Package ‘deepnet’

October 13, 2022

Type Package

Title Deep Learning Toolkit in R

Version 0.2.1

Date 2014-03-20

Author Xiao Rong

Maintainer Xiao Rong <runxiao@gmail.com>

Description Implement some deep learning architectures and neural network algorithms, including BP, RBM, DBN, Deep autoencoder and so on.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-24 12:29:27 UTC

R topics documented:

dbn.dnn.train ......................................................... 2
load.mnist .............................................................. 3
nn.predict ............................................................... 3
nn.test ................................................................. 4
nn.train ................................................................. 5
rbm.down ................................................................. 6
rbm.train ............................................................... 7
rbm.up ................................................................. 8
sae.dnn.train ......................................................... 8

Index 10
Training a Deep neural network with weights initialized by DBN

Usage

dbn.dnn.train(x, y, hidden = c(1), activationfun = "sigm", learningrate = 0.8, momentum = 0.5, learningrate_scale = 1, output = "sigm", numepochs = 3, batchsize = 100, hidden_dropout = 0, visible_dropout = 0, cd = 1)

Arguments

x
matrix of x values for examples

y
vector or matrix of target values for examples

hidden
vector for number of units of hidden layers. Default is c(10).

activationfun
activation function of hidden unit. Can be "sigm", "linear" or "tanh". Default is "sigm" for logistic function

learningrate
learning rate for gradient descent. Default is 0.8.

momentum
momentum for gradient descent. Default is 0.5.

learningrate_scale
learning rate will be multiplied by this scale after every iteration. Default is 1.

numepochs
number of iteration for samples. Default is 3.

batchsize
size of mini-batch. Default is 100.

output
function of output unit, can be "sigm", "linear" or "softmax". Default is "sigm".

hidden_dropout
drop out fraction for hidden layer. Default is 0.

visible_dropout
drop out fraction for input layer. Default is 0.

cd
number of iteration for Gibbs sample of CD algorithm.

Examples

Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
x <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
y <- c(rep(1, 50), rep(0, 50))
dnn <- dbn.dnn.train(x, y, hidden = c(5, 5))
## predict by dnn
test_Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
test_Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
test_x <- matrix(c(test_Var1, test_Var2), nrow = 100, ncol = 2)
nn.test(dnn, test_x, y)
Value

return raw output value of neural network. For classification task, return the probability of a class

Author(s)

Xiao Rong

Examples

Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
x <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
y <- c(rep(1, 50), rep(0, 50))
nn <- nn.train(x, y, hidden = c(5))
# predict by nn

test_Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
test_Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
test_x <- matrix(c(test_Var1, test_Var2), nrow = 100, ncol = 2)
yny <- nn.predict(nn, test_x)

nn.test(nn, x, y, t = 0.5)

Arguments

nn
new samples to predict
y
new samples’ label
t
threshold for classification. If nn.predict value >= t then label 1, else label 0

Value

error rate

Author(s)

Xiao Rong
**Examples**

```
Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
x <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
y <- c(rep(1, 50), rep(0, 50))
nn <- nn.train(x, y, hidden = c(5))
test_Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
test_Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
test_x <- matrix(c(test_Var1, test_Var2), nrow = 100, ncol = 2)
err <- nn.test(nn, test_x, y)
```

---

**nn.train**

*Training Neural Network*

**Description**

Training single or multiple hidden layers neural network by BP.

**Usage**

```r
nn.train(x, y, initW = NULL, initB = NULL, hidden = c(10), activationfun = "sigm",
         learningrate = 0.8, momentum = 0.5, learningrate_scale = 1, output = "sigm",
         numepochs = 3, batchsize = 100, hidden_dropout = 0, visible_dropout = 0)
```

**Arguments**

- `x`: matrix of x values for examples
- `y`: vector or matrix of target values for examples
- `initW`: initial weights. If missing chosen at random
- `initB`: initial bias. If missing chosen at random
- `hidden`: vector for number of units of hidden layers. Default is c(10).
- `activationfun`: activation function of hidden unit. Can be "sigm", "linear" or "tanh". Default is "sigm" for logistic function
- `learningrate`: learning rate for gradient descent. Default is 0.8.
- `momentum`: momentum for gradient descent. Default is 0.5.
- `learningrate_scale`: learning rate will be multiplied by this scale after every iteration. Default is 1.
- `numepochs`: number of iteration for samples. Default is 3.
- `batchsize`: size of mini-batch. Default is 100.
- `output`: function of output unit, can be "sigm", "linear" or "softmax". Default is "sigm".
- `hidden_dropout`: drop out fraction for hidden layer. Default is 0.
- `visible_dropout`: drop out fraction for input layer. Default is 0.
Author(s)

Xiao Rong

Examples

Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
x <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
y <- c(rep(1, 50), rep(0, 50))
nn <- nn.train(x, y, hidden = c(5))

rbm.down
Generate visible vector by hidden units states

Description

Generate visible vector by hidden units states

Usage

rbm.down(rbm, h)

Arguments

rbm an rbm object trained by function train.rbm
h hidden units states

Value

generated visible vector

Author(s)

Xiao Rong

Examples

Var1 <- c(rep(1, 50), rep(0, 50))
Var2 <- c(rep(0, 50), rep(1, 50))
x3 <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
r1 <- rbm.train(x3, 3, numepochs = 20, cd = 10)
h <- c(0.2, 0.8, 0.1)
v <- rbm.down(r1, h)
rbm.train

Training a RBM (restricted Boltzmann Machine)

Description

Training a RBM (restricted Boltzmann Machine)

Usage

rbm.train(x, hidden, numepochs = 3, batchsize = 100, learningrate = 0.8,
learningrate_scale = 1, momentum = 0.5, visible_type = "bin", hidden_type = "bin",
cd = 1)

Arguments

x matrix of x values for examples
hidden number of hidden units
visible_type activation function of input unit. Only support "sigm" now
hidden_type activation function of hidden unit. Only support "sigm" now
learningrate learning rate for gradient descent. Default is 0.8.
momentum momentum for gradient descent. Default is 0.5.
learningrate_scale learning rate will be multiplied by this scale after every iteration. Default is 1.
numepochs number of iteration for samples Default is 3.
batchsize size of mini-batch. Default is 100.
cd number of iteration for Gibbs sample of CD algorithm.

Author(s)

Xiao Rong

Examples

Var1 <- c(rep(1, 50), rep(0, 50))
Var2 <- c(rep(0, 50), rep(1, 50))
x3 <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
r1 <- rbm.train(x3, 10, numepochs = 20, cd = 10)
rbm.up

*Infer hidden units state by visible units*

**Description**

Infer hidden units states by visible units

**Usage**

```r
rbm.up(rbm, v)
```

**Arguments**

- **rbm**: an rbm object trained by function `train.rbm`
- **v**: visible units states

**Value**

hidden units states

**Author(s)**

Xiao Rong

**Examples**

```r
Var1 <- c(rep(1, 50), rep(0, 50))
Var2 <- c(rep(0, 50), rep(1, 50))
x3 <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
r1 <- rbm.train(x3, 3, numepochs = 20, cd = 10)
v <- c(0.2, 0.8)
h <- rbm.up(r1, v)
```

sae.dnn.train

*Training a Deep neural network with weights initialized by Stacked AutoEncoder*

**Description**

Training a Deep neural network with weights initialized by Stacked AutoEncoder

**Usage**

```r
sae.dnn.train(x, y, hidden = c(1), activationfun = "sigm", learningrate = 0.8, momentum = 0.5, learningrate_scale = 1, output = "sigm", sae_output = "linear", numepochs = 3, batchsize = 100, hidden_dropout = 0, visible_dropout = 0)
```
Arguments

- **x**: matrix of x values for examples
- **y**: vector or matrix of target values for examples
- **hidden**: vector for number of units of hidden layers. Default is c(10).
- **activationfun**: activation function of hidden unit. Can be "sigm", "linear" or "tanh". Default is "sigm" for logistic function.
- **learningrate**: learning rate for gradient descent. Default is 0.8.
- **momentum**: momentum for gradient descent. Default is 0.5.
- **learningrate_scale**: learning rate will be multiplied by this scale after every iteration. Default is 1.
- **numepochs**: number of iteration for samples. Default is 3.
- **batchsize**: size of mini-batch. Default is 100.
- **output**: function of output unit, can be "sigm", "linear" or "softmax". Default is "sigm".
- **sae_output**: function of autoencoder output unit, can be "sigm", "linear" or "softmax". Default is "linear".
- **hidden_dropout**: drop out fraction for hidden layer. Default is 0.
- **visible_dropout**: drop out fraction for input layer. Default is 0.

Author(s)

Xiao Rong

Examples

```r
Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
x <- matrix(c(Var1, Var2), nrow = 100, ncol = 2)
y <- c(rep(1, 50), rep(0, 50))
dnn <- sae.dnn.train(x, y, hidden = c(5, 5))
## predict by dnn
test_Var1 <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))
test_Var2 <- c(rnorm(50, -0.8, 0.2), rnorm(50, 2, 1))
test_x <- matrix(c(test_Var1, test_Var2), nrow = 100, ncol = 2)
nn.test(dnn, test_x, y)
```
Index

dbn.dnn.train, 2
load.mnist, 3
nn.predict, 3
nn.test, 4
nn.train, 5
rbm.down, 6
rbm.train, 7
rbm.up, 8
sae.dnn.train, 8