Package ‘deepdive’

March 26, 2020

Type Package
Title Deep Learning for General Purpose
Version 1.0.1
Author Rajesh Balakrishnan
Maintainer Rajesh Balakrishnan <rajeshbalakrishnan24@gmail.com>

Description Aims to provide simple intuitive functions to create quick prototypes of artificial neural network or deep learning models. In addition novel ensemble models like 'deeptree' and 'deepforest' has been included which combines decision trees and neural network.

License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0

R topics documented:

deepforest ....................................................... 2
depenet ......................................................... 5
deeptree ........................................................... 7
predict.deepforest ............................................. 10
predict.deepnet ............................................... 11
predict.deeptree ............................................... 11
variableImportance .......................................... 12

Index 13
**Description**

Build or train bagged deeptree or deepnet of multiple architecture. Based on error choice either select best model or average multiple model with random variable cut, data cut and architecture.

**Usage**

```r
deepforest(
x, y,
  networkCount = 3,
  layerChoice = c(2:3),
  unitsChoice = c(4:10),
  cutVarSizePercent = 0.6,
  cutDataSizePercent = 0.6,
  activation = c("sigmoid", "sigmoid"),
  reluLeak = 0,
  modelType = "regress",
  iterations = 500,
  eta = 10^-2,
  seed = 2,
  gradientClip = 0.8,
  regularisePar = 0,
  optimiser = "adam",
  parMomentum = 0.9,
  inputSizeImpact = 1,
  parRmsPropZeroAdjust = 10^-8,
  parRmsProp = 0.9999,
  treeLeaves = NA,
  treeMinSplitPercent = 0.3,
  treeMinSplitCount = 100,
  treeCp = 0.01,
  errorCover = 0.2,
  treeAugment = TRUE,
  printItrSize = 100,
  showProgress = TRUE,
  stopError = 0.01,
  miniBatchSize = NA,
  useBatchProgress = TRUE
)
```

**Arguments**

- `x` a data frame with input variables
y  a data frame with output variable
networkCount  Integer, Number of deepnet or deeptree to build
layerChoice  vector, different layer choices
unitsChoice  vector, number of units choice
cutVarSizePercent  ratio, percentage of variable to for each network
cutDataSizePercent  ratio, percentage of data to for each network
activation  choose from "sigmoid","relu","sin","cos","none". Activations will be randomly chosen from chosen. Default is relu and sin
reluLeak  numeric. Applicable when activation is "relu". Specify value between 0 any number close to zero below 1. Eg: 0.01,0.001 etc
modelType  one of "regress","binary","multiClass". "regress" for regression will create a linear single unit output layer. "binary" will create a single unit sigmoid activated layer. "multiClass" will create layer with units corresponding to number of output classes with softmax activation.
iterations  integer. This indicates number of iterations or epochs in backpropagation. The default value is 500.
eta  numeric. Hyperparameter, sets the Learning rate for backpropagation. Eta determines the convergence ability and speed of convergence.
seed  numeric. Set seed with this parameter. Incase of sin activation sometimes changing seed can yield better results. Default is 2
gradientClip  numeric. Hyperparameter numeric value which limits gradient size for weight update operation in backpropagation. Default is 0.8. It can take any positive value.
regularisePar  numeric. L2 Regularisation Parameter.
optimiser  one of "gradientDescent","momentum","rmsProp","adam". Default value "adam"
parMomentum  numeric. Applicable for optimiser "momentum" and "adam"
inputSizeImpact  numeric. Adjusts the gradient size by factor of percentage of rows in input. For very small data set setting this to 0 could yeild faster result. Default is 1.
parRmsPropZeroAdjust  numeric. Applicable for optimiser "rmsProp" and "adam"
parRmsProp  numeric. Applicable for optimiser "rmsProp" and "adam"
treeLeaves  vector. Optional, leaves numbers from externally trained tree model can be supplied here. If supplied then model will not build a explicit tree and just fit a neural network to mentioned leaves.
treeMinSplitPercent  numeric. This parameter controls depth of tree setting min split count for leaf subdivision as percentage of observations. Final minimum split will be chosen as max of count calculated with treeMinSplitPercent and treeMinSplitCount. Default 0.3. Range 0 to 1.
treeMinSplitCount
numeric. This parameter controls depth of tree setting min split count. Final minimum split will be chosen as max of count calculated with treeMinSplitPercent and treeMinSplitCount. Default 30

treeCp
complexity parameter. \texttt{rpart.control}

errorCover
Ratio. Default is 0.2 i.e all models within 20 percent error of best model will be selected.

treeAugment
logical. If True fits deeptree and if False fits deepnet. Default is T

printItrSize
numeric. Number of iterations after which progress message should be shown. Default value 100 and for iterations below 100 at least 5 messages will be seen

showProgress
logical. True will show progress and F will not show progress

stopError
Numeric. Rmse at which iterations can be stopped. Default is 0.01, can be set as NA in case all iterations needs to run.

miniBatchSize
integer. Set the mini batch size for mini batch gradient

useBatchProgress
logical. Applicable for miniBatch, setting T will use show rmse in Batch and F will show error on full dataset. For large dataset set T

Value
returns model object which can be passed into \texttt{predict.deepforest}

Examples

\begin{verbatim}
require(deepdive)

x<-data.frame(x1=runif(10),x2=runif(10))
y<-data.frame(y=10*x$x1+20*x$x2+20)

mdeepf<-deepforest(x,y,
    networkCount=2,
    layerChoice=c(2:3),
    unitsChoice=c(4:10),
    cutVarSizePercent=0.6,
    cutDataSizePercent=0.6,
    activation = c("relu","sin"),
    reluLeak=0.01,
    modelType = "regress",
    iterations = 10,
    eta = 10 ^-2,
    seed=2,
    gradientClip=0.8,
    regularisePar=0,
    optimiser="adam",
    parMomentum=0.9,
    inputSizeImpact=1,
    parRmsPropZeroAdjust=10^-8,
    parRmsProp=0.9999,
)
\end{verbatim}
**deepnet**

```r
deepnet( 
  x, 
  y, 
  hiddenLayerUnits = c(2, 2), 
  activation = c("sigmoid", "relu"), 
  reluLeak = 0, 
  modelType = c("regress"), 
  iterations = 500, 
  eta = 10^-2, 
  seed = 2, 
  gradientClip = 0.8, 
  regularisePar = 0, 
  optimiser = "adam", 
  parMomentum = 0.9, 
  inputSizeImpact = 1, 
  parRmsPropZeroAdjust = 10^-8, 
  parRmsProp = 0.9999, 
  printItrSize = 100, 
  showProgress = TRUE, 
  stopError = 0.01, 
  miniBatchSize = NA, 
  useBatchProgress = FALSE, 
  ignoreNAerror = FALSE, 
  normalise = TRUE 
)
```

**Description**

Build and train an Artificial Neural Network of any depth in a single line code. Choose the hyperparameters to improve the accuracy or generalisation of the model.

**Usage**

deenet(
  x, 
  y, 
  hiddenLayerUnits = c(2, 2), 
  activation = c("sigmoid", "relu"), 
  reluLeak = 0, 
  modelType = c("regress"), 
  iterations = 500, 
  eta = 10^-2, 
  seed = 2, 
  gradientClip = 0.8, 
  regularisePar = 0, 
  optimiser = "adam", 
  parMomentum = 0.9, 
  inputSizeImpact = 1, 
  parRmsPropZeroAdjust = 10^-8, 
  parRmsProp = 0.9999, 
  printItrSize = 100, 
  showProgress = TRUE, 
  stopError = 0.01, 
  miniBatchSize = NA, 
  useBatchProgress = FALSE, 
  ignoreNAerror = FALSE, 
  normalise = TRUE 
)
Arguments

x a data frame with input variables

y a data frame with output variable

hiddenLayerUnits a numeric vector, length of vector indicates number of hidden layers and each element in vector indicates corresponding hidden units Eg: c(6,4) for two layers, one with 6 hidden units and other with 4 hidden units. Note: Output layer is automatically created.

activation one of "sigmoid","relu","sin","cos","none". The default is "sigmoid". Choose an activation per hidden layer.

reluLeak numeric. Applicable when activation is "relu". Specify value between 0 any number close to zero below 1. Eg: 0.01,0.001 etc

modelType one of "regress","binary","multiClass". "regress" for regression will create a linear single unit output layer. "binary" will create a single unit sigmoid activated layer. "multiClass" will create layer with units corresponding to number of output classes with softmax activation.

iterations integer. This indicates number of iterations or epochs in backpropagation. The default value is 500.

eta numeric. Hyperparameter, sets the learning rate for backpropagation. Eta determines the convergence ability and speed of convergence.

seed numeric. Set seed with this parameter. Incase of sin activation sometimes changing seed can yield better results. Default is 2

gradientClip numeric. Hyperparameter numeric value which limits gradient size for weight update operation in backpropagation. Default is 0.8. It can take any positive value.

regularisePar numeric. L2 Regularisation Parameter.

optimiser one of "gradientDescent","momentum","rmsProp","adam". Default value "adam"

parMomentum numeric. Applicable for optimiser "momentum" and "adam"

inputSizeImpact numeric. Adjusts the gradient size by factor of percentage of rows in input. For very small data set setting this to 0 could yield faster result. Default is 1.

parRmsPropZeroAdjust numeric. Applicable for optimiser "rmsProp" and "adam"

parRmsProp numeric. Applicable for optimiser "rmsProp" and "adam"

printItrSize numeric. Number of iterations after which progress message should be shown. Default value 100 and for iterations below 100 atleast 5 messages will be seen

showProgress logical. True will show progress and F will not show progress

stopError Numeric. Rmse at which iterations can be stopped. Default is 0.01, can be set as NA in case all iterations needs to run.

miniBatchSize integer. Set the mini batch size for mini batch gradient

useBatchProgress logical. Applicable for miniBatch, setting T will use show rmse in Batch and F will show error on full dataset. For large dataset set T

ignoreNAerror logical. Set T if iteration needs to be stopped when predictions become NA

normalise logical. Set F if normalisation not required. Default T
Value

returns model object which can be passed into predict.deepnet

Examples

```r
require(deepdive)

x <- data.frame(x1 = runif(10),x2 = runif(10))
y<- data.frame(y=20*x$x1 +30*x$x2+10)

#train
modelnet<-deepnet(x,y,c(2,2),
activation = c('relu','sigmoid'),
reluLeak = 0.01,
modelType = "regress",
iterations =5,
eta=0.8,
optimiser="adam")

#predict
predDeepNet<-predict.deepnet(modelnet,newData=x)

#evaluate
sqrt(mean((predDeepNet$ypred-y$y)^2))
```

---

deepnet  | Descition Tree augmented by Artificial Neural Network

Description

This models divides the input space by fitting a tree followed by artificial neural network to each of leaf. Decision tree model is built using rpart package and neural network using deepdive. Feature of stacking predictions from other models is also made available.

Usage

```r
deepnet(
  x, 
  y, 
  hiddenLayerUnits = c(2, 2), 
  activation = c("sigmoid", "sigmoid"),
  reluLeak = 0,
  modelType = "regress",
  iterations = 500,
  eta = 10^-2,
  seed = 2,
)```
gradientClip = 0.8,
regularisePar = 0,
optimiser = "adam",
parMomentum = 0.9,
inputSizeImpact = 1,
parRmsPropZeroAdjust = 10^-8,
parRmsProp = 0.9999,
treeLeaves = NA,
treeMinSplitPercent = 0.3,
treeMinSplitCount = 30,
treeCp = 0.01,
stackPred = NA,
printItrSize = 100,
showProgress = TRUE,
stopError = 0.01,
miniBatchSize = NA,
useBatchProgress = TRUE,
ignoreNAerror = FALSE
)

Arguments

x

A data frame with input variables

y

A data frame with output variable

hiddenLayerUnits

A numeric vector, length of vector indicates number of hidden layers and each element in vector indicates corresponding hidden units. Eg: c(6,4) for two layers, one with 6 hidden units and other with 4 hidden units. Note: Output layer is automatically created.

activation

One of "sigmoid","relu","sin","cos","none". The default is "sigmoid". Choose a activation per hidden layer

reluLeak

Numeric. Applicable when activation is "relu". Specify value between 0 any number close to zero below 1. Eg: 0.01,0.001 etc

modelType

One of "regress","binary","multiClass". "regress" for regression will create a linear single unit output layer. "binary" will create a single unit sigmoid activated layer. "multiClass" will create layer with units corresponding to number of output classes with softmax activation.

iterations

Integer. This indicates number of iterations or epochs in backpropagation. The default value is 500.

eta

Numeric. Hyperparameter, sets the Learning rate for backpropagation. Eta determines the convergence ability and speed of convergence.

seed

Numeric. Set seed with this parameter. In case of sin activation sometimes changing seed can yield better results. Default is 2

gradientClip

Numeric. Hyperparameter numeric value which limits gradient size for weight update operation in backpropagation. Default is 0.8. It can take any positive value.
regularisePar: numeric, L2 Regularisation Parameter.

optimiser: one of "gradientDescent", "momentum", "rmsProp", "adam". Default value: "adam"

parMomentum: numeric. Applicable for optimiser "momentum" and "adam"

inputSizeImpact: numeric. Adjusts the gradient size by factor of percentage of rows in input. For very small data set setting this to 0 could yield faster result. Default is 1.

parRmsPropZeroAdjust: numeric. Applicable for optimiser "rmsProp" and "adam"

parRmsProp: numeric. Applicable for optimiser "rmsProp" and "adam"

treeLeaves: vector. Optional, leaves numbers from externally trained tree model can be supplied here. If supplied then model will not build a explicit tree and just fit a neural network to mentioned leaves.

treeMinSplitPercent: numeric. This parameter controls depth of tree setting min split count for leaf subdivision as percentage of observations. Final minimum split will be chosen as max of count calculated with treeMinSplitPercent and treeMinSplitCount. Default 0.3. Range 0 to 1.

treeMinSplitCount: numeric. This parameter controls depth of tree setting min split count. Final minimum split will be chosen as max of count calculated with treeMinSplitPercent and treeMinSplitCount. Default 30.

treeCp: complexity parameter. rpart.control

stackPred: vector. Predictions from buildnet or other models can be supplied here. If for certain leaf stackPred accuracy is better then stackPred predictions will be chosen.

printItrSize: numeric. Number of iterations after which progress message should be shown. Default value 100 and for iterations below 100 atleast 5 messages will be seen.

showProgress: logical. True will show progress and F will not show progress.

stopError: Numeric. Rmse at which iterations can be stopped. Default is 0.01, can be set as NA in case all iterations need to run.

miniBatchSize: integer. Set the mini batch size for mini batch gradient.

useBatchProgress: logical. Applicable for miniBatch, setting T will use show rmse in Batch and F will show error on full dataset. For large dataset set T.

ignoreNAError: logical. Set T if iteration needs to be stopped when predictions become NA.

**Value**

returns model object which can be passed into predict.deeptree

**Examples**

require(deepdive)
x <- data.frame(x1 = runif(10), x2 = runif(10))

y <- data.frame(y = 20*x$x1 + 30*x$x2 + 10)

deepTreeMod <- deetree(x, y,
  hiddenLayerUnits = c(4, 4),
  activation = c("relu", "sin"),
  reluLeak = 0.01,
  modelType = 'regress',
  iterations = 1000,
  eta = 0.4,
  seed = 2,
  gradientClip = 0.8,
  regularisePar = 0,
  optimiser = "adam",
  parMomentum = 0.9,
  inputSizeImpact = 1,
  parRmsPropZeroAdjust = 10^-8,
  parRmsProp = 0.9999,
  treeLeaves = NA,
  treeMinSplitPercent = 0.4,
  treeMinSplitCount = 100,
  stackPred = NA,
  stopError = 4,
  miniBatchSize = 64,
  useBatchProgress = TRUE,
  ignoreNAerror = FALSE)

predict.deepforest Predict Function for DeepForest

Description

Predict Function for DeepForest

Usage

## S3 method for class 'deepforest'
predict(object, newData, ...)

Arguments

object deepforest model object
newData pass dataframe for prediction
... further arguments passed to or from other methods.
predict.deepnet  Predict Function for Deepnet

Description
Predict Function for Deepnet

Usage
```r
## S3 method for class 'deepnet'
predict(object, newData, ...)
```

Arguments
- `object`: deepnet model object
- `newData`: pass dataframe for prediction
- `...`: further arguments passed to or from other methods.

Value
returns predictions vector or dataframe

predict.deeptree  Predict Function for Deeptree

Description
Predict Function for Deeptree

Usage
```r
## S3 method for class 'deeptree'
predict(object, newData, treeLeaves = NA, stackPred = NA, ...)
```

Arguments
- `object`: deeptree model object
- `newData`: pass dataframe for prediction
- `treeLeaves`: Pass vector with tree leaves if fit outside deeptree. default NA.
- `stackPred`: Pass stackPred of prediction data if it was passed in deeptree
- `...`: further arguments passed to or from other methods.
**variableImportance**

**Value**

returns predictions vector or dataframe

---

**variableImportance**  
Variable importance for models in this library

---

**Description**

Variable importance for models in this library

**Usage**

```
variableImportance(model, x, y, showPlot = T, seed = 2)
```

**Arguments**

- **model**: Model object
- **x**: a data frame with input variables
- **y**: a data frame with output variable
- **showPlot**: logical. True will show importance plot. Default True
- **seed**: Set seed with this parameter. Incase of sin activation sometimes changing seed can yield better results. Default is 2

**Value**

returns variable importance data frame
Index

deeplearnpred, 2
deeps, 5
deeptree, 7

predict.deeplearnpred, 4, 10
predict.deeps, 7, 11
predict.deeptree, 9, 11

treecrit, 4, 9

variableImportance, 12