Package 'bigSurvSGD'

October 12, 2022

Type Package

Title Big Survival Analysis Using Stochastic Gradient Descent

Version 0.0.1

Date 2020-09-11

Description Fits Cox model via stochastic gradient descent. This implementation avoids computational instability of the standard Cox Model when dealing large datasets. Furthermore, it scales up with large datasets that do not fit the memory. It also handles large sparse datasets using proximal stochastic gradient descent algorithm. For more details about the method, please see Aliasghar Tarkhan and Noah Simon (2020) <arXiv:2003.00116v2>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.4), bigmemory, doParallel, survival

Depends foreach, parallel, R (>= 3.5.0)

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.1.0

BugReports https://github.com/atarkhan/bigSurvSGD/issues

NeedsCompilation yes

Author Aliasghar Tarkhan [aut, cre],
Noah Simon [aut]

Maintainer Aliasghar Tarkhan <atarkhan@uw.edu>

Repository CRAN

Date/Publication 2020-10-01 08:40:02 UTC

R topics documented:

---

bigSurvSGD *Big survival data analysis using stochastic gradient descent*

Description

Fits Cox model via stochastic gradient descent (SGD). This implementation avoids computational instability of the standard Cox Model when datasets are large. Furthermore, it scales up with very large datasets that do not fit the memory. It also handles large sparse datasets using the proximal stochastic gradient descent algorithm. For more details about the method, please see Aliasghar Tarkhan and Noah Simon (2020) <arXiv:2003.00116v2>.

Usage

```
bigSurvSGD(
  formula = Surv(time = time, status = status) ~ .,
  data,
  norm.method = "standardize",
  features.mean = NULL,
  features.sd = NULL,
  opt.method = "AMSGrad",
  beta.init = NULL,
  beta.type = "averaged",
  lr.const = 0.12,
  lr.tau = 0.5,
  strata.size = 20,
  batch.size = 1,
  num.epoch = 100,
  b1 = 0.9,
  b2 = 0.99,
  eps = 1e-08,
  inference.method = "plugin",
  num.boot = 1000,
  num.epoch.boot = 100,
  boot.method = "SGD",
  lr.const.boot = 0.12,
  lr.tau.boot = 0.5,
  num.sample.strata = 1000,
  sig.level = 0.05,
  beta0 = 0,
  alpha = NULL,
  lambda = NULL,
  nlambda = 100,
  num.strata.lambda = 10,
  lambda.scale = 1,
  parallel.flag = FALSE,
  num.cores = NULL,
```

```
  bigmemory.flag = FALSE,
  num.rows.chunk = 1e+06,
  col.names = NULL
)

## S3 method for class 'bigSurvSGD'
print(x, ...)

## S3 method for class 'bigSurvSGD'
plot(x, ...)
```

Arguments

formula	a formula in format of Surv(time=time, status=status)~feature1+feature2+... describing time-to-event variable, status variable, and features to be included in model. Default is "Surv(time, status)~." that regresses on all the features included in the dataset.
data	survival dataset. It can be in form of data.frame or a path to a .csv file if we aim not to read data off the memory. If we aim to read data off the memory, it must be a path to a .csv data.
norm.method	normalization method before starting the analysis. "center" only centers the features by subtracting the mean, "scale" only scales the features by dividing features to their standard deviation, "normalization" does both centering and scaling, and "none" does not perform any pre-processing. The default is "normalization".
features.mean	mean vector of features used for normalization. The default is NULL where our alorithm calculates it.
features.sd	standard deviation vector of features used for normalization. The default is NULL where our alorithm calculates it.
opt.method	optimization algorithm: "SGD" estimates the coefficients using the standard stochastic gradient descent; "ADAM" estimates the coefficients using ADAM optimizer; "AMSGrad" estimates the coefficients using AMSGrad optimizer. The default is "AMSGrad".
beta.init	initialization for coefficient. The default is NULL where our algorithm starts with an all-zero vector.
beta.type	type of coefficient to be returned. If specified as "single", the last updated coefficient is returned. If specified as "averaged", the Polyak-Ruppert (i.e., average over iterates) is returned. The default is "averaged".
lr.const	proportional constant for the learning rate. The higher values give faster but noisier estimates and vice versa. The default is 0.12 for "AMSGrad" optimizer.
lr.tau	the power of iteration index in the learning rate. The bigger value represents the faster decay in the lerning rate and vice versa. The default is 0.5.
strata.size	strata size. The default is 20 patients per stratum.
batch.size	batch size. The default is 1 stratum per batch.
num.epoch	Number of epochs for the SGD-based algorithms. The default is 100.

b1 hyper parameter for "AMSGrad" and "ADAM". The default is 0.9. See https://arxiv.org/abs/1412.6980 for "ADMA" and https://arxiv.org/abs/1904.03590 for "AMSGrad".

b2 hyper parameter for "AMSGrad" and "ADAM". The default is 0.99.

eps hyper parameter for "AMSGrad" and "ADAM". The default is 1e-8.

inference.method method for inference, i.e., constructing confidence interval (CI): "bootstrap" constructs CI usin non-parametric bootstrap; "plugin": constructs CI using asymptotic properties of U-statistics; The default is "plugin" which returns estimates, confidence intervals, test statistics, and p-values.

num.boot number of boostrap resamples. Default is 1000.

num.epoch.boot number of epochs for each boorstrap resamples. Default is 100.

boot.method optimization method for bootstrap. Default is "SGD".

lr.const.boot proportional constant for the learning rate for bootstrap resamples. Defauls is "0.12"

lr.tau.boot power of iteration index in the learning rate for bootstrap resamples. Defauls is "0.5"

num.sample.strata number of sample strata per observation to estimate standard error using plugin method. Default value is 1000.

sig.level significance level for constructing (1-sig.level) confidence interval. Default is 0.05.

beta0 null vector of coefficients for calculating p-value using plugin method. Default is zero vector.

alpha penalty coeficient between 0 and 1. alpha=0 only considers the ridge penlaty and alpha=1 only considers the lasso penalty. Otherwise, it considers a convex combination of these two penalties. Defualt is NULL, i.e., no penalty.

lambda coeficient for the elastic net penalty. There are three possible scenarios: (1) If alpha is defined NULL, no penalty (ridge or lasso) is considered regardless of values of lambda; (2) If alpha is not NULL but lambda is NULL, it first calculates the largest value of lambda (lambda.max) for which all coefficients become zero. Then it considers an exponentially decreasing sequence of lambda starting from lambda.max ges toward lambda.min (lambda.min=0.01*lambda.max if p>n, otherwise lambda.min=0.0001*lambda.max) and return their corresponding coefficients. (3) If a value for lambda is specified, our algorithm returns coefficients for specified pair of (lambda, alpha). The default is NULL.

nlambda number of elements to be considered for scenario (2) above. Default is 100.

num.strata.lambda number of sample strata to estimate maximum lambda (lambda.max) when alpha is not NULL and lambda=NULL (see lambda).

lambda.scale we scale lambda.max to make sure we start with a lambda for which we get all coefficients equal to 0. Default is 1.

parallel.flag to specify if we want to use parallel computing for inference. Default is "F", i.e., no parallel computing.

num.cores	number of cores for parallel computing. The default is "NULL" for which if parallel.flag=T, it uses all available cores on your system.
bigmemory.flag	determins if data needs to be read off the memory in case data does not fit memory. Default is F, not to use bigmemoty package.
num.rows.chunk	maximum number of rows per chunk to be read off the memory. This is crucial for the large datasets that do not fit memory. Use fewer number of rows for the large number of features, especially if you receive an error due to lack of memory. The default value is 1e6 rows.
col.names	a vector of characters for column names of data. If NULL, the column names of dataset "data" will be selected. The default is NULL (i.e., reads columns of given dataset).
x	a 'bigSurvSGD' object
...	additional argument used

Value

coef: Log of hazards ratio. If no inference is used, it returns a vector for estimated coefficients: If inference is used, it returns a matrix including estimates and confidence intervals of coefficients. In case of penalization, it resturns a matrix with columns corresponding to lambdas.

coef.exp: Exponentiated version of coef (hazards ratio).

lambda: Returns lambda(s) used for penalizarion.

alpha: Returns alpha used for penalizarion.

features.mean: Returns means of features, if given or calculated

features.sd: Returns standard deviations of features, if given or calculated.

Examples

```
# Simulated survival data - just estimation and no confidence interval
data(survData) # a dataset with 1000 observations (rows) and 10 features (columns)
resultsBig <- bigSurvSGD(formula=Surv(time, status)~.,data=survData, inference.method="none",
parallel.flag=TRUE, num.cores=2)
resultsBig



# Simulated survival data
data(survData) # a dataset with 1000 observations (rows) and 10 features (columns)
resultsBig <- bigSurvSGD(formula=Surv(time, status)~.,data=survData, inference="none",
parallel.flag=TRUE, num.cores=2)
resultsBig




# Simulated survival data to be read off the memory
data(survData) # a dataset with 1000 observations (rows) and 10 features (columns)
# Save dataset survSGD as bigSurvSGD to be read chunk-by-chunk off the memory
write.csv(survData, file.path(tempdir(), "bigSurvData.csv"), row.names = FALSE)
```

```
dataPath <- file.path(tempdir(), "bigSurvData.csv") # path to where data is
resultsBigOffMemory <- bigSurvSGD(formula=Surv(time, status)~., data=dataPath,
bigmemory.flag=TRUE, parallel.flag=TRUE, num.cores=2)
resultsBigOffMemory



# Simulated sparse survival data
data(sparseSurvData) # a sparse data with 100 observations (rows) and 150 features (columns)
resultsBigSparse <- bigSurvSGD(formula=Surv(time, status)~.,data=sparseSurvData,
alpha=0.9, lambda=0.1)
resultsBigSparse
```

`lambdaMaxC` *Calculates the maximum penalty coefficient lambda for which all coefficients become zero*

Description

Calculates the maximum penalty coefficient lambda for which all coefficients become zero

---

`oneChunkC` *Updates the coefficients based on one pass of data*

Description

Updates the coefficients based on one pass of data

`oneObsPlugingC` *Calculates the gradient and Hessian corresponding to one individual*

Description

Calculates the gradient and Hessian corresponding to one individual

sparseSurvData *Simulated sparse survival dataset*

Description

Simulated sparse survival dataset

Usage

```
data(sparseSurvData)
```

Format

An object of class data.frame including 100 observations (rows) and 150 features (columns)

References

Ralf Bender, Thomas Augustin, and Maria Blettner (Generating survival times to simulate Cox proportional hazards models)

Examples

```
data(sparseSurvData)
```

survData *Simulated survival dataset*

Description

Simulated survival dataset

Usage

```
data(survData)
```

Format

An object of class data.frame including 1000 observations (rows) and 10 features (columns)

References

Ralf Bender, Thomas Augustin, and Maria Blettner (Generating survival times to simulate Cox proportional hazards models)

Examples

```
data(survData)
```

Index