DRNGS Manual

A New Method for Genomic Selection Under Deep Residual Network.

Version: 0.1.0

Date: November, 14th 2021

Author: Linlin Gu and Ming Fang

Maintainer:

Linlin Gu: gulinlin_141006@163.com

Prof. Ming Fang: fangming618@126.com

Table of Contents

1 Brief introduction

Genomic selection (GS) is a new method of selective breeding using single nucleotide polymorphism (SNP) markers throughout the whole genome. At present, there have been many GS models, but the conventional GS models are linear statistical models, which cannot capture the complex relationships among the genotypes, which limits the prediction accuracy of the models. Therefore, we developed a new deep learning GS method, named DRNGS. The characteristics of the new method are as follows :(1) deep residual network is used to predict GEBV, which can capture the complex relationships within genotypes and improve the prediction accuracy; (2) The strategies of convolution and pooling are adopted to reduce the complexity of high-dimensional genotype data and accelerate the computation speed; (3) The batch normalization layer (BN) is introduced into the model, which speeds up the convergence rate of the model. We provided a new solution to the problems of slow convergence speed and long computation time in deep learning.

2 DRNGS function

DRNGS	Build a genomic selection prediction model using the deep residual network.

2.1 Description

The DRNGS function is deep learning for prediction of genetic values with deep residual network.

2.2 Usage

```
DRNGS(trainMat = trainMat, trainPheno = trainPheno, validMat = validMat,
        validPheno = validPheno, markerImage = markerImage,
        ResFrame = ResFrame, device_type = "gpu", gpuNum = 0,
        eval_metric = "mae", num_round = 100, array_batch_size= 30,
        learning_rate = 0.01, momentum = 0.5, wd = 0.00001,
        randomseeds = 0, initializer_idx = 0.01, verbose = TRUE)
```

2.3 Arguments

trainMat	A genotype matrix (N x M; N individuals, M markers) for training model.
trainPheno	Vector (N * 1) of phenotype for training model.
validMat	A genotype matrix for valiting trained model.
validPheno	Vector (N * 1) of phenotype for valiting trained model.
markerImage	(String) This gives a "i * j" image format that the (M×1) markers informations of each individual will be encoded. If the image size exceeds the original SNP number, 0 will be polished the lack part. If the image size is less than the original SNP number, the last SNP(s) will be descaled.
ResFrame	A list containing the following element for residual neural network framework:

- Res_kernel: A vector (K * 1) gives convolutional kernel sizes (width × height) to filter image matrix for K convolutional layers, respectively.
- Res_num_filter: A vector (K * 1) gives number of convolutional kernels for K convolutional layers, respectively.
- Res_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".
- Res_stride: A character (K * 1) strides for K convolutional kernel.
- Respool_type: A character (K * 1) types of K pooling layers select from "avg", "max", "sum", respectively.
- Respool_kernel: A character (K * 1) K pooling kernel sizes (width * height) for K pooling layers.
- Respool_stride: A Character (K * 1) strides for K pooling kernels.
- fullayer_num_hidden: A 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: A numeric ((H-1) * 1) selecting types of active function from "relu", "sigmoid", "softrelu" and "tanh" for full connected layers.

	● drop_float: Numeric.
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 = 100.
array_batch_size	(Integer) It defines number of samples that going to be propagated through the network for each update weight, default 30.
learning_rate	The learn rate for training process, default 0.01.
momentum	(Float, 0~1) Momentum for moving average, default 0.5.
wd	(Float, 0~1) Weight decay, default 0.00001.
randomseeds	Set the seed used by mxnet device-specific random number, default 0.
initializer_idx	The initialization scheme for parameters, default 0.01.
verbose	logical (default=TRUE) Specifies whether to print information on the iterations during training.

3 predict.DRNGS function

predict.DRNGS	Predicting phenotypes from genotypes using DRNGS.

3.1 Description

Predict trait values using trained DRNGS prediction model.

3.2 Usage

predict.DRNGS(train_model, testMat, markerImage)

3.3 Arguments

train_model	Trained prediction model obtained from the DRNGS function.
testMat	A genotype matrix (T * M; T individuals, M markers).

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

4 cvSampleIndex function

cvSampleIndex	Generate Sample Indices for Training Sets and Testing Sets.

4.1 Description

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

4.2 Usage

cvSampleIndex(sampleNum, cross = 10, seed = 1, randomSeed = FALSE)

4.3 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. The default value is FALSE.

5 Build in data

An example datasets ‘wheat_example’ is a list that including the wheat yield datasets. The list including Markers and y. The Markers is a matrix (599 * 1225), each row represent 1,225 markers information for one individuals. The y is the real phenotype value for each individual. The ‘wheat_example’ can be loaded with data(wheat_example).

```
Data
wheat_example          Large list (2 elements, 5.7 Mb)
  Markers: num [1:599, 1:1225] 0 1 1 0 0 1 1 1 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:1225] "wPt.0538" "wPt.8463" "wPt.6348" "wPt.9992" .
  y : Named num [1:599] 1.672 -0.253 0.342 0.785 0.998 ...
  ..- attr(*, "names")= chr [1:599] "775" "2166" "2167" "2465" ...
```

5.1 Running build-in data

```
library("DRNGS")
data(wheat_example)
Markers <- wheat_example$Markers
y <- wheat_example$y
cvSampleList <- cvSampleIndex(length(y),10,1)
cvIdx <- 1
trainIdx <- cvSampleList[[cvIdx]]$trainIdx
testIdx <- cvSampleList[[cvIdx]]$testIdx
trainMat <-Markers[trainIdx,]
trainPheno <- y[trainIdx]
validIdx <- sample(1:length(trainIdx),floor(length(trainIdx)*0.1))
validMat <- trainMat[validIdx,]
validPheno <- trainPheno[validIdx]
trainMat <- trainMat[-validIdx,]
trainPheno <- trainPheno[-validIdx]
testMat <- Markers[testIdx,]
testPheno <- y[testIdx]
Res_kernel <- c("1*18")
Res_stride <- c("1*1")
Res_num_filter <- c(8)
Respool_type <- c("max")
Respool_kernel <- c("1*4")
Respool_stride <- c("1*4")
Res_layer_num <- c(1)
Res_act_type <- c("relu")
block_kernel <- c("1*17")
block_stride <- c("1*1")
block_pad <- c("0*8")
block_num_filter <- c(8)
block_act_type <- c("relu")
fullayer_num_hidden <- c(32,1)
fullayer_act_type <- c("relu")
drop_float <- c(0.2,0.1,0.05)
ResFrame <- list(Res_kernel=Res_kernel, Res_stride=Res_stride,
                 Res_num_filter=Res_num_filter,Res_act_type=Res_act_type,
                 block_kernel=block_kernel,block_stride=block_stride,
                 block_pad=block_pad,block_num_filter=block_num_filter,
                 block_act_type=block_act_type,Respool_type=Respool_type,
                 Respool_kernel=Respool_kernel,Respool_stride=Respool_stride,
                 fullayer_num_hidden=fullayer_num_hidden,
                 fullayer_act_type=fullayer_act_type,drop_float=drop_float,
                 Res_layer_num=Res_layer_num)
```

```
markerImage = paste0("1*",ncol(trainMat))
train_model<-DRNGS(trainMat = trainMat, trainPheno = trainPheno, validMat = validMat,
                   validPheno = validPheno, markerImage = markerImage, ResFrame =
                   ResFrame,device_type = "gpu", gpuNum = 0, eval_metric =
                   "mae",num_round = 100,array_batch_size= 30, learning_rate =
                   0.01,momentum = 0.5,wd = 0.00001, randomseeds = 0, initializer_idx =
                   0.01,verbose = TRUE)
Predict_value<-predict.DRNGS(GSModel = train_model, testMat = testMat,
                             markerImage = markerImage )
```

6 Code availability

The source code of DRNGS is freely available.

7 How to access help

If users have any bugs or issues or any suggestions are available, feel free to contact:
Linlin Gu: gulinlin_141006@163.com
Prof. Ming Fang: fangming618@126.com