Package 'gradDescent'

January 25, 2018

Maintainer Lala Septem Riza <lala.s.riza@upi.edu>

Type Package

Title Gradient Descent for Regression Tasks

Version 3.0

URL https://github.com/drizzersilverberg/gradDescentR

Date 2018-01-03

Author Galih Praja Wijaya, Dendi Handian, Imam Fachmi Nasrulloh, Lala Septem Riza, Rani Megasari, Enjun Junaeti

Description An implementation of various learning algorithms based on Gradient Descent for dealing with regression tasks.
The variants of gradient descent algorithm are :
Mini-Batch Gradient Descent (MBGD), which is an optimization to use training data partially to reduce the computation load.
Stochastic Gradient Descent (SGD), which is an optimization to use a random data in learning to reduce the computation load drastically.
Stochastic Average Gradient (SAG), which is a SGD-based algorithm to minimize stochastic step to average.
Momentum Gradient Descent (MGD), which is an optimization to speed-up gradient descent learning.
Accelerated Gradient Descent (AGD), which is an optimization to accelerate gradient descent learning.
Adagrad, which is a gradient-descent-based algorithm that accumulate previous cost to do adaptive learning.
Adadelta, which is a gradient-descent-based algorithm that use hessian approximation to do adaptive learning.
RMSprop, which is a gradient-descent-based algorithm that combine Adagrad and Adadelta adaptive learning ability.
Adam, which is a gradient-descent-based algorithm that mean and variance moment to do adaptive learning.
Stochastic Variance Reduce Gradient (SVRG), which is an optimization SGD-based algorithm to accelerates the process toward converging by reducing the gradient.
Semi Stochastic Gradient Descent (SSGD),which is a SGD-based algorithm that combine GD and SGD to accelerates the process toward converging by choosing one of the gradients at a time.

Stochastic Recursive Gradient Algorithm (SARAH), which is an optimization algorithm similarly SVRG to accelerates the process toward converging by accumulated stochastic information. Stochastic Recursive Gradient Algorithm+ (SARAHPlus), which is a SARAH practical variant algorithm to accelerates the process toward converging provides a possibility of earlier termination.

License GPL (>= 2) | file LICENSE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-25 13:33:54 UTC

R topics documented:

ADADELTA *ADADELTA Method Learning Function*

Description

A function to build prediction model using ADADELTA method.

Usage

```
ADADELTA(dataTrain, maxIter = 10, momentum = 0.9, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
maxIter	the maximal number of iterations.
momentum	a float value represent momentum give a constant speed to learning process.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD with an optimization to create an adaptive learning rate by hessian approximation correction approach. Correction and has less computation load than ADAGRAD. This method create an exclusive learning rate and doesn't need alpha parameter, but uses momentum parameter same as MGD and AGD.

Value

a vector matrix of theta (coefficient) for linear model.

References

M. D. Zeiler Adadelta: An Adaptive Learning Rate Method, arXiv: 1212.5701v1, pp. 1-6 (2012)

See Also

ADAGRAD, RMSPROP, ADAM

Examples

```
##################################
## Learning and Build Model with ADADELTA
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with ADADELTA
ADADELTAmodel <- ADADELTA(splitedDataSet$dataTrain)
#show result
print(ADADELTAmodel)
```

ADAGRAD *ADAGRAD Method Learning Function*

Description

A function to build prediction model using ADAGRAD method.

Usage

```
ADAGRAD(dataTrain, alpha = 0.1, maxIter = 10, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD with an optimization to create an adaptive learning rate with an approach that accumulate previous cost in each iteration.

Value

a vector matrix of theta (coefficient) for linear model.

References

J. Duchi, E. Hazan, Y. Singer Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, Journal of Machine Learning Research 12, pp. 2121-2159 (2011)

See Also

ADADELTA, RMSPROP, ADAM

Examples

```
##################################
## Learning and Build Model with ADAGRAD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with ADAGRAD
ADAGRADmodel <- ADAGRAD(splitedDataSet$dataTrain)
#show result
print(ADAGRADmodel)
```

ADAM *ADADELTA Method Learning Function*

Description

A function to build prediction model using ADAM method.

Usage

```
ADAM(dataTrain, alpha = 0.1, maxIter = 10, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD with an optimization to create an adaptive learning rate by two moment estimation called mean and variance.

Value

a vector matrix of theta (coefficient) for linear model.

References

D.P Kingma, J. Lei Adam: a Method for Stochastic Optimization, International Conference on Learning Representation, pp. 1-13 (2015)

See Also

ADAGRAD, RMSPROP, ADADELTA

Examples

```
##################################
## Learning and Build Model with ADAM
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with ADAM
ADAMmodel <- ADAM(splitedDataSet$dataTrain)
#show result
print(ADAMmodel)
```

AGD *Accelerated Gradient Descent (AGD) Method Learning Function*

Description

A function to build prediction model using Accelerated Gradient Descent (AGD) method.

Usage

```
AGD(dataTrain, alpha = 0.1, maxIter = 10, momentum = 0.9, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
momentum	a float value represent momentum give a constant speed to learning process.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD and MGD with optimization to accelerate the learning with momentum constant in each iteration.

Value

a vector matrix of theta (coefficient) for linear model.

References

Y. Nesterov A method for unconstrained convex minimization problem with the rate of convergence O (1/k2), Soviet Mathematics Doklady 27 (2), pp. 543-547 (1983)

See Also

MGD

Examples

```
###################################
## Learning and Build Model with AGD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with AGD
AGDmodel <- AGD(splitedDataSet$dataTrain)
#show result
print(AGDmodel)
```

GD *Gradient Descent (GD) Method Learning Function*

Description

A function to build prediction model using Gradient Descent method.

Usage

```
GD(dataTrain, alpha = 0.1, maxIter = 10, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function build a prediction model using Gradient Descent (GD) method. Gradient Descent is a first order optimization algorithm to find a local minimum of an objective function by searching along the steepest descent direction. In machine learning, it is mostly used for dealing with supervised learning, which is regression task. By using GD, we construct a model represented in a linear equation that maps the relationship between input variables and the output one. In other words, GD determines suitable coefficient of each variables. So, that the equation can express the mapping correctly.

Value

a vector matrix of theta (coefficient) for linear model.

References

L.A. Cauchy, "Methode generale pour la resolution des systemes d equations", Compte Rendu a l Academie des Sciences 25, pp. 536-538 (1847)

See Also

MBGD

Examples

```
#################################
## Learning and Build Model with GD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with GD
GDmodel <- GD(splitedDataSet$dataTrain)
#show result
print(GDmodel)
```

`gradDescentR.learn` *GradDescent Learning Function*

Description

A top-level funtion to generate/learn the model from numerical data using a selected gradient descent method.

Usage

```
gradDescentR.learn(dataSet, featureScaling = TRUE,
  scalingMethod = "VARIANCE", learningMethod = "GD", control = list(),
  seed = NULL)
```

Arguments

`dataSet`	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
`featureScaling`	a boolean value that decide to do feature scaling on dataset. The default value is TRUE, which the function will do feature scaling.
`scalingMethod`	a string value that represent the feature scaling method to be used. There are two option for this arguments: `"VARIANCE"` and `"MINMAX"`. The default value is `"VARIANCE"`, which the function will do Variance Scaling/ Standardization to dataset.
`learningMethod`	a string value that represent the learning method to do model building. There are ten option for this arguments: `"GD"`, `"MBGD"`, `"SGD"`, `"SAGD"`, `"MGD"`, `"AGD"`, `"ADAGRAD"`, `"ADADELTA"`, `"RMSPROP"`, `"ADAM"`, `"SSGD"`, `"SVRG"`, `"SARAH"` and `"SARAHPlus"`. The default value is `"GD"`, which the function will to Gradient Descent learning.

`control` a list containing all arguments, depending on the learning algorithm to use. The following list are parameters required for each methods.

- `alpha`: a float value in interval of [0,1] that represent the step-size or learning rate of the learning. The default value is 0.1.
- `maxIter`: a integer value that represent the iteration/loop/epoch of the learning. The default value is 10, which the function will do 10 times learning calculation.

`seed` a integer value for static random. Default value is NULL, which the the function will not do static random.

Details

This function makes accessible all learning methods that are implemented in this package. All of the methods use this function as interface for the learning stage, so users do not need to call other functions in the learning phase. In order to obtain good results, users need to adjust some parameters such as the number of labels, the type of the shape of the membership function, the maximal number of iterations, the step size of the gradient descent, or other method-dependent parameters which are collected in the `control` parameter. After creating the model using this function, it can be used to predict new data with `predict`.

Value

The `gradDescentRObject`.

See Also

`predict`

Examples

```
##################################
## Learning and Build Model with GD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## train dataset
modelObject <- gradDescentR.learn(dataSet)
```

`gradDescentRData` *Data set of the package*

Description

This is a dataset that collected experimentaly by Kennedy in 1954 to obtain the density value of CO2. Parameter used in the experiment are temperature and pressure, which it can be a parameter to obtain compressibility factor value of CO2.

References

Kennedy, G. C. (1954). Pressure-Volmue Temperature Relations in CO2 at Elevated Temperatures and Pressures. American Journal of Science, 225-241.

`MBGD` *Mini-Batch Gradient Descent (MBGD) Method Learning Function*

Description

A function to build prediction model using Mini-Batch Gradient Descent (MBGD) method.

Usage

```
MBGD(dataTrain, alpha = 0.1, maxIter = 10, nBatch = 2, seed = NULL)
```

Arguments

`dataTrain`	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
`alpha`	a float value representing learning rate. Default value is 0.1
`maxIter`	the maximal number of iterations.
`nBatch`	a integer value representing the training data batch.
`seed`	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on `GD` method with optimization to use the training data partially. MBGD has a parameter named batchRate that represent the instances percentage of training data.

Value

a vector matrix of theta (coefficient) for linear model.

References

A. Cotter, O. Shamir, N. Srebro, K. Sridharan Better Mini-Batch Algoritms via Accelerated Gradient Methods, NIPS, pp. 1647- (2011)

See Also

`GD`

Examples

```
#################################
## Learning and Build Model with MBGD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with 0.8 batch rate MBGD
MBGDmodel <- MBGD(splitedDataSet$dataTrain, nBatch=2)
#show result
print(MBGDmodel)
```

MGD *Momentum Gradient Descent (MGD) Method Learning Function*

Description

A function to build prediction model using Momentum Gradient Descent (MGD) method.

Usage

```
MGD(dataTrain, alpha = 0.1, maxIter = 10, momentum = 0.9, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
momentum	a float value represent momentum give a constant speed to learning process.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD with an optimization to speed-up the learning by adding a constant momentum.

Value

a vector matrix of theta (coefficient) for linear model.

References

N. Qian On the momentum term in gradient descent learning algorithms., Neural networks : the official journal of the International Neural Network Society, pp. 145-151- (1999)

See Also

AGD

Examples

```
#################################
## Learning and Build Model with MGD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with MGD
MGDmodel <- MGD(splitedDataSet$dataTrain)
#show result
print(MGDmodel)
```

---

minmaxDescaling *Min-Max Scaling Revert Function*

Description

A function to revert the value that has been done by min-max scaling method.

Usage

```
minmaxDescaling(dataSet, minmaxParameter)
```

Arguments

dataSet	a data.frame that representing dataset ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataSet must have at least two columns and ten rows of data that contain only numbers (integer or float).
minmaxParameter	a matrix that has value of minmax scaling parameter, such as minimum value and maximum value of data that can be used to restore the original value of dataset. This parameter is exclusively produced by varianceScaling function.

Details

This function changes the value of min-max scaled dataset that produced by varianceScaling function and represented by data.frame object.

Value

a data.frame representing reverted dataset value

See Also

minmaxScaling

Examples

```
##################################
## Revert Min-Max Scaling
## load R Package data
data(gradDescentRData)
## get z-factor Data
dataSet <- gradDescentRData$CompressilbilityFactor
fsr <- minmaxScaling(dataSet)
rfsr <- minmaxDescaling(fsr$scaledDataSet, fsr$scalingParameter)
```

---

minmaxScaling *The Min-Max Feature Scaling Function*

Description

A function to do feature scaling to dataset with min-max scaling method.

Usage

```
minmaxScaling(dataSet)
```

Arguments

dataSet	a data.frame that representing dataset ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataSet must have at least two columns and ten rows of data that contain only numbers (integer or float).

Details

This function changes the value of dataset that represented by data.frame object into min-max scaled value that has interval between 0 to 1.

Value

a list contains feature scaled dataset and scaling parameter

See Also

minmaxDescaling

Examples

```
#################################
## Feature scaling with Min-Max Scaling Method
## load R Package data
data(gradDescentRData)
## get z-factor Data
dataSet <- gradDescentRData$CompressilbilityFactor
## do min-max scaling to dataset
featureScalingResult <- minmaxScaling(dataSet)
## show result
print(featureScalingResult$scaledDataSet)
print(featureScalingResult$scalingParameter)
```

---

predict.gradDescentRObject

The gradDescentR prediction stage

Description

This is the main function to obtain a final result as predicted values for all methods in this package. In order to get predicted values, this function is run using an `gradDescentRObject`, which is typically generated using `gradDescentR.learn`.

Usage

```
## S3 method for class 'gradDescentRObject'
predict(object, newdata, ...)
```

Arguments

`object`	an `gradDescentRObject`.
`newdata`	a data frame or matrix ($m \times n$) of data for the prediction process, where m is the number of instances and n is the number of input variables. It should be noted that the testing data must be expressed in numbers (numerical data).
`...`	the other parameters (not used)

Value

The predicted values.

See Also

gradDescentR.learn

Examples

```
###################################
## Predict NewData Using Model Object
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## train dataset
modelObject <- gradDescentR.learn(dataSet)
## create new data input
temp <- c(273.1, 353.1, 363.1)
pres <- c(24.675, 24.675, 24.675)
conf <- c(0.8066773, 0.9235751, 0.9325948)
zfac <- data.frame(temp, pres, conf)
## predict
prediction_data <- predict(modelObject, zfac)


###################################
## Predict NewData Using Model Object
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## train dataset
modelObject <- gradDescentR.learn(dataSet, featureScaling=TRUE, scalingMethod="VARIANCE",
               learningMethod="SARAHPlus", control=list(), seed=NULL)
## create new data input
temp <- c(273.1, 353.1, 363.1)
pres <- c(24.675, 24.675, 24.675)
conf <- c(0.8066773, 0.9235751, 0.9325948)
zfac <- data.frame(temp, pres, conf)
## predict
prediction_data <- predict(modelObject, zfac)
```

`prediction` *Predicting Function for Linear Model*

Description

A function to predict testing data with built gradient descent model

Usage

```
prediction(model, dataTestInput)
```

Arguments

`model`	a matrix of coefficients used as a linear model to predict testing data input. This parameter exclusively produced by the gradient-descent-based learning function.
`dataTestInput`	a data.frame represented dataset with input variables only ($m \times n - 1$), where m is the number of instances and n is the number of input variables only.

Details

This function used to predict testing data with only input variable named `dataTestInput`. The `model` parameter is the coefficients that produced by gradient-descent-based learning function. The result of this function is a dataset that contains `dataTestInput` combined with prediction data as the last column of dataset.

Value

a data.frame of testing data input variables and prediction variables.

See Also

GD, MBGD, SGD, SAGD, MGD, AGD, ADAGRAD, ADADELTA, RMSPROP, ADAM, SSGD, SVRG, SARAH, SARAHPlus

Examples

```
#################################
## Predict Testing Data Using GD Model
## load R Package data
data(gradDescentRData)
## get z-factor Data
dataSet <- gradDescentRData$CompressilbilityFactor
## do variance scaling to dataset
featureScalingResult <- varianceScaling(dataSet)
## split dataset
splitedDataSet <- splitData(featureScalingResult$scaledDataSet)
## built model using GD
model <- GD(splitedDataSet$dataTrain)
## separate testing data with input only
dataTestInput <- (splitedDataSet$dataTest)[,1:ncol(splitedDataSet$dataTest)-1]
## predict testing data using GD model
prediction <- prediction(model,dataTestInput)
## show result()
prediction


#################################
## Predict Testing Data Using SARAHPlus Model
## load R Package data
data(gradDescentRData)
## get z-factor Data
dataSet <- gradDescentRData$CompressilbilityFactor
## do variance scaling to dataset
```

```
featureScalingResult <- varianceScaling(dataSet)
## split dataset
splitedDataSet <- splitData(featureScalingResult$scaledDataSet)
## built model using SARAHPlus
model <- SARAHPlus(splitedDataSet$dataTrain, alpha=0.1, maxIter=10,
         innerIter=10, gammaS=0.125, seed=NULL)
## separate testing data with input only
dataTestInput <- (splitedDataSet$dataTest)[,1:ncol(splitedDataSet$dataTest)-1]
## predict testing data using GD model
prediction <- prediction(model,dataTestInput)
## show result()
prediction
```

RMSE *RMSE Calculator Function*

Description

A function to calculate error using Root-Mean-Square-Error

Usage

```
RMSE(outputData, prediction)
```

Arguments

`outputData`	a data.frame represented dataset with output variable only ($m \times 1$), where m is the number of instances has one variable, which is the output.
`prediction`	a data.frame represented prediction data with output variable only ($m \times 1$), where m is the number of instances has one variable, which is the output.

Details

This function used to calculate the error between two variables. `outputData` is the first parameter of this function representing the real output value. `prediction` is the second parameter of this function representing the prediction value.

Value

a float value represent the average error of the prediction

Examples

```
#################################
## Calculate Error using RMSE
## load R Package data
data(gradDescentRData)
## get z-factor Data
```

```
dataSet <- gradDescentRData$CompressilbilityFactor
## do variance scaling to dataset
featureScalingResult <- varianceScaling(dataSet)
## split dataset
splitedDataSet <- splitData(featureScalingResult$scaledDataSet)
## built model using GD
model <- GD(splitedDataSet$dataTrain)
## separate testing data with input only
dataTestInput <- (splitedDataSet$dataTest)[,1:ncol(splitedDataSet$dataTest)-1]
## predict testing data using GD model
predictionData <- prediction(model, dataTestInput)
## calculate error using rmse
errorValue <- RMSE(splitedDataSet$dataTest[,ncol(splitedDataSet$dataTest)],
predictionData[,ncol(predictionData)])
## show result
errorValue
```

RMSPROP *ADADELTA Method Learning Function*

Description

A function to build prediction model using RMSPROP method.

Usage

```
RMSPROP(dataTrain, alpha = 0.1, maxIter = 10, momentum = 0.9,
  seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
momentum	a float value represent momentum give a constant speed to learning process.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD with an optimization to create an adaptive learning rate by RMS cost and hessian approximation correction approach. In other word, this method combine the ADAGRAD and ADADELTA approaches.

Value

a vector matrix of theta (coefficient) for linear model.

References

M. D. Zeiler Adadelta: An Adaptive Learning Rate Method, arXiv: 1212.5701v1, pp. 1-6 (2012)

See Also

ADAGRAD, ADADELTA, ADAM

Examples

```
#################################
## Learning and Build Model with RMSPROP
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with RMSPROP
RMSPROPmodel <- RMSPROP(splitedDataSet$dataTrain)
#show result
print(RMSPROPmodel)
```

SAGD *Stochastic Average Gradient Descent (SAGD) Method Learning Function*

Description

A function to build prediction model using Stochastic Average Gradient Descent (SAGD) method.

Usage

```
SAGD(dataTrain, alpha = 0.1, maxIter = 10, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD that only compute one instances of of training data stochasticaly. But SAGD has an averaging control optimization to decide between do the coefficient update or not randomly. This optimization will speed-up the learning, if it doesn't perform computation and update the coefficient.

Value

a vector matrix of theta (coefficient) for linear model.

References

M. Schmidt, N. Le Roux, F. Bach Minimizing Finite Sums with the Stochastic Average Gradient, INRIA-SIERRA Project - Team Departement d'informatique de l'Ecole Normale Superieure, (2013)

See Also

SGD

Examples

```
###################################
## Learning and Build Model with SAGD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with SAGD
SAGDmodel <- SAGD(splitedDataSet$dataTrain)
#show result
print(SAGDmodel)
```

SARAH *Stochastic Recursive Gradient Algorithm (SARAH) Method Learning Function*

Description

A function to build prediction model using SARAH method.

Usage

```
SARAH(dataTrain, alpha = 0.1, maxIter = 10, innerIter = 10, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations in outerloop.
innerIter	the maximal number of iterations in innerloop.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function Similarly to SVRG, SARAH iterations are divided into the outer loop where a full gradient is computed and the inner loop where only stochastic gradient is computed. Unlike the case of SVRG, the steps of the inner loop of SARAH are based on accumulated stochastic information.

Value

a vector matrix of theta (coefficient) for linear model.

References

Lam M. Nguyen, Jie Lu, Katya Scheinberg, Martin Takac SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient, arXiv preprint arXiv:1703.00102, (2017)

See Also

SVRG, SSGD, SARAHPlus

Examples

```
##################################
## Learning and Build Model with SARAH
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with SARAH
SARAHmodel <- SARAH(splitedDataSet$dataTrain)
#show result
print(SARAHmodel)
```

SARAHPlus *Stochastic Recursive Gradient Algorithm+ (SARAH+) Method Learning Function*

Description

A function to build prediction model using SARAH+ method.

Usage

```
SARAHPlus(dataTrain, alpha = 0.1, maxIter = 10, innerIter = 10,
  gammaS = 0.125, seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations in outerloop.
innerIter	the maximal number of iterations in innerloop.
gammaS	a float value to provide sufficient reduction. Default value is 0.125
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function is practical variant of SARAH, SARAHPlus provides a possibility of earlier termination and unnecessary careful choices of maximum innerloop size, and it also covers the classical gradient descent when we set gammaS = 1 (since the while loop does not proceed).

Value

a vector matrix of theta (coefficient) for linear model.

References

Lam M. Nguyen, Jie Lu, Katya Scheinberg, Martin Takac SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient, arXiv preprint arXiv:1703.00102, (2017)

See Also

SVRG, SSGD, SARAH

Examples

```
#################################
## Learning and Build Model with SARAH+
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with SARAH+
SARAHPlusmodel <- SARAHPlus(splitedDataSet$dataTrain)
#show result
print(SARAHPlusmodel)
```

SGD *Stochastic Gradient Descent (SGD) Method Learning Function*

Description

A function to build prediction model using Stochastic Gradient Descent (SGD) method.

Usage

```
SGD(dataTrain, alpha = 0.1, maxIter = 10, seed = NULL)
```

Arguments

`dataTrain`	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
`alpha`	a float value representing learning rate. Default value is 0.1
`maxIter`	the maximal number of iterations.
`seed`	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on GD method with optimization to use only one instance of training data stochasticaly. So, SGD will perform fast computation and the learning. However, the learning to reach minimum cost will become more unstable.

Value

a vector matrix of theta (coefficient) for linear model.

References

N. Le Roux, M. Schmidt, F. Bach A Stochastic Gradient Method with an Exceptional Convergence Rate for Finite Training Sets, Advances in Neural Information Processing Systems, (2011)

See Also

SAGD

Examples

```
#################################
## Learning and Build Model with SGD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with SGD
SGDmodel <- SGD(splitedDataSet$dataTrain)
#show result
print(SGDmodel)
```

`splitData` *The Data Spliting Function*

Description

A function to split dataset into training and testing data

Usage

```
splitData(dataSet, dataTrainRate = 0.5, seed = NULL)
```

Arguments

`dataSet`	a data.frame that representing dataset ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataSet must have at least two columns and ten rows of data that contain only numbers (integer or float).
`dataTrainRate`	a float number between 0 to 1 representing the training data rate of given dataset. This parameter has default value of 0.5.
`seed`	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function split dataset into training and testing data. By default, this function split dataset into 50 `dataTest`. You can decide the training data rate by change the value of `dataTrainRate`. Example, if you want to set the training data rate by 80 As the remaining of `dataTrainRate` value, which is 0.2, will be set as `dataTest` rate.

Value

a list contains data.frame of training data and testing data.

Examples

```
##################################
## Splitting Dataset into Training and Testing Data
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
#show result
print(splitedDataSet$dataTrain)
print(splitedDataSet$dataTest)
```

`SSGD` *Semi Stochastic Gradient Descent (SSGD) Method Learning Function*

Description

A function to build prediction model using SSGD method.

Usage

```
SSGD(dataTrain, alpha = 0.1, maxIter = 10, lamda = 0, innerIter = 10,
  seed = NULL)
```

Arguments

`dataTrain`	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
`alpha`	a float value representing learning rate. Default value is 0.1
`maxIter`	the maximal number of iterations in outerloop.
`lamda`	a float value to generate random value from innerIter with probability for inner-loop.

innerIter	the maximal number of iterations in innerloop.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function combines elements from both GD and SGD. SSGD starts by computing the full gradient once and then proceeds with stochastic updates by choosing one of the gradients at a time.

Value

a vector matrix of theta (coefficient) for linear model.

References

George Papamakarios Comparison of Modern Stochastic Optimization Algorithms, (2014)

See Also

SVRG, SARAH, SARAHPlus

Examples

```
##################################
## Learning and Build Model with SSGD
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with SSGD
SSGDmodel <- SSGD(splitedDataSet$dataTrain)
#show result
print(SSGDmodel)
```

SVRG *Stochastic Variance Reduce Gradient (SVRG) Method Learning Function*

Description

A function to build prediction model using SVRG method.

Usage

```
SVRG(dataTrain, alpha = 0.1, maxIter = 10, innerIter = 10, option = 2,
  seed = NULL)
```

Arguments

dataTrain	a data.frame that representing training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
alpha	a float value representing learning rate. Default value is 0.1
maxIter	the maximal number of iterations in outerloop.
innerIter	the maximal number of iterations in innerloop.
option	is an option to set the theta. option 1 set the theta with the last theta in innerloop. option 2 set the theta with random theta from 1 to last innerloop.
seed	a integer value for static random. Default value is NULL, which means the function will not do static random.

Details

This function based on SGD with an optimization that accelerates the process toward converging by reducing the gradient in SGD

Value

a vector matrix of theta (coefficient) for linear model.

References

Rie Johnson, Tong Zang Accelerating Stochastic Gradient Descent using Predictive Variance Reduction, Advances in Neural Information Processing Systems, pp. 315-323 (2013)

See Also

SSGD, SARAH, SARAHPlus

Examples

```
##################################
## Learning and Build Model with SVRG
## load R Package data
data(gradDescentRData)
## get z-factor data
dataSet <- gradDescentRData$CompressilbilityFactor
## split dataset
splitedDataSet <- splitData(dataSet)
## build model with SVRG
SVRGmodel <- SVRG(splitedDataSet$dataTrain)
#show result
print(SVRGmodel)
```

varianceDescaling *Variance/Standardization Revert Function*

Description

A function to revert the value that has been done by variance/ . standardization scaling method.

Usage

```
varianceDescaling(dataSet, varianceParameter)
```

Arguments

dataSet	a data.frame that representing dataset ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataSet must have at leas two columns and ten rows of data that contain only numbers (integer or float).
varianceParameter	a matrix that has value of variance scaling parameter, such as mean value and standard deviation value of data that can be used to restore the original value of dataset. This parameter is exclusively produced by varianceScaling function.

Details

This function changes the value of variance scaled dataset that produced by varianceScaling function and represented by data.frame object.

Value

a data.frame representing reverted dataset value

See Also

varianceScaling

Examples

```
##################################
## Revert Variance Scaling
## load R Package data
data(gradDescentRData)
## get z-factor Data
dataSet <- gradDescentRData$CompressilbilityFactor
fsr <- varianceScaling(dataSet)
rfsr <- varianceDescaling(fsr$scaledDataSet, fsr$scalingParameter)
```

varianceScaling *The Variance/Standardization Feature Scaling Function*

Description

A function to do feature scaling to dataset with variance/standardization scaling method .

Usage

```
varianceScaling(dataSet)
```

Arguments

dataSet	a data.frame that representing dataset to be processed. dataSet must have at leas two columns and ten rows of data that contain only numbers (integer or float). The last column to the left will be defined as output variable.

Details

This function changes the value of dataset that represented by data.frame object into variance scaled value that has interval value near -1 to 1.

Value

a list contains feature scaled dataset and scaling parameter

See Also

varianceDescaling

Examples

```
#################################
## Feature scaling with Variance Scaling Method
## load R Package data
data(gradDescentRData)
## get z-factor Data
dataSet <- gradDescentRData$CompressilbilityFactor
## do variance scaling to dataset
featureScalingResult <- varianceScaling(dataSet)
## show result
print(featureScalingResult$scaledDataSet)
print(featureScalingResult$scalingParameter)
```

Index