Package: TeachNet (via r-universe)

September 16, 2024

Type Package

Title Fits Neural Networks to Learn About Backpropagation

Version 0.7.1

Date 2018-11-20

Author Georg Steinbuss

Maintainer Georg Steinbuss <gspam@steinbuss.de>

Depends methods

Description Can fit neural networks with up to two hidden layer and two different error functions. Also able to handle a weight decay. But just able to compute one output neuron and very slow.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-11-27 16:30:11 UTC

Contents

`TeachNet-package` *Fit neural networks with up to 2 hidden layers and one output neuron*

Description

Can fit neural networks with up to two hidden layers and two different error functions. But just able to compute one output neuron. Also able to handle a weight decay.

Details

Package:	TeachNet
Type:	Package
Version:	0.7
Date:	2013-11-20
License:	GPL (>= 2)

The function TeachNet trains the neural network and also does some testing at the end. It's also possible to get the final weights returned. In the beginning the weights are initialized with a standard normal distribution. But this package is due to its very slow code just to understand the backpropagation algorithm. A good package for real training of neural networks is for example 'nnet'.

Author(s)

Georg Steinbuss

Maintainer: Who to complain to <gspam@steinbuss.de>

References

Predicting credit default using neural networks, Georg Steinbuss 2013

accuracy.me *Computes accuracy*

Description

For a given observation and prediction this function computes the accuracy of the prediction.

Usage

```
accuracy.me(obs, predict, thres = 0.5)
```

Arguments

obs	The observations
predict	The predictions for the observations
thres	A threshold up to which a prediction is class 0 or 1. A value from 0 to 1.

Value

Returns a 1 x 3 matrix with the percentage of observations with class zero, with class one and last the accuracy of the prediction.

Author(s)

Georg Steinbuss

See Also

confusion

computeGrad1 *Computes a gradient*

Description

This function computes the gradient for a one hidden layer network.

Usage

```
computeGrad1(x, y, I, H, weights, f, f_d, m_f)
```

Arguments

x	properties of observation
y	characteristic of observation (zero or one)
I	numbers of input neurons
H	numbers of hidden neurons
weights	the weights with that the gradient should be computed
f	the activation function of the neural network
f_d	the derivative of the activation function
m_f	the function for the interim value m. It is two times the output of the network minus the observed characteristic.

Value

A Weights class with the gradient parts

Author(s)

Georg Steinbuss

See Also

Weights-class computeGrad2

---

computeGrad2 *Computes a gradient*

Description

This function computes the gradient for a two hidden layer network.

Usage

```
computeGrad2(x, y, I, M, H, weights, f, f_d, m_f)
```

Arguments

x	properties of observation
y	characteristic of observation (zero or one)
I	numbers of input neurons
M	number of neurons in first hidden layer
H	number of neurons in second hidden layer
weights	the weights with that the gradient should be computed
f	the activation function of the neural network
f_d	the derivative of the activation function
m_f	the function for the interim value m. It is two times the output of the network minus the observed characteristic.

Value

A Weights2 class with the gradient parts

Author(s)

Georg Steinbuss

See Also

Weights-class computeGrad2

`computeOutput1` *Computes output*

Description

Computes output (prediction) for a one hidden layer network for one observation

Usage

```
computeOutput1(x, weights)
```

Arguments

`x`	properties of observation
`weights`	weights of the neural network

Value

Returns a single numeric value.

Author(s)

Georg Steinbuss

computeOutput2 *Computes output*

Description

Computes output (prediction) for a two hidden layers network for one observation

Usage

```
computeOutput2(x, weights)
```

Arguments

x	properties of observation
weights	weights of the neural network

Value

Returns a single numeric value.

Author(s)

Georg Steinbuss

confusion *Computes confusion matrix*

Description

Computes confusion matrix for a specific threshold

Usage

```
confusion(pred, obs, threshold = 0.5)
```

Arguments

pred	the prediction
obs	the observation
threshold	A threshold up to which a prediction is class 0 or 1. A value from 0 to 1

Value

Returns a confusion matrix

Author(s)

Georg Steinbuss

createWeights1 *Creates random weights*

Description

Creates random weights for a single hidden layer network

Usage

```
createWeights1(I, H)
```

Arguments

I	number of input neurons
H	number of hidden neurons

Value

Returns a S4 class object Weights

Author(s)

Georg Steinbuss

See Also

Weights-class

createWeights2 *Creates random weights*

Description

Creates random weights for a two hidden layers network

Usage

```
createWeights2(I, H)
```

Arguments

I	number of input neurons
H	vector with first element the number of hidden neurons in the first hidden layer second element for the second hidden layer

Value

Returns a S4 class object Weights2

Author(s)

Georg Steinbuss

See Also

Weights2-class

`crossEntropy` *Cross entropy*

Description

The error function cross entropy

Usage

```
crossEntropy(x, y)
```

Arguments

x	properties of observation
y	characteristic of observation (zero or one)

Value

returns a single numeric value

Author(s)

Georg Steinbuss

See Also

linksquaredError

find.Threshold *Finds best threshold*

Description

Finds the best threshold to transform probabilities in classes

Usage

```
find.Threshold(obs, stepsize = 0.1, predict)
```

Arguments

obs	class of observation
stepsize	in which step size the threshold is raised
predict	prediction of network

Author(s)

Georg Steinbuss

fitTeachNet1 *One step in backpropagation*

Description

One step in the backpropagation algorithm for a one hidden layer network

Usage

```
fitTeachNet1(data, weights, hidden.structure, learning.rate, f, f_d, decay, m_f, er)
```

Arguments

data	the data set
weights	current weights
hidden.structure	the number of neurons in the hidden layer
learning.rate	rate by which factor for backpropagation gets smaller
f	activation function
f_d	derivative of activation function
decay	value of weight decay
m_f	interim value m
er	error function

Value

returns new the weight after gradient update

Author(s)

Georg Steinbuss

`fitTeachNet2` *One step in backpropagation*

Description

One step in the backpropagation algorithm for a two hidden layers network

Usage

```
fitTeachNet2(data, weights, hidden.structure, learning.rate, f, f_d, decay, m_f, er)
```

Arguments

`data`	the data set
`weights`	current weights
`hidden.structure`	vector with first element the number of hidden neurons in the first hidden layer second element for the second hidden layer
`learning.rate`	rate by which factor for backpropagation gets smaller
`f`	activation function
`f_d`	derivative of activation function
`decay`	value of weight decay
`m_f`	interim value m
`er`	error function

Value

returns the new weight after gradient update

Author(s)

Georg Steinbuss

logistic *Logistic function*

Description

Computes the value of the logistic function

Usage

```
logistic(x)
```

Arguments

x	Input for logistic function

Author(s)

Georg Steinbuss

logistic.differential *Differential of logistic function*

Description

Computes value for the differential of the logistic function

Usage

```
logistic.differential(x)
```

Arguments

x	Input for differential of logistic function

Author(s)

Georg Steinbuss

predict.Weights *Computes prediction*

Description

This function computes for a given data set and weights of a one hidden layer network, a prediction from a TeachNet neural network.

Usage

```
## S3 method for class 'Weights'
predict(object, newdata, delete.firstColumn=TRUE, ...)
```

Arguments

object	The Weights object TeachNet returned after training.
newdata	The data set you which to predict. Has to have the same variables as the used training data set (except for the class variable) and has to be scaled (Z-Scores)!
delete.firstColumn	When class variable is first column, set to TRUE
...	additional arguments affecting the predictions produced

Value

returns a vector with the predictions of TeachNet

Author(s)

Georg Steinbuss

See Also

predict.Weights2, Weights-class, Weights2-class

predict.Weights2 *Computes prediction*

Description

This function computes for a given data set and weights of a two hidden layer network, a prediction from a TeachNet neural network.

Usage

```
## S3 method for class 'Weights2'
predict(object, newdata, delete.firstColumn=TRUE, ...)
```

Arguments

`object`	The Weights2 object TeachNet returned after training.
`newdata`	The data set you which to predict. Has to have the same variables as the used training data set (except for the class variable) and has to be scaled (Z-Scores)!
`delete.firstColumn`	When class variable is first column, set to TRUE
`...`	additional arguments affecting the predictions produced

Value

returns a vector with the predictions of TeachNet

Author(s)

Georg Steinbuss

See Also

predict.Weights, Weights-class, Weights2-class

`squaredError` *Computes squared error*

Description

Computes squared difference between two values

Usage

```
squaredError(x, y)
```

Arguments

`x`	value 1
`y`	value 2

Author(s)

Georg Steinbuss

sumCrossEntropy *Sums up cross entropy*

Description

Computes the full value of the cross entropy for TeachNet

Usage

```
sumCrossEntropy(weights, data, h2)
```

Arguments

weights	current weights
data	data frame
h2	number of neurons in second hidden layer

Author(s)

Georg Steinbuss

See Also

squaredError

sumSquaredError *Sums up squared error*

Description

Computes the full value of the squared error for TeachNet

Usage

```
sumSquaredError(weights, data, h2)
```

Arguments

weights	current weights
data	data frame
h2	number of neurons in second hidden layer

Author(s)

Georg Steinbuss

See Also

crossEntropy

`TeachNet` *Fits the neural network*

Description

The function TeachNet trains the neural network for a two class classification and also does some testing at the end. The class attribute os assumed to be the first column and coded as 1. Data gets scaled with Z-scores before training. It's also possible to get the final weights returned.

Usage

```
TeachNet(data, hidden.structure = -1, threshold.TotalError = 0.1, stepMax = 100,
learning.rate = 0.9, acc.fct = "logistic", err.fct = "sse", startWeights = NULL,
decay = 0, sameSample = FALSE, sampleLength = 0.7, all = FALSE, eval = TRUE)
```

Arguments

`data`	A data frame. The first column must be the class (0,1), the others the input variables (just numerical).
`hidden.structure`	The number of hidden neurons. A vector for two hidden layers. Default, -1 means that the automatic rule is applied (number of hidden neurons = number of variables divided by two, one hidden layer).
`threshold.TotalError`	Algorithm stops if total error falls below this threshold
`stepMax`	The maximum steps the algorithm does. One step is equal to one update of the weights (one cycle through the training set) for that he has to calculate the gradient of the total error.
`learning.rate`	Multiplicative factor by which the actual learning rate is iteratively reduced until the new error is smaller than the old one
`acc.fct`	The activation function that is used. In this version only "logistic" possible.
`err.fct`	The error function that is used. You can choose "sse" for sum squared error, or "ce" for cross entropy.
`startWeights`	This is where you can give TeachNet weights to start with.
`decay`	The factor for the weight decay.
`sameSample`	If TRUE the training and test data will be a data set with nearly same number of class 0 and 1. Randomly chosen out of the data.
`sampleLength`	Ratio that implies rows of the training data set depending on the full data set. Should be a number greater than 0 and less than 1. Test data set has size (1 - sampleLenght).
`all`	If TRUE training data is the whole dataset (test data is training data).
`eval`	If TRUE evaluation is computed.

Details

In the beginning the weights are initialized with a standard normal distribution. But this package is due to its very slow code just to understand the backpropagation algorithm. A good package for real training of neural networks is for example 'nnet'.

Value

TeachNet returns a S4 class object Weights for one hidden layer or Weights2 for two hidden layer. In addition if 'all' is FALSE, it prints an Evaluation. First part is the best found Threshold and V (V=True Positive - False Positive) for the prediction on the test Dataset. Then a confusion matrix and the accuracy of the model compared to the percentage of observation with class zero and one.

Author(s)

Georg Steinbuss

See Also

Weights-class, Weights2-class, predict.Weights predict.Weights2

Examples

```
df <- sample(c(rep(1,20),rep(0,20)))
income <- c(rnorm(40,mean=1000,sd=10))
debt <- rnorm(40,mean=0.5,sd=0.1)
data <- data.frame(df, income, debt)

weights <- TeachNet(data,sameSample=TRUE,sampleLength=0.9,stepMax=2)
```

---

transformPrediction *Transforms prediction*

Description

Transforms prediction from prediction to class

Usage

```
transformPrediction(pred, threshold)
```

Arguments

pred	Prediction
threshold	A threshold up to which a prediction is class 0 or 1. A value from 0 to 1

Author(s)

Georg Steinbuss

Weights-class *Weights objects*

Description

Contains the weights for a one hidden layer neural network in TeachNet the here cold "Arguments" are the slots in the S4 class Weights

Arguments

alpha	Intercept from output layer
alpha_h	Intercept from hidden layer
w_h	Weights from hidden layer to output layer
w_ih	Weights from input layer to hidden layer

Author(s)

Georg Steinbuss

See Also

Weights2-class

Examples

```
H <- 3 # number of neurons in hidden layer
I <- 6 # number of inputs

random_weights <- new("Weights", alpha = rnorm(1), alpha_h = rnorm(H), w_h = rnorm(H),
                      w_ih = matrix(nrow=I,ncol=H, data=rnorm(I*H)))
```

Weights2-class *Weights2 objects*

Description

Contains the weights for a two hidden layer neural network in TeachNet the here cold "Arguments" are the slots in the S4 class Weights2

Arguments

alpha	Intercept from output layer
alpha_1m	Intercept from hidden layer
alpha_2h	Intercept from second hidden layer
w_h	Weights from second hidden layer to output layer
q_mh	Weights from first hidden layer to second hidden layer
w_im	Weights from input layer to first hidden layer

Author(s)

Georg Steinbuss

See Also

Weights-class

Examples

```
M <- 3 # number of neurons in first hidden layer
H <- 3 # number of neurons in second hidden layer
I <- 6 # number of inputs

random_weights <- new("Weights2", alpha = rnorm(1), alpha_1m = rnorm(M), alpha_2h = rnorm(H),
                      w_h = rnorm(H), q_mh = matrix(nrow=M,ncol=H, data=rnorm(M*H)),
                      w_im = matrix(nrow=I,ncol=M, data=rnorm(I*M)))
```

Index