Package ‘deepnet’

February 19, 2015

Type Package
Title deep learning toolkit in R
Version 0.2
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 2014-03-20 10:03:43

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
dbn.dnn.train (Training a Deep neural network with weights initialized by DBN)

Description

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.

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 <- 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)

load.mnist <- c(rnorm(50, 1, 0.5), rnorm(50, -0.6, 0.2))

load.mnist <- Load MNIST Data Set

Description

Load MNIST Data Set

Usage

load.mnist(dir)

Arguments

dir | dir of mnist dataset

Value

mnist dataset train$n number of train samples train$x pix of every train sample image train$y label of every train sample image train$yy one-of-c vector of label of train sample image test$n number of test samples test$x pix of every test sample image test$y label of every test sample image test$yy one-of-c vector of label of test sample image

Author(s)

Xiao Rong

nn.predict <- Predict new samples by Trained NN

Description

Predict new samples by Trained NN

Usage

nn.predict(nn, x)

Arguments

nn | neural network trained by function nn.train
x | new samples to predict
Value

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

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))
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)
yy <- nn.predict(nn, test_x)
```

----

**nn.test**

Test new samples by Trained NN

Description

Test new samples by Trained NN, return error rate for classification

Usage

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

Arguments

- **nn**
  - neural network trained by function nn.train
- **x**
  - 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**

```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))
nn <- nnNtrain(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)
ero <- nnNtest(nn, test_x, y)
```

---

**nn.train**  
*Training Neural Network*

**Description**

Training single or mutiple hidden layers neural network by BP

**Usage**

```r
nnNtrain(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

```r
var1 <- c(rnorm(U0L 1L 0NU)L rnorm(U0L -0NVL 0N2))
var2 <- c(rnorm(U0L -0N8L 0N2)L rnorm(U0L 2L 1))
x <- matrix(c(var1L var2)L nrow = 100L ncol = 2)
y <- c(rep(1L U0)L rep(0L U0))
nn <- nnNtrain(xL yL hidden = c(U))
```

---

rbm.down

Generate visible vector by hidden units states

**Description**

Generate visible vector by hidden units states

**Usage**

```r
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**

```r
Var1 <- c(rep(1L 50), rep(0L 50))
Var2 <- c(rep(0L 50), rep(1L 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)
```
Training a RBM(restricted Boltzmann Machine)

Usage

\begin{verbatim}
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)
\end{verbatim}

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

\begin{verbatim}
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)
\end{verbatim}
Description
Infer hidden units states by visible units

Usage
\[ \text{rbm.up}(\text{rbm}, \text{v}) \]

Arguments
- \text{rbm}: an rbm object trained by function train.rbm
- \text{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)
```

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

Usage
\[ \text{sae.dnn.train}(x, y, \text{hidden} = \text{c}(1), \text{activationfun} = \text{"sigm"}, \text{learningrate} = 0.8, \text{momentum} = 0.5, \text{learningrate_scale} = 1, \text{output} = \text{"sigm"}, \text{sae_output} = \text{"linear"}, \text{numepochs} = 3, \text{batchsize} = 100, \text{hidden_dropout} = 0, \text{visible_dropout} = 0) \]
sae.dnn.train

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