Package ‘DeepGS’

February 15, 2017

Title Genomic Selection Using the Deep Learning Technique

Version 1.0

Author Chuang Ma, Zhixu Qiu, Qian Cheng and Wenlong Ma

Maintainer Zhixu Qiu <zhixuqiu2015@gmail.com>, Chuang Ma <chuangma2006@gmail.com>

Description
The R package 'DeepGS' can be used to perform genomic selection (GS), which is a promising breeding strategy in plants and animals. DeepGS predicts phenotypes using genome-wide genotypic markers with an advanced machine learning technique (deep learning). The effectiveness of DeepGS has been demonstrated in predicting eight phenotypic traits on a population of 2000 Iranian bread wheat (Triticum aestivum) lines from the wheat gene bank of the International Maize and Wheat Improvement Center (CIMMYT).

Depends R (>= 3.3.1), mxnet (>= 0.6)

License GPL-2|GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

R topics documented:

cvSampleIndex *Generate Sample Indices for Training Sets and Testing Sets*

Description

This function generates indices for samples in training and testing sets for performing the N-fold cross validation experiment.

Usage

```
cvSampleIndex(sampleNum, cross = 5, seed = 1, randomSeed = FALSE)
```

Arguments

`sampleNum`	The number of samples needed to be partitioned into training and testing sets.
`cross`	The fold of cross validation.
`seed`	An integer used as the seed for data partition. The default value is 1.
`randomSeed`	Logical variable, default FALSE.

Value

A list and each element including $trainIdx $testIdx and $cvIdx

$trainIdx The index of training samples.

$testIdx The index of testing samples.

$cvIdx The cross validation index.

Author(s)

Chuang Ma, Zhixu Qiu, Qian Cheng, Wenlong Ma

Examples

```
## Not run
## Load example data ##
data(wheat_example)
## 5-fold cross validation
b <- cvSampleIndex(sampleNum = 2000,cross = 5,seed = 1)

## End (Not run)
```

meanNDCG	*Calculate mean Normalied Doscounted Cumulative Gain*

Description

This function calculates the mean normalied doscounted cumulative gain(meanNDCG) value for evaluting the prediction performance of a genomic selection prediction model in selecting top k individuals with high breeding value.

Usage

```
meanNDCG(realScores, predScores, topK = 10)
```

Arguments

`realScores`	A numeric vector is the real breeding values of the validation individual for a trait.
`predScores`	A numeric vector or matrix is the prediction breeding value predicted by genomic selection model of the individuals.
`topK`	A numeric vector is the number of excellent individuals.

Author(s)

Chuang Ma, Zhixu Qiu, Qian Cheng, Wenlong Ma

Examples

```
## Not run
refer_value <- runif(100)
pred_value <- sin(refer_value) + cos(refer_value)
meanNDCG(realScores = refer_value,predScores = pred_value, topK = 10)
## End (Not run)
```

predict_GSModel	*predict_GSModel*

Description

Predict trait values using trained deep learning genomic selection prediction model.

Usage

```
predict_GSModel(GSModel, testMat, imageSize)
```

Arguments

`GSModel`	Trained prediction model obtained from the DeepGSModel function.
`testMat`	A genotype matrix(T * M; T individuals, M markers)
`imageSize`	(String)this gives a "i * j" image format that the (M x1)markers informations of each individual will be encoded.

Author(s)

Chuang Ma , Qian Cheng, Zhixu Qiu,Wenlong Ma

`train_GSModel` *Build a genomic selection prediction model using the deep learning technique*

Description

The function applies the deep convolutional neural network to build a prediction model for genomic selection

Usage

```
train_GSModel(trainMat, trainPheno, imageSize, cnnFrame, device_type,
  gpuNum = "max", eval_metric = "mae", num_round = 10,
  array_batch_size = 128, learning_rate = 0.01, momentum = 0.9, wd = 0,
  randomseeds = NULL, initializer_idx = 0.01, ...)
```

Arguments

`trainMat` A genotype matrix(N x M; N individuals,M markers)

`trainPheno` Vector (N * 1) of phenotype.

`imageSize` (String)this gives a "i * j" image format that the (M x1)markers informations of each individual will be encoded.

`cnnFrame` A list containing the following element for convolutional neural network (CNN) framework:

- conv_kernel: A vector(K * 1) gives convolutional kernel sizes(width x height) to filter image matrix for K convolutional layers,respectively.
- conv_num_filter: A vector(K * 1) gives number of convolutional kernels for K convolutional layers,respectively.
- pool_act_type: A vector(K * 1) gives types of active function will define outputs of K convolutional layers which will be an input of corresponding pool layer, respectively.It include "relu","sigmoid","softrelu" and "tanh".
- pool_type: Character, K * 1) types of K pooling layers select from "avg", "max", "sum",respectively.
- pool_kernel: (Character, K * 1) K pooling kernel sizes(width x height) for K pooling layers.

- pool_stride: (Character, K * 1)strides for K pooling kernels.
- fullayer_num_hidden: Numeric, H * 1) number of hidden neurons for H full connected layers, respectively. the last full connected layer's number of hidden nerurons must is one.
- fullayer_act_type: Numeric, (H-1) * 1) selecting types of active function from "relu", "sigmoid", "softrelu" and "tanh" for full connected layers.

`device_type`	Selecting "cpu" or "gpu" device to construct predict model.
`gpuNum`	(Integer) number of GPU devices, if using multiple GPU(gpuNum > 1), the parameter momentum must greater than 0.
`eval_metric`	(String) A approach for evaluating the performance of training process, it include "mae", "rmse" and "accuracy", default "mae".
`num_round`	(Integer) The number of iterations over training data to train the model,default=10.
`array_batch_size`	(Integer) it defines number of samples that going to be propagated through the network for each update weight,default 128.
`learning_rate`	The learn rate for training process.
`momentum`	(Float,0~1) Momentum for moving average, default 0.9.
`wd`	(Float,0~1) weight decay,default 0.
`randomseeds`	set the seed used by mxnet device-specific random number. generators
`initializer_idx`	The initialization scheme for parameters.
`...`	Parameters for construncting neural networks used in `mxnet`.

Author(s)

Chuang Ma , Zhixu Qiu, Qian Cheng, Wenlong Ma

Examples

```
data(wheat_example)
Markers <- wheat_example$Markers
y <- wheat_example$y
cvSampleList <- cvSampleIndex(length(y),10,1)
#### cross validation set
cvIdx <- 1
trainIdx <- cvSampleList[[cvIdx]]$trainIdx
testIdx <- cvSampleList[[cvIdx]]$testIdx
# ########################### set DeepGS paramater
conv_kernel <- c("4*4","5*5") ## convolution kernels (fileter shape)
conv_num_filter <- c(20,28)  ## number of filters
pool_act_type <- c("relu","relu") ## active function for next pool
pool_type <- c("max","max") ## Max pooling shape
pool_kernel <- c("2*2","2*2") ## pooling shape
pool_stride <- c("2*2","2*2") ## number of pool kernerls
fullayer_num_hidden <- c(56,1)
fullayer_act_type <- c("sigmoid")
cnnFrame <- list(conv_kernel =conv_kernel,conv_num_filter = conv_num_filter,
              pool_act_type = pool_act_type,pool_type = pool_type,pool_kernel =pool_kernel,
                 pool_stride = pool_stride,fullayer_num_hidden= fullayer_num_hidden,
                 fullayer_act_type = fullayer_act_type)

trainGSmodel <- train_GSModel(trainMat = Markers[trainIdx,],trainPheno = y[trainIdx],
                           imageSize = "35*35", cnnFrame = cnnFrame,device_type = "cpu",
                            gpuNum = 1, eval_metric = "mae", num_round = 30,
                            array_batch_size= 100,learning_rate = 0.01,  momentum = 0,
                            wd = 0, randomseeds = 0,initializer_idx = 0.01)
predscores <- predict_GSModel(GSModel = trainGSmodel,testMat = Markers[testIdx,],
                            imageSize = "35*35" )
```

`wheat_example` *Run a examples for an in-development function.*

Description

A list,involve:

- Marker: A matrix(599 * 1225), each row represent 1225 markers information for one individuals.
- y: The real phenotype value for each individual.

Usage

```
data(wheat_example)
```

Index