update value for the biases of the hidden units.

See Also

RBM

RBM

getHiddenUnitStates

Returns a list with the states of the hidden units.

Description

Returns a list with the states of the hidden units.

Usage

```
getHiddenUnitStates(rbm)
## S4 method for signature 'RBM'
getHiddenUnitStates(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The states of the hidden units.

See Also

RBM

getLayer 21

getLayer

Returns a layer from the DArch object.

Description

The function returns the layer with the given index which contains all weights, functions for the neurons, and possible additional parameters for the training.

Usage

```
getLayer(darch,index=1)
## S4 method for signature 'DArch'
getLayer(darch, index = 1)
```

Arguments

darch A instance of the class DArch.
index The index of the layer.Default is 1.

See Also

DArch

getLayerField

Returns the field of a layer from the DArch object.

Description

The function returns the field given by the fieldIndex of the layer given by the layerIndex.

Usage

```
getLayerField(darch,layerIndex=1,fieldIndex=3)
## S4 method for signature 'DArch'
getLayerField(darch, layerIndex = 1, fieldIndex = 3)
```

Arguments

darch A instance of the class DArch.

layerIndex The index of the layer.

fieldIndex The index of the field in the layer.

See Also

22 getLayers

getLayerFunction

Returns the neuron function of a layer from the DArch object.

Description

The function returns the neuron function of the layer with the given index.

Usage

```
getLayerFunction(darch,index=1)
## S4 method for signature 'DArch'
getLayerFunction(darch, index = 1)
```

Arguments

darch A instance of the class DArch.

index The index of the layer.Default is 1.

See Also

 $\label{lem:decomposition} DArch\,sigmoidUnit,\,binSigmoidUnit,\,sigmoidUnitDerivative,\,linearUnitDerivative,\,softmaxUnit,\,softmaxUnitDerivative$

getLayers

Returns the a list of layers from the DArch object.

Description

The function returns the layers list which contains all weights, functions for the neurons, and possible additional parameters for the training.

Usage

```
getLayers(darch)
## S4 method for signature 'DArch'
getLayers(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

getLayerWeights 23

getLayerWeights

Returns the weights of a layer from the DArch object.

Description

The function returns the weights of the layer with the given index.

Usage

```
getLayerWeights(darch,index=1)
## S4 method for signature 'DArch'
getLayerWeights(darch, index = 1)
```

Arguments

darch

A instance of the class DArch.

index

The index of the layer. Default is 1.

See Also

DArch

getLearnRateBiases

Returns the learning rate for the bias weigths of the DArch object.

Description

Returns the learning rate for the bias weigths of the DArch object.

Usage

```
getLearnRateBiases(darch)
## S4 method for signature 'DArch'
getLearnRateBiases(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

 ${\tt getLearnRateBiasHidden}$

Returns the learning rate for the hidden biases.

Description

Returns the learning rate for the hidden biases.

Usage

```
getLearnRateBiasHidden(rbm)
## S4 method for signature 'RBM'
getLearnRateBiasHidden(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The learning rate for the hidden biases

See Also

RBM

RBM

getLearnRateBiasVisible

Returns the learning rate for the visible biases.

Description

Returns the learning rate for the visible biases.

Usage

```
getLearnRateBiasVisible(rbm)

## S4 method for signature 'RBM'
getLearnRateBiasVisible(rbm)
```

Arguments

rbm

A instance of the class RBM.

getLearnRateWeights 25

Value

The learning rate for the visible biases

See Also

RBM

RBM

 ${\tt getLearnRateWeights}$

Returns the learn rate of the weights.

Description

Returns the learn rate of the weights.

Usage

```
getLearnRateWeights(net)
## S4 method for signature 'Net'
getLearnRateWeights(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

 ${\tt getMomentum}$

Returns the momentum of the Net.

Description

Returns the momentum of the Net.

Usage

```
getMomentum(net)
## S4 method for signature 'Net'
getMomentum(net)
```

26 getNumHidden

Arguments

net

A instance of the class Net.

See Also

Net

 ${\tt getMomentumSwitch}$

Returns the momentum switch of the Net.

Description

Returns the momentum switch of the Net.

Usage

```
getMomentumSwitch(net)
## S4 method for signature 'Net'
getMomentumSwitch(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

getNumHidden

Returns the number of hidden units of the RBM

Description

Returns the number of hidden units of the RBM

Usage

```
getNumHidden(rbm)
## S4 method for signature 'RBM'
getNumHidden(rbm)
```

Arguments

rbm

A instance of the class RBM.

getNumVisible 27

Value

The number of hidden units of the RBM

See Also

RBM

RBM

 ${\tt getNumVisible}$

Returns the number of visible units of the RBM

Description

Returns the number of visible units of the RBM

Usage

```
getNumVisible(rbm)
## S4 method for signature 'RBM'
getNumVisible(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The number of visible units of the RBM

See Also

RBM

RBM

28 getPosPhaseData

getOutput

Returns the output of the RBM

Description

Returns the output of the RBM

Usage

```
getOutput(rbm)
## S4 method for signature 'RBM'
getOutput(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The output of the RBM

See Also

RBM

getPosPhaseData

Returns the data for the positive phaes.

Description

Returns the data for the positive phaes.

Usage

```
getPosPhaseData(rbm)
## S4 method for signature 'RBM'
getPosPhaseData(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The data for the positive phaes.

getRBMList 29

See Also

RBM

RBM

getRBMList

Returns a list of RBMs of the DArch object.

Description

This function returns a list of RBMs of the DArch which are needed for the pre training of the network.

Usage

```
getRBMList(darch)
## S4 method for signature 'DArch'
getRBMList(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

DArch

getStats

Returns the list of statistics for the network

Description

The list of statistics can contain values about errors, miss classifications and other usefull things from the pre-training or fine-tuning of a deep architecture.

Usage

```
getStats(net)
## S4 method for signature 'Net'
getStats(net)
```

Arguments

net

A instance of the class Net.

See Also

Net

30 getVisibleBiasesInc

getVisibleBiases

Returns the biases of the visible units.

Description

Returns the biases of the visible units.

Usage

```
getVisibleBiases(rbm)
## S4 method for signature 'RBM'
getVisibleBiases(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The biases of the visible units.

See Also

RBM

RBM

getVisibleBiasesInc

Returns the update value for the biases of the visible units.

Description

Returns the update value for the biases of the visible units.

Usage

```
getVisibleBiasesInc(rbm)

## S4 method for signature 'RBM'
getVisibleBiasesInc(rbm)
```

Arguments

rbm

A instance of the class RBM.

getVisibleUnitStates 31

Value

The update value for the biases of the visible units.

See Also

RBM

RBM

getVisibleUnitStates Returns a list with the states of the visible units.

Description

Returns a list with the states of the visible units.

Usage

```
getVisibleUnitStates(rbm)
## S4 method for signature 'RBM'
getVisibleUnitStates(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The states of the visible units.

See Also

RBM

32 getWeightInc

getWeightCost

Returns the weigth cost for the training

Description

Returns the weigth cost for the training

Usage

```
getWeightCost(rbm)
## S4 method for signature 'RBM'
getWeightCost(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The weigth cost for the training

See Also

RBM

RBM

getWeightInc

Returns the update value for the weights.

Description

Returns the update value for the weights.

Usage

```
getWeightInc(rbm)
## S4 method for signature 'RBM'
getWeightInc(rbm)
```

Arguments

rbm

A instance of the class RBM.

getWeights 33

Value

The update value for the weights.

See Also

RBM

RBM

getWeights

Returns the weigths of the RBM.

Description

Returns the weigths of the RBM.

Usage

```
getWeights(rbm)
## S4 method for signature 'RBM'
getWeights(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The weigths of the RBM.

See Also

RBM

RBM

34 linearUnitDerivative

linearUnit Linear unit function.

Description

The function calculates the activation of the units and returns it.

Usage

linearUnit(data, weights)

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

Value

A list with the linear activation of the unit in the first entry.

See Also

DArch, sigmoidUnit, binSigmoidUnit, sigmoidUnitDerivative, linearUnitDerivative, softmaxUnit, softmaxUnitDerivative

linearUnitDerivative Linear unit function with unit derivatives.

Description

The function calculates the activation of the units and returns a list, in which the first entry is the linear activation of the units and the second entry is the derivative of the transfer function.

Usage

linearUnitDerivative(data,weights)

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

Value

A list with the linear activation in the first entry and the derivative of the activation in the second entry

linearUnitFunc 35

See Also

DArch, sigmoidUnit, binSigmoidUnit, sigmoidUnitDerivative, linearUnit, softmaxUnit, softmaxUnitDerivative

linearUnitFunc Calculates the linear neuron output no transfer function

Description

Calculates the linear neuron output with no transfer function from real value input saved in the first entry of the list dataList.

Usage

linearUnitFunc(rbm, dataList, biases, weights, runParams)

Arguments

rbm A instance of the class RBM.

dataList A list with the data matrices for the calculations.

biases The biases for the calculations

weights The weight matrix for the calculations

runParams Parameters which indicates the status of the training.

Details

The return value is a list with the output of the neurons as first entry and binary representation calculated through a comparison of the output with random numbers. The random numbers a generated with the function rnorm.

Value

The real value and binary activations for the units

See Also

36 loadRBM

loadDArch

Loads a DArch network

Description

Loads the DArch object from the filename given through the parameter name plus the ending ".net".

Usage

```
loadDArch(name="darch")
```

Arguments

name

The name of the file without the ending ".net".

Details

Make sure when you load a DArch object that every file written by the saveDArch-Funktion, specially when the parameter ff of the saved DArch object is TRUE, are in the working directory.

Value

darch - The loaded deep architecture

See Also

saveDArch, loadRBMFFWeights

loadRBM

Loads a RBM network

Description

Loads the RBM object from the filename given through the parameter name plus the ending ".net".

Usage

```
loadRBM(name="rbm")
```

Arguments

name

The name of the file without the ending ".net".

Details

Make sure when you load a RBM object that every file written by the saveRBM-Funktion, specially when the parameter ff of the saved RBM object is TRUE, are in the working directory

loadRBMFFWeights 37

Value

rbm - The loaded RBM

See Also

saveRBM, loadRBMFFWeights, saveRBMFFWeights

loadRBMFFWeights

Loads weigths and biases for a RBM network from a ffData file.

Description

Loads the weigths and the biases for the given RBM object from the filename given through the parameter name. See ffload for more details

Usage

loadRBMFFWeights(rbm,name)

Arguments

rbm A instance of the class RBM.

name The name of the file without the ending ".net".

Value

rbm - The RBM with the loaded weights and biases.

See Also

ffload, saveRBM, loadRBM, saveRBMFFWeights

 ${\tt makeStartEndPoints}$

Makes start- and end-points for the batches.

Description

The start- and end-points are used for dividing the data into batches.

Usage

```
makeStartEndPoints(batchSize, numRows)
```

38 minimize

Arguments

batchSize Desired batch size

numRows Number of rows of the data

Details

If the data is not divedable by the batchSize the last batch will contain the rest of the data. The function returns a list with in which the first entry is a list with the values for the start and end points for reading the data matrix. The second entry is the number of batches.

See Also

Net

minimize

Minimize a differentiable multivariate function.

Description

This function is a direct translation from the Matlab source code of the minimize function from Carl Edward Rasmussen.

Usage

```
minimize( X, f, length, ...)
```

Arguments

X Starting point. An Array of the weights.

f Function for calculating the function value and the partial derivatives

length Maximum number of line searches or maximum allowed number of function

evaluations if negative

... Additional Parameters for the function f

Details

Minimize a differentiable multivariate function.

```
Usage: [X, fX, i] \leftarrow minimize(X, f, length, P1, P2, P3, ...)
```

where the starting point is given by "X" (D by 1), and the function named in the string "f", must return a function value and a vector of partial derivatives of f wrt X, the "length" gives the length of the run: if it is positive, it gives the maximum number of line searches, if negative its absolute gives the maximum allowed number of function evaluations. You can (optionally) give "length" a second component, which will indicate the reduction in function value to be expected in the first line-search (defaults to 1.0). The parameters P1, P2, P3, ... are passed on to the function f.

minimizeAutoencoder 39

The function returns when either its length is up, or if no further progress can be made (ie, we are at a (local) minimum, or so close that due to numerical problems, we cannot get any closer). NOTE: If the function terminates within a few iterations, it could be an indication that the function values and derivatives are not consistent (ie, there may be a bug in the implementation of your "f" function). The function returns the found solution "X", a vector of function values "fX" indicating the progress made and "i" the number of iterations (line searches or function evaluations, depending on the sign of "length") used. The Polack-Ribiere flavour of conjugate gradients is used to compute search directions, and a line search using quadratic and cubic polynomial approximations and the Wolfe-Powell stopping criteria is used together with the slope ratio method for guessing initial step sizes. Additionally a bunch of checks are made to make sure that exploration is taking place and that extrapolation will not be unboundedly large. See also: checkgrad Copyright (C) 2001 - 2006 by Carl Edward Rasmussen (2006-09-08).

Value

The function returns the found solution "X", a vector of function values "fX" indicating the progress made and "i" the number of iterations (line searches or function evaluations, depending on the sign of "length") used.

See Also

DArch, minimizeAutoencoder, minimizeClassifier

minimizeAutoencoder

Conjugate gradient for a autoencoder network

Description

This function trains a DArch autoencoder network with the conjugate gradient method.

Usage

minimizeAutoencoder(darch, trainData, targetData, epoch, length)

Arguments

darch A instance of the class DArch.

trainData The training data matrix

targetData The labels for the training data epoch The actual epoch of the training

length Numbers of line search

40 minimizeClassifier

Details

This function is build on the basis of the code from G. Hinton et. al. (http://www.cs.toronto.edu/~hinton/MatlabForSciencePaj-last visit 06.06.2013) for the fine tuning of deep belief nets. The original code is located in the files 'backpropclassify.m', 'CG_MNIST.m' and 'CG_CLASSIFY_INIT.m'. It implements the fine tuning for a classification net with backpropagation using a direct translation of the minimize function from C. Rassmussen (available at http://www.gatsby.ucl.ac.uk/~edward/code/minimize/ - last visit 06.06.2013) to R.

Value

The trained DArch object.

See Also

DArch fineTuneDArch

minimizeClassifier Conjugate gradient for a classification network

Description

This function trains a DArch classifier network with the conjugate gradient method.

Usage

minimizeClassifier(darch,trainData,targetData,epoch,length,switchLayers)

Arguments

darch A instance of the class DArch.
trainData The training data matrix

targetData The labels for the training data epoch The actual epoch of the training

length Numbers of line search

switchLayers Indicates when to train the full network instead of only the upper two layers

Details

This function is build on the basis of the code from G. Hinton et. al. (http://www.cs.toronto.edu/~hinton/MatlabForSciencePajelast visit 06.06.2013) for the fine tuning of deep belief nets. The original code is located in the files 'backpropclassify.m', 'CG_MNIST.m' and 'CG_CLASSIFY_INIT.m'. It implements the fine tuning for a classification net with backpropagation using a direct translation of the minimize function from C. Rassmussen (available at http://www.gatsby.ucl.ac.uk/~edward/code/minimize/ - last visit 06.06.2013) to R. The parameter switchLayers is for the switch between two training type. Like in the original code, the top two layers can be trained alone until epoch is equal to epochSwitch. Afterwards the entire network will be trained.

mseError 41

Value

The trained DArch object.

See Also

DArch

mseError

Mean quared error function

Description

The function calculates the mean quared error (MSE) from the original and estimate parameters.

Usage

```
mseError(original, estimate)
```

Arguments

original The original data matrix estimate The calculated data matrix

Value

A list with the name of the error function in the first entry and the error value in the second entry

See Also

quadraticError, crossEntropyError

Net

Abtract class for neural networks.

Description

This is a abstract class for neural networks. It provides some functionalitys used in more than one network type.

Slot

batchSize: Object of class "numeric". The batch size for the training and test data during the learning.

errorFunction: Object of class "function". Function for error calculation.

ff: Object of class "logical". Indicates if the package ff is used to save the network data.

genWeightFunction: Object of class "function". A function for generate random initialised weight matrix.

42 newDArch

Author(s)

Martin Drees

See Also

DArch, RBM

newDArch

Constructor function for DArch objects.

Description

Generate a new DArch object with the given parameters.

Usage

```
newDArch(layers,batchSize,ff=FALSE,
logLevel=INFO, genWeightFunc=generateWeights)
```

Arguments

layers Array of layer sizes.
batchSize Size of the batches

ff Indicates whether the ff package is used for the weights, biases and outputs

logLevel The logging level. See setLogLevel for details. genWeightFunc The function for generating the weight matrices

Details

It is recommended to use this function for generating a new DArch object, because this function generates and sets all the necessary parameters like the internally used RBM networks, the list of statistiks (stats) etc.

Value

The new DArch object

newRBM 43

newRBM	Constructor function for RBM object.
--------	--------------------------------------

Description

```
TODO: Doc ...
```

Usage

```
newRBM(numVisible, numHidden, batchSize, ff = FALSE, logLevel = INFO,
  genWeightFunc = generateWeights)
```

Arguments

numVisible Number of visible units.

numHidden Number of hidden units.

batchSize Size of the batches

ff Indicates whether the ff package is used for the weights, biases and outputs

logLevel The logging level. See setLogLevel for details.

genWeightFunc The function for generating the weight matrices

Value

The new RBM object

preTrainDArch	Pre trains a DArch network	

Description

This function pre trains a DArch network with the contrastive divergence method

Usage

```
preTrainDArch(darch,trainData,maxEpoch=1,numCD=1,...)
## S4 method for signature 'DArch'
preTrainDArch(darch, trainData, maxEpoch = 1, numCD = 1,
...)
```

44 quadraticError

Arguments

darch A instance of the class DArch.

trainData The data matrix for the training

maxEpoch The number of epochs

numCD The number of CD iterations

... Additional parameters for the function trainRBM

Details

The function runs for every RBM in the attribute rbmList the training function trainRBM copies after the training the weights and biases into the corresponding layer of the DArch network.

See Also

DArch, RBM, trainRBM

quadraticError

Quadratic error function

Description

The function calculates the quadradic error from the original and estimate parameters.

Usage

```
quadraticError(original, estimate)
```

Arguments

original The original data matrix
estimate The calculated data matrix

Value

A list with the name of the error function in the first entry and the error value in the second entry

See Also

mseError, crossEntropyError

RBM

Class for Restricted-Bolzmann-Machine

Description

This class represents a Restricted-Bolzmann-Machine

Slots

learnRateBiasVisible: Object of class "numeric". Learning rate of the visible biases.

learnRateBiasHidden: Object of class "numeric". Learning rate of the hidden biases.

weightCost: Object of class "numeric". Weigth cost for the update of the weigths.

numHidden: Object of class "numeric". Number of hidden units.

numVisible: Object of class "numeric". Number of visible units.

weights: Object of class "matrix". Weight matrix.

weightInc: Object of class "matrix". Matrix of update values for the Weight.

output: Object of class "matrix". Output matrix of the RBM.

visibleBiases: Object of class "array". Visible biases array.

visibleBiasesInc: Object of class "array". Array of update values for the visible biases

visibleUnitFunction: Object of class "function". Unit function for the visible units.

visibleUnitStates: Object of class "list". States of the visible units.

hiddenBiases: Object of class "array". Hidden biases array.

hiddenBiasesInc: Object of class "array". Array of update values for the hidden biases.

hiddenUnitFunction: Object of class "function". Unit function for the hidden units.

hiddenUnitStates: Object of class "list". States of the hidden units.

updateFunction: Object of class "function". Function for updating the weights and biases.

posPhaseData: Object of class "list". Attribute to save the positive phase data during the training.

ffWeights: Object of class "ff_matrix". Weight ff matrix. Used when the ff attribute is TRUE.

ffOutput: Object of class "ff_matrix". Output ff matrix of the RBM. Used when the ff attribute is TRUE.

ffHiddenBiases: Object of class "ff_array". Hidden biases ff array. Used when the ff attribute is TRUE.

ffVisibleBiases: Object of class "ff_array". Hidden biases ff array. Used when the ff attribute is TRUE.

Author(s)

Martin Drees

See Also

Net, DArch, trainRBM

46 readMNIST

rbmUpdate

Function for updating the weights and biases of an RBM

Description

This function updates the weights and biases for an RBM network. It is saved in the attribute updateFunction of the RBM object and called from the training function trainRBM.

Usage

```
rbmUpdate(rbm)
```

Arguments

rbm

A instance of the class RBM.

Value

The updated RBM.

See Also

RBM

readMNIST

Function for generating ff files of the MNIST Database

Description

This function reads the MNIST-Database, randomized it and saved it in the files "train" for the training data and "test" for test data.

Usage

```
readMNIST(folder)
```

Arguments

folder

The location of the MNIST-Database files.

Details

When the data is read the variables for the training data is trainData and trainLabels and for the test data testData and testLabels. To start the function The files "train-images-idx3-ubyte", "train-labels-idx1-ubyte', "t10k-images-idx3-ubyte", and "t10k-labels-idx1-ubyte" have to be in the folder given by the parameter folder. The folder name must end with a slash.

removeLayerField 47

removeLayerField

Removes a layer from the DArch object

Description

This function removes the layer with the given index from the DArch object.

Usage

```
removeLayerField(darch, index)
## S4 method for signature 'DArch'
removeLayerField(darch, index)
```

Arguments

darch A instance of the class DArch.

index The index of the layer.

Value

The DArch object without the layer.

See Also

DArch

resetDArch

Resets the weights and biases of the DArch object

Description

This function resets the weights and biases of the DArch object and all RBM objects if the parameter resetRBMs is TRUE.

Usage

```
resetDArch(darch,resetRBMs=TRUE)
## S4 method for signature 'DArch'
resetDArch(darch, resetRBMs = TRUE)
```

Arguments

darch A instance of the class DArch.

resetRBMs If true the RBMs are also reseted.

48 resetRBM

Details

When the parameter resetRBMs is FALSE then the trained weights and biases are copied from the RBM objects to the layers.

See Also

DArch

resetExecOutput

Resets the output list of the DArch object

Description

This function sets the attribute executeOutput of the DArch object to an empty list.

Usage

```
resetExecOutput(darch)
## S4 method for signature 'DArch'
resetExecOutput(darch)
```

Arguments

darch

A instance of the class DArch.

See Also

DArch

resetRBM

Resets the weights and biases of the RBM object

Description

This function resets the weights and biases of the RBM object.

Usage

```
resetRBM(rbm)
## S4 method for signature 'RBM'
resetRBM(rbm)
```

rpropagation 49

Arguments

rbm A instance of the class RBM.

See Also

RBM

rpropagation Resilient-Backpropgation training for deep architectures.

Description

The function traines a deep architecture with the resilient backpropagation algorithm. It is able to use four different types of training (see details). For details of the resilient backpropagation algorith see the references.

Usage

```
rpropagation(darch,trainData,targetData,epoch,method="iRprop+",
decFact=0.5,incFact=1.2,weightDecay=0,
initDelta=0.0125,minDelta=0.000001,maxDelta=50)
```

Arguments

darch The deep architecture to train

trainData The training data

targetData The expected output for the training data

epoch The number of training iterations

method The method for the training. Default is "iRprop+" decFact Decreasing factor for the training. Default is 0.5. incFact Increasing factor for the training Default is 1.2.

weightDecay Weight decay for the training. Default is 0

initDelta Initialisation value for the update. Default is 0.0125.

minDelta Lower bound for step size. Default is 0.000001

maxDelta Upper bound for step size. Default is 50

50 runDArch

Details

The code for the calculation of the weight change is a translation from the matlab code from the Rprop Optimization Toolbox implemented by R. Calandra (see References).

Copyright (c) 2011, Roberto Calandra. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. 4. If used in any scientific publications, the publication has to refer specifically to the work published on this webpage.

This software is provided by us "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for particular purpose are disclaimed. In no event shall the copyright holders or any contributor be liable for any direct, indirect, incidental, special, exemplary, or consequential damages however caused and on any theory of liability whether in contract, strict liability or tort arising in any way out of the use of this software, even if advised of the possibility of such damage.

Value

darch - The trained deep architecture

References

M. Riedmiller, H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, pp 586-591. IEEE Press, 1993.

C. Igel, M. Huesken. Improving the Rprop Learning Algorithm, Proceedings of the Second International Symposium on Neural Computation, NC 2000, ICSC Academic Press, Canada/Switzerland, pp. 115-121., 2000.

Kohavi, R., A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, Proceedings of the 14th Int. Joint Conference on Artificial Intelligence 2, S. 1137-1143, Morgan Kaufmann, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

See Also

DArch

runDArch

Execute the darch

Description

Runs the darch in a feed forward manner and saves the generated outputs for every layer in the list executeOutput from the darch. To get the outputs call

saveDArch 51

Usage

```
runDArch(darch, data)
```

Arguments

darch A instance of the class DArch.

data The input data to execute the darch on.

Value

The DArch object with the calculated outputs

See Also

DArch

saveDArch

Saves a DArch network

Description

Saves the DArch object to the filename given through the parameter name plus the ending ".net".

Usage

```
saveDArch(darch,name="darch",saveRBM=TRUE)
## S4 method for signature 'DArch'
saveDArch(darch, name = "darch", saveRBM = TRUE)
```

Arguments

darch A instance of the class DArch.

name The name for the file. Default value is "darch". saveRBM Boolean value to indicate if the RBM's are saved.

Details

If the field ff of the DArch object is TRUE then the weights are saved in seperate ff-files named by the parameter name plus the string "-W" and the number of the layer. In the same way the weights from the RBM's of the DArch are saved, but only if the parameter saveRBM is TRUE. For more information about the how the weights and biases from the RBM's are saved see saveRBMFFWeights. If the parameter saveRBM is FALSE the field rbmList of the DArch object ist overwritten by an empty list.

See Also

loadDArch, saveRBMFFWeights

52 saveRBMFFWeights

saveRBM

Saves a RBM network

Description

Saves the RBM object to the filename given through the parameter name plus the ending ".net".

Usage

```
saveRBM(rbm,name="rbm")
## S4 method for signature 'RBM'
saveRBM(rbm, name = "rbm")
```

Arguments

rbm A instance of the class RBM.

name The name for the file. Default value is "rbm".

Details

If the field ff of the RBM object is TRUE then the weights are saved in seperate ff-files through the funktion saveRBMFFWeights.

See Also

loadRBM, saveRBMFFWeights loadRBMFFWeights

saveRBMFFWeights

Saves weights and biases of a RBM network into a ffData file.

Description

Saves the weigths and the biases for the given RBM object to the filename given through the parameter name.

Usage

```
saveRBMFFWeights(rbm,name="saveName")
## S4 method for signature 'RBM'
saveRBMFFWeights(rbm, name = "saveName")
```

Arguments

rbm A instance of the class RBM.

name The name for the file.

setBatchSize<- 53

Details

The weights and biases are saved in one file with the name given through the parameter name and the string "-WB". See ffsave for more details.

See Also

ffsave, loadRBM, saveRBM, loadRBMFFWeights

setBatchSize<-

Sets the batch size of the Net.

Description

Sets the batch size of the Net.

Usage

```
setBatchSize(net) <- value</pre>
```

Arguments

net A instance of the class Net.
value Object of the class numeric.

See Also

Net

setCancel<-

Set whether the learning shall be canceled.

Description

Set whether the learning shall be canceled.

Usage

```
setCancel(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.

value Boolean value if the learning shall canceled

See Also

54 setErrorFunction<-

setCancelMessage<- Sets the cancel message.

Description

Sets the cancel message.

Usage

```
setCancelMessage(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.

value The message for the termination

See Also

DArch

setErrorFunction<- Sets the error function of the Net.

Description

Sets the error function of the Net.

Usage

```
setErrorFunction(net) <- value</pre>
```

Arguments

net A instance of the class Net.
value Object of the class function.

See Also

Net

setExecuteFunction<- 55

setExecuteFunction<- Sets the execution function for the network

Description

Sets the execution function for the network

Usage

```
setExecuteFunction(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.

value The execution function for the network.

See Also

DArch

setFF<- Sets if the weights are saved as ff objects

Description

Sets if the weights are saved as ff objects

Usage

```
setFF(net) <- value</pre>
```

Arguments

net A instance of the class Net.

value Boolean value which indicates if the weights are saved as ff objects

See Also

Net

56 setFineTuneFunction<-

setFinalMomentum<-

Sets the final momentum of the Net.

Description

Sets the final momentum of the Net.

Usage

```
setFinalMomentum(net) <- value</pre>
```

Arguments

net A instance of the class Net.

value Object of the class numeric.

See Also

Net

 $\verb|setFineTuneFunction| <- Sets the fine tuning function for the network|$

Description

Sets the fine tuning function for the network

Usage

```
setFineTuneFunction(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.

value The fine tuning function for the network.

See Also

setGenWeightFunction<-

Sets the function for generating weight matrices.

Description

The function have to return a matrix with number of units in the lower layer as number of rows and number of units in the upper layer as the nubmer of columns.

Usage

```
setGenWeightFunction(net) <- value</pre>
```

Arguments

net A instance of the class Net.
value Object of the class function.

See Also

Net

setHiddenBiases<-

Sets the biases of the hidden units for the RBM object

Description

Sets the biases of the hidden units for the RBM object

Usage

```
setHiddenBiases(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The biases of the hidden units for the RBM object.

See Also

58 setHiddenUnitFunction<-

setHiddenBiasesInc<- Sets the update value for the biases of the hidden units

Description

Sets the update value for the biases of the hidden units

Usage

```
setHiddenBiasesInc(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The update value for the biases of the hidden units.

See Also

RBM

setHiddenUnitFunction<-

Sets the unit function of the hidden units

Description

Sets the unit function of the hidden units

Usage

```
setHiddenUnitFunction(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The unit function of the hidden units

See Also

setHiddenUnitStates<- 59

setHiddenUnitStates<- Sets the states of the hidden units</pre>

Description

Sets the states of the hidden units

Usage

```
setHiddenUnitStates(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.
value The states of the hidden units

See Also

RBM

setLayer<-

Sets a layer with the given index for the network

Description

Sets a layer with the given index for the network

Usage

```
setLayer(darch,index) <- value</pre>
```

Arguments

darch A instance of the class DArch.

index The index of the layer value The layer for the network.

See Also

60 setLayerFunction<-

setLayerField<- Sets a field in a layer.

Description

Sets the field on position fieldIndex of the layer given by the layerIndex to the value.

Usage

```
setLayerField(darch,layerIndex,fieldIndex) <- value</pre>
```

Arguments

darch A instance of the class DArch.

layerIndex The index of the layer fieldIndex The index of the field

value The value for the layer field

Value

The darch with the updated layer

See Also

DArch

setLayerFunction<- Sets the function for a layer with the given index

Description

Sets the function for a layer with the given index

Usage

```
setLayerFunction(darch,index) <- value</pre>
```

Arguments

 $\mbox{darch} \qquad \mbox{A instance of the class DArch}.$

index The index of the layer.

value The function for the layer.

See Also

setLayers<-

setLayers<-

Sets the layers for the network

Description

Sets the layers for the network

Usage

```
setLayers(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.
value The layers for the network.

See Also

DArch

setLayerWeights<-

Sets the weights of a layer with the given index

Description

Sets the weights of a layer with the given index

Usage

```
setLayerWeights(darch,index) <- value</pre>
```

Arguments

darch A instance of the class DArch.

index The index of the layer.

value The weights for the layer.

See Also

setLearnRateBiasHidden<-

setLearnRateBiases<- Sets the learning rate for the biases

Description

Sets the learning rate for the biases

Usage

```
setLearnRateBiases(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.

value The learning rate for the biases.

See Also

DArch

setLearnRateBiasHidden<-

Sets the learnig rates of the biases for the hidden units

Description

Sets the learnig rates of the biases for the hidden units

Usage

```
setLearnRateBiasHidden(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The learnig rates of the biases for the hidden units

See Also

setLearnRateBiasVisible<- 63

setLearnRateBiasVisible<-

Sets the learnig rates of the biases for the visible units

Description

Sets the learnig rates of the biases for the visible units

Usage

```
setLearnRateBiasVisible(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The learnig rates of the biases for the visible units

See Also

RBM

setLearnRateWeights<- Sets the learning rate for the weights.

Description

Sets the learning rate for the weights.

Usage

```
setLearnRateWeights(net) <- value</pre>
```

Arguments

net A instance of the class Net.
value Object of the class numeric.

See Also

Net

setLogLevel<-

Sets the log level for the Net.

Description

The log levels a defined by the futile.logger package. The following levels a available:

64 setMomentum<-

TRACE DEBUG INFO WARN ERROR FATAL

Usage

```
setLogLevel(net) <- value</pre>
```

Arguments

net A instance of the class Net.
value Object of the class numeric.

See Also

Net

setMomentum<-

Sets the momentum of the Net.

Description

Sets the momentum of the Net.

Usage

```
setMomentum(net) <- value</pre>
```

Arguments

net A instance of the class Net.
value Object of the class numeric.

See Also

Net

setMomentumSwitch<- 65

setMomentumSwitch<-

Sets the momentum switch of the Net.

Description

Sets the momentum switch of the Net.

Usage

```
setMomentumSwitch(net) <- value</pre>
```

Arguments

net A instance of the class Net.

value Object of the class numeric.

See Also

Net

setNumHidden<-

Sets the number of hidden units

Description

Sets the number of hidden units

Usage

```
setNumHidden(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.
value The number of hidden units

See Also

setOutput<-

setNumVisible<-

Sets the number of visible units

Description

Sets the number of visible units

Usage

```
setNumVisible(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The number of visible units

See Also

RBM

setOutput<-

Sets the output of the RBM object

Description

Sets the output of the RBM object

Usage

```
setOutput(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.
value The output of the RBM object

See Also

setPosPhaseData<-

setPosPhaseData<-

Sets the positive phase data for the training

Description

Sets the positive phase data for the training

Usage

```
setPosPhaseData(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The positive phase data for the training

See Also

RBM

setRBMList<-

Sets the list of RBMs

Description

Sets the list of RBMs

Usage

```
setRBMList(darch) <- value</pre>
```

Arguments

darch A instance of the class DArch.

value The list of RBMs.

See Also

68 setUpdateFunction<-

setStats<-

Adds a list of statistics to the network

Description

The list of statistics can contain values about errors, miss classifications and other usefull things from the pre-training or fine-tuning of a deep architecture.

Usage

```
setStats(net) <- value</pre>
```

Arguments

net A instance of the class Net.

value Statistics for the Net.

setUpdateFunction<-

Sets the update function of the RBM object

Description

Sets the update function of the RBM object

Usage

```
setUpdateFunction(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The update function of the RBM object.

See Also

setVisibleBiases<-

setVisibleBiases<-

Sets the biases of the visible units for the RBM object

Description

Sets the biases of the visible units for the RBM object

Usage

```
setVisibleBiases(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The biases of the visible units for the RBM object.

See Also

RBM

setVisibleBiasesInc<- Sets the update value for the biases of the visible units

Description

Sets the update value for the biases of the visible units

Usage

```
setVisibleBiasesInc(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The update value for the biases of the visible units.

See Also

70 setVisibleUnitStates<-

setVisibleUnitFunction<-

Sets the unit function of the visible units

Description

Sets the unit function of the visible units

Usage

```
setVisibleUnitFunction(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The unit function of the visible units

See Also

RBM

setVisibleUnitStates<-

Sets the states of the visible units

Description

Sets the states of the visible units

Usage

```
setVisibleUnitStates(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.
value The states of the visible units

See Also

setWeightCost<- 71

setWeightCost<-

Sets the weight costs for the training

Description

Sets the weight costs for the training

Usage

```
setWeightCost(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The weight costs for the training

See Also

RBM

setWeightInc<-

Sets the update values for the weights

Description

Sets the update values for the weights

Usage

```
setWeightInc(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.

value The update values for the weights

See Also

72 sigmoidUnit

setWeights<-

Sets the weights of the RBM object

Description

Sets the weights of the RBM object

Usage

```
setWeights(rbm) <- value</pre>
```

Arguments

rbm A instance of the class RBM.
value The weights of the RBM object.

See Also

RBM

sigmoidUnit

Sigmoid unit function.

Description

The function calculates the activation and returns the result through the sigmoid transfer function.

Usage

```
sigmoidUnit(data,weights)
```

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

Value

A list with the activation of the unit in the first entry.

See Also

 ${\tt DArch, bin Sigmoid Unit, sigmoid Unit Derivative, linear Unit Derivative, softmax Unit, softmax Unit Derivative} \\$

sigmoidUnitDerivative 73

sigmoidUnitDerivative Sigmoid unit function with unit derivatives.

Description

The function calculates the activation and returns a list which the first entry is the result through the sigmoid transfer function and the second entry is the derivative of the transfer function.

Usage

```
sigmoidUnitDerivative(data,weights)
```

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

Value

A list with the activation in the first entry and the derivative of the transfer function in the second entry

See Also

DArch, sigmoid Unit, bin Sigmoid Unit, linear Unit Derivative, softmax Unit, softmax Unit Derivative, softmax Unit, softmax Unit Derivative, sof

sigmUnitFunc	Calculates the neuron output with the sigmoid function

Description

Calculates the neuron output with the sigmoid function from binary input saved in the second entry of the list dataList.

Usage

```
sigmUnitFunc(rbm, dataList, biases, weights, runParams)
```

Arguments

rbm A instance of the class RBM.

dataList A list with the data matrices for the calculations.

biases The biases for the calculations

weights The weight matrix for the calculations

runParams Parameters which indicates the status of the training.

Details

The return value is a list with the output of the sigmoid function as first entry and binary representation calculated through a comparison of the output with random numbers. The random numbers a generated with the function runif.

Value

The real value and binary activations for the units

See Also

DArch

sigmUnitFuncSwitch

Calculates the neuron output with the sigmoid function

Description

Calculates the neuron output with the sigmoid function either from the real value or binary input saved in the list dataList.

Usage

sigmUnitFuncSwitch(rbm, dataList, biases, weights, runParams)

Arguments

rbm A instance of the class RBM.

dataList A list with the data matrices for the calculations.

biases The biases for the calculations

weights The weight matrix for the calculations

runParams Parameters which indicates the status of the training. "actualCD" and "finishCD"

are needed (see trainRBM)

Details

The return value is a list with the output of the sigmoid function as first entry and binary representation calculated through a comparison of the output with random numbers. The random numbers a generated with the function runif. If the parameter runParams["actualCD"] or runParams["finishCD"] is equal one, the calculation is made with the real value data (dataList[[1]]), otherwise with the binary representations (dataList[[2]]).

Value

The real value and binary activations for the units

softmaxUnit 75

See Also

DArch

softmaxUnit Softmax unit function.

Description

The function calculates the activation of the units and returns a list, in which the first entry is the result through the softmax transfer function.

Usage

```
softmaxUnit(data,weights)
```

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

Value

A list with the softmax activation in the first entry

See Also

DArch, sigmoidUnit, binSigmoidUnit, sigmoidUnitDerivative, linearUnit, linearUnitDerivative, softmaxUnitDerivative

softmaxUnitDerivative Softmax unit function with unit derivatives.

Description

The function calculates the activation of the units and returns a list, in which the first entry is the result through the softmax transfer function and the second entry is the derivative of the transfer function.

Usage

softmaxUnitDerivative(data, weights)

Arguments

data The data matrix for the calculation

weights The weight and bias matrix for the calculation

76 trainRBM

Value

A list with the softmax activation in the first entry and the derivative of the transfer function in the second entry

See Also

DArch, sigmoidUnit, binSigmoidUnit, sigmoidUnitDerivative, linearUnit, linearUnitDerivative, softmaxUnit

trainRBM

Trains a RBM with contrastive divergence

Description

The function trains a restricted bolzmann machine (RBM) with the contrastive divergence method.

Usage

```
trainRBM(rbm,trainData,maxEpoch=1,numCD=1,...)
## S4 method for signature 'RBM'
trainRBM(rbm, trainData, maxEpoch = 1, numCD = 1, ...)
```

Arguments

rbm A instance of the class RBM.
trainData The data matrix for the training
maxEpoch The number of training iterations

numCD Number of contrastive divergence iterations
... Additional parameters for the unit functions

Details

This function is build on the basis of the code from G. Hinton et. al. (http://www.cs.toronto.edu/~hinton/MatlabForSciencePaj-last visit 06.06.2013) for the pre training of deep belief nets. The original code is located in the files 'rbm.m' and 'rbmhidlinear.m'. It iterates in every epoche over the batches and calculates the updates for the weights. If it is the first CD iteration or the CD iterations are finished, the hidden units are calculated with the real value activations of the visible units, otherwise with the binary activations. To tell the unit functions the actual state of the training, the function generates a array with the following running parameters and passes them to the units: Maximal epochs: "maxEpoch", Actual epochs: "actualEpoch", Number of batches: "numBatches", Actual batch: "actualBatch", Maximal CD iterations: "numCD", Actual CD iteration: "actualCD", CD is finished: "finishCD". (see source code from sigmUnitFuncSwitch for an example).

See Also

Index

*Topic Architectures darch, 9 *Topic Backpropagation darch, 9 *Topic Bolzmann	<pre>addLayer, 5 addLayer, DArch-method (addLayer), 5 addLayerField, 6 addLayerField, DArch-method</pre>
darch, 9	
*Topic Conjugate	backpropagation, $6, 8, 12, 18$
darch, 9	binSigmoidUnit, 5, 7, 22, 34, 35, 72, 73, 75,
*Topic Contrastive	76
darch, 9	
*Topic Deep-Belief-Networks	crossEntropyError, 8, 41, 44
darch, 9	
*Topic Deep	DArch, 5–7, 8, 11, 12, 14–16, 18, 21–23, 29,
darch, 9	34, 35, 39–45, 47, 48, 50, 51, 53–56,
*Topic Divergence	59–62, 67, 72–76
darch, 9	darch, 9
*Topic Gradient	DArch-class (DArch), 8
darch, 9	darch-package (darch), 9
*Topic Machines	
darch, 9	ff, <i>12</i> , <i>41–43</i>
*Topic NN	ffload, <i>37</i>
darch, 9	ffsave, <i>53</i>
*Topic Nets	fineTuneDArch, 8, 11, 18, 40
darch, 9	fineTuneDArch,DArch-method
*Topic Networks	(fineTuneDArch), 11
darch, 9	futile.logger, 63
*Topic Neural	
darch, 9	generateRBMs, 12
*Topic Resilient	generateRBMs,DArch-method
darch, 9	(generateRBMs), 12
*Topic Restricted	generateWeights, 13
darch, 9	getBatchSize, 13
*Topic darch	<pre>getBatchSize,Net-method(getBatchSize),</pre>
darch, 9	13
*Topic package	getCancel, 14
darch, 9	<pre>getCancel, DArch-method (getCancel), 14</pre>
,	getCancelMessage, 14
addExecOutput, 4	getCancelMessage,DArch-method
addExecOutput,DArch-method	(getCancelMessage), 14
(addExecOutput), 4	getErrorFunction, 15

<pre>getErrorFunction,Net-method</pre>	getLearnRateBiasHidden,RBM-method
(getErrorFunction), 15	(getLearnRateBiasHidden), 24
getExecOutput, 15	getLearnRateBiasVisible, 24
getExecOutput,DArch-method	getLearnRateBiasVisible,RBM-method
(getExecOutput), 15	(getLearnRateBiasVisible), 24
getExecOutputs, 16	getLearnRateWeights, 25
getExecOutputs,DArch-method	getLearnRateWeights,Net-method
(getExecOutputs), 16	(getLearnRateWeights), 25
getExecuteFunction, 16	getMomentum, 25
getExecuteFunction,DArch-method	getMomentum, Net-method (getMomentum), 25
(getExecuteFunction), 16	getMomentumSwitch, 26
getFF, 17	getMomentumSwitch,Net-method
getFF,Net-method (getFF), 17	(getMomentumSwitch), 26
getFinalMomentum, 17	getNumHidden, 26
getFinalMomentum, Net-method	<pre>getNumHidden,RBM-method(getNumHidden),</pre>
(getFinalMomentum), 17	26
getFineTuneFunction, 18	getNumVisible, 27
getFineTuneFunction,DArch-method	getNumVisible,RBM-method
(getFineTuneFunction), 18	(getNumVisible), 27
	getOutput, 28
getGenWeightFunction, 18 getGenWeightFunction, Net-method	getOutput, RBM-method (getOutput), 28
	getPosPhaseData, 28
(getGenWeightFunction), 18	getPosPhaseData,RBM-method
getHiddenBiases, 19	(getPosPhaseData), 28
getHiddenBiases, RBM-method	getRBMList, 8, 29
(getHiddenBiases), 19	getRBMList, DArch-method (getRBMList), 29
getHiddenBiasesInc, 19	getStats, 29
getHiddenBiasesInc,RBM-method	getStats, Net-method (getStats), 29
(getHiddenBiasesInc), 19	getVisibleBiases, 30
getHiddenUnitStates, 20	getVisibleBiases,RBM-method
getHiddenUnitStates, RBM-method	(getVisibleBiases), 30
(getHiddenUnitStates), 20	getVisibleBiasesInc, 30
getLayer, 21	getVisibleBiasesInc,RBM-method
getLayer, DArch-method (getLayer), 21	(getVisibleBiasesInc), 30
getLayerField, 21	getVisibleUnitStates, 31
getLayerField, DArch-method	getVisibleUnitStates,RBM-method
(getLayerField), 21	(getVisibleUnitStates), 31
getLayerFunction, 22	getWeightCost, 32
<pre>getLayerFunction,DArch-method</pre>	getWeightCost,RBM-method
(getLayerFunction), 22	(getWeightCost), 32
getLayers, 22	getWeightInc, 32
<pre>getLayers, DArch-method (getLayers), 22</pre>	<pre>getWeightInc,RBM-method(getWeightInc),</pre>
getLayerWeights, 23	32
getLayerWeights,DArch-method	getWeights, 33
(getLayerWeights), 23	getWeights, RBM-method (getWeights), 33
getLearnRateBiases, 23	
getLearnRateBiases,DArch-method	linearUnit, 5, 7, 34, 35, 73, 75, 76
(getLearnRateBiases), 23	linearUnitDerivative, 5, 7, 22, 34, 34, 72,
getLearnRateBiasHidden, 24	73, 75, 76

linearUnitFunc, 35	<pre>setBatchSize(setBatchSize<-), 53</pre>
loadDArch, 36, 51	setBatchSize<-,53
loadRBM, 36, 37, 52, 53	setBatchSize<-,Net-method
loadRBMFFWeights, 36, 37, 37, 52, 53	(setBatchSize<-), 53
	setCancel (setCancel<-), 53
makeStartEndPoints, 37	setCancel<-, 53
minimize, 38, 40	setCancel<-,DArch-method(setCancel<-)
minimizeAutoencoder, 7, 12, 18, 39, 39	53
minimizeClassifier, 7, 12, 18, 39, 40	<pre>setCancelMessage (setCancelMessage<-),</pre>
mseError, 8, 41, 44	54
N + 0 12 12 15 17 10 25 26 20 20 41	setCancelMessage<-, 54
Net, 9, 12, 13, 15, 17, 18, 25, 26, 29, 38, 41,	setCancelMessage<-,DArch-method
45, 53–57, 63–65, 68	(setCancelMessage<-), 54
Net-class (Net), 41	setErrorFunction (setErrorFunction<-),
newDArch, 8, 42	54
newRBM, 43	setErrorFunction<-,54
nnoTnoinDAmoh 8 42	setErrorFunction<-,Net-method
preTrainDArch, 8, 43	(setErrorFunction<-), 54
preTrainDArch, DArch-method	setExecuteFunction (354
(preTrainDArch), 43	(setExecuteFunction<-), 55
quadraticError, 8, 41, 44	setExecuteFunction<-, 55
quadratici (01, 0, 71, 11	setExecuteFunction<-, DArch-method
RBM, 8, 9, 12, 19, 20, 24–33, 35, 37, 42, 44, 45,	(setExecuteFunction<-), 55
46–49, 52, 57–59, 62, 63, 65–74, 76	•
RBM-class (RBM), 45	setFF (setFF<-), 55
rbmUpdate, 46	setFF<-, 55
readMNIST, 46	setFF<-, Net-method (setFF<-), 55
removeLayerField, 47	<pre>setFinalMomentum(setFinalMomentum<-),</pre>
removeLayerField, DArch-method	56
(removeLayerField), 47	setFinalMomentum<-, 56
resetDArch, 47	setFinalMomentum<-,Net-method
resetDArch, DArch-method (resetDArch), 47	(setFinalMomentum<-), 56
resetExecOutput, 48	setFineTuneFunction
resetExecOutput,DArch-method	(setFineTuneFunction<-), 56
(resetExecOutput), 48	setFineTuneFunction<-, 56
resetRBM, 48	setFineTuneFunction<-,DArch-method
resetRBM, RBM-method (resetRBM), 48	(setFineTuneFunction<-), 56
rnorm, 35	setGenWeightFunction
rpropagation, 7, 12, 18, 49	(setGenWeightFunction<-),57
runDArch, 8, 50	<pre>setGenWeightFunction<-, 57</pre>
runif, 7, 74	<pre>setGenWeightFunction<-,Net-method</pre>
	(setGenWeightFunction<−), 57
saveDArch, <i>36</i> , 51	setHiddenBiases(setHiddenBiases<-), 57
<pre>saveDArch,DArch-method(saveDArch),51</pre>	setHiddenBiases<-, 57
saveRBM, 36, 37, 52, 53	setHiddenBiases<-,RBM-method
saveRBM, RBM-method (saveRBM), 52	(setHiddenBiases<-), 57
saveRBMFFWeights, <i>37</i> , <i>51</i> , <i>52</i> , 52	setHiddenBiasesInc
saveRBMFFWeights,RBM-method	(setHiddenBiasesInc<-), 58
(saveRBMFFWeights).52	setHiddenBiasesInc<-,58

setHiddenBiasesInc<-,RBM-method	(setLearnRateWeights<-), 63
(setHiddenBiasesInc<-), 58	setLearnRateWeights<-,63
setHiddenUnitFunction	setLearnRateWeights<-,Net-method
(setHiddenUnitFunction<−), 58	(setLearnRateWeights<-), 63
setHiddenUnitFunction<-,58	setLogLevel, 42, 43
setHiddenUnitFunction<-,RBM-method	setLogLevel (setLogLevel<-), 63
(setHiddenUnitFunction<-), 58	setLogLevel<-, 63
setHiddenUnitStates	setLogLevel<-,Net-method
(setHiddenUnitStates<-), 59	(setLogLevel<-), 63
setHiddenUnitStates<-,59	setMomentum (setMomentum<-), 64
setHiddenUnitStates<-,RBM-method	setMomentum<-, 64
(setHiddenUnitStates<-), 59	setMomentum<-,Net-method
setLayer (setLayer<-), 59	(setMomentum<-), 64
	setMomentumSwitch
setLayer<-, 59	
setLayer<-, DArch-method (setLayer<-), 59	(setMomentumSwitch<-), 65
setLayerField (setLayerField<-), 60	setMomentumSwitch<-,65
setLayerField<-,60	setMomentumSwitch<-,Net-method
setLayerField<-,DArch-method	(setMomentumSwitch<-), 65
(setLayerField<-), 60	setNumHidden(setNumHidden<-), 65
<pre>setLayerFunction (setLayerFunction<-),</pre>	setNumHidden<-,65
60	setNumHidden<-,RBM-method
setLayerFunction <-, 60	(setNumHidden<−), 65
setLayerFunction<-,DArch-method	<pre>setNumVisible (setNumVisible<-), 66</pre>
(setLayerFunction<-), 60	setNumVisible<-,66
setLayers (setLayers<-), 61	setNumVisible<-,RBM-method
setLayers<-,61	(setNumVisible<-), 66
<pre>setLayers<-,DArch-method(setLayers<-),</pre>	setOutput (setOutput<-), 66
61	setOutput<-, 66
setLayerWeights (setLayerWeights<-), 61	setOutput<-,RBM-method(setOutput<-),66
setLayerWeights<-,61	setPosPhaseData(setPosPhaseData<-), 67
setLayerWeights<-,DArch-method	setPosPhaseData<-, 67
(setLayerWeights<-), 61	setPosPhaseData<-,RBM-method
setLearnRateBiases	(setPosPhaseData<-), 67
(setLearnRateBiases<-), 62	setRBMList (setRBMList<-), 67
setLearnRateBiases<-, 62	setRBMList<-,67
setLearnRateBiases<-,DArch-method	setRBMList<-,DArch-method
(setLearnRateBiases<-), 62	(setRBMList<-), 67
setLearnRateBiasHidden	setStats (setStats<-), 68
(setLearnRateBiasHidden<-), 62	setStats<-, 68
setLearnRateBiasHidden<-,62	<pre>setStats<-,Net-method(setStats<-),68</pre>
setLearnRateBiasHidden<-,RBM-method	setUpdateFunction
(setLearnRateBiasHidden<-), 62	(setUpdateFunction < -), 68
setLearnRateBiasVisible	setUpdateFunction<-,68
(setLearnRateBiasVisible<-), 63	setUpdateFunction<-,RBM-method
setLearnRateBiasVisible<-,63	(setUpdateFunction<-), 68
setLearnRateBiasVisible<-,RBM-method	<pre>setVisibleBiases (setVisibleBiases<-),</pre>
(setLearnRateBiasVisible<-), 63	69
setLearnRateWeights	setVisibleBiases<69

```
setVisibleBiases<-,RBM-method
         (setVisibleBiases<-), 69
setVisibleBiasesInc
         (setVisibleBiasesInc<-), 69
setVisibleBiasesInc<-, 69
setVisibleBiasesInc<-,RBM-method</pre>
         (setVisibleBiasesInc<-), 69
setVisibleUnitFunction
         (setVisibleUnitFunction<-), 70
setVisibleUnitFunction<-,70
setVisibleUnitFunction<-,RBM-method</pre>
         (setVisibleUnitFunction<-), 70</pre>
setVisibleUnitStates
         (setVisibleUnitStates<-), 70</pre>
setVisibleUnitStates<-, 70</pre>
setVisibleUnitStates<-,RBM-method
         (setVisibleUnitStates<-), 70</pre>
setWeightCost (setWeightCost<-), 71</pre>
setWeightCost<-,71</pre>
setWeightCost<-,RBM-method</pre>
         (setWeightCost<-), 71
setWeightInc(setWeightInc<-), 71</pre>
setWeightInc<-,71</pre>
setWeightInc<-,RBM-method</pre>
        (setWeightInc<-), 71</pre>
setWeights (setWeights<-), 72</pre>
setWeights<-,72
setWeights<-,RBM-method(setWeights<-),</pre>
         72
sigmoidUnit, 5, 7, 22, 34, 35, 72, 73, 75, 76
sigmoidUnitDerivative, 5, 7, 22, 34, 35, 72,
         73, 75, 76
sigmUnitFunc, 73
sigmUnitFuncSwitch, 74, 76
softmaxUnit, 5, 7, 22, 34, 35, 72, 73, 75, 76
softmaxUnitDerivative, 5, 7, 22, 34, 35, 72,
         73, 75, 75
trainRBM, 44-46, 74, 76
trainRBM, RBM-method (trainRBM), 76
```