bigSurvSGD: Big Survival Data Analysis via Stochastic Gradient Descent

Aliasghar (Arash) Tarkhan and Noah Simon

2020-07-20

Introduction

We give a short tutorial on using bigSurvSGD package. bigSurvSGD package proposed a new framework based on stochastic gradient descent algorithm for fitting Cox proportional hazards model. In prticular, it works well when the datasets are too large to fit the momory where the standard Cox proportional hazards model fails. This package also fit regularized Cox proportional hazards model by adding an elastic net penalty. We give a couple of examples on how to use this package.

Installing and loading package

Intallating package

Like many other R packages, the simplest way to obtain `bigSurvSGD` is to install it directly from CRAN:

```
install.packages("bigSurvSGD", repos = "http://cran.us.r-project.org")
#> Installing package into '/home/arash/R/x86_64-pc-linux-gnu-library/3.6'
#> (as 'lib' is unspecified)
#> Warning: package 'bigSurvSGD' is not available (for R version 3.6.1)
```

Users may change the `repos` options depending on their locations and preferences.

Loading package

We load the package `bigSurvSGD` as folowing

```
library(bigSurvSGD)
#> Loading required package: foreach
#> Loading required package: parallel
```

Fitting Cox model

Loading data

We first load dataset `flchain` included in survival package:

```
library(survival)
data("flchain") # FLCHAIN data included in survival package
data("survData")
```

Note that the current version only supports numerical features without missing data. For example, for `flchain` data, we remove missing data and convert variable `sex` into numerical (binary) variable.

```r
flchain <- flchain[,-11] # removes chapter variable as cause of death
flchain <- na.omit(flchain) # removes missing values
flchain$sex <- ifelse(flchain$sex=="F",1,0) # converts sex variable to 0 and 1
```

Now we use `bigSurvSGD` function to estimate coefficient

```r
fitBigSurvSGD <- bigSurvSGD(formula=Surv(time, status)~.,data=survData,
                            parallel.flag=TRUE, num.cores = 2,
                            inference.method = "none")
fitBigSurvSGD
#> $coef
#>             [,1]
#>  [1,] 0.9970035
#>  [2,] 1.0417054
#>  [3,] 0.9200843
#>  [4,] 1.0593661
#>  [5,] 0.9542722
#>  [6,] 1.0282457
#>  [7,] 1.0219863
#>  [8,] 0.9907794
#>  [9,] 0.9170343
#> [10,] 0.9131081
#> attr(,"names")
#>  [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10"
#>
#> $coef.exp
#>             [,1]
#>  [1,] 2.710149
#>  [2,] 2.834046
#>  [3,] 2.509502
#>  [4,] 2.884542
#>  [5,] 2.596780
#>  [6,] 2.796156
#>  [7,] 2.778709
#>  [8,] 2.693333
#>  [9,] 2.501860
#> [10,] 2.492056
#> attr(,"names")
#>  [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10"
#>
#> $lambda
#> [1] 0
#>
#> $alpha
#> [1] 0
#>
#> $features.mean
#>          X1          X2          X3          X4          X5          X6
#> -0.02081123 -0.03844643 -0.01947003  0.02845033 -0.01993854  0.01782694
#>          X7          X8          X9         X10
#> -0.01202580  0.04699838 -0.00599775  0.01309362
#>
#> $features.sd
#>        X1        X2        X3        X4        X5        X6        X7        X8
```

```
#> 1.0170818 1.0161579 1.0041936 0.9956090 1.0160725 0.9824744 0.9993097 1.0000475
#>        X9       X10
#> 0.9905759 1.0180216
```

Using standard Cox Model we have

```
fitCox <- coxph(formula=Surv(time, status)~.,data=survData)
fitCox
#> Call:
#> coxph(formula = Surv(time, status) ~ ., data = survData)
#>
#>        coef exp(coef) se(coef)     z      p
#> X1  0.98008   2.66467  0.04374 22.41 <2e-16
#> X2  1.01072   2.74757  0.04430 22.82 <2e-16
#> X3  0.90612   2.47471  0.04297 21.09 <2e-16
#> X4  1.01932   2.77131  0.04460 22.85 <2e-16
#> X5  0.95135   2.58920  0.04401 21.62 <2e-16
#> X6  1.00906   2.74303  0.04490 22.48 <2e-16
#> X7  0.98892   2.68834  0.04421 22.37 <2e-16
#> X8  0.97359   2.64743  0.04449 21.88 <2e-16
#> X9  0.91224   2.48990  0.04505 20.25 <2e-16
#> X10 0.90848   2.48054  0.04359 20.84 <2e-16
#>
#> Likelihood ratio test=1748  on 10 df, p=< 2.2e-16
#> n= 1000, number of events= 823
```

Users need to use `formula` to specify `time-to-event` and `status` variables if they were nemaed deifferently from `time` and `status`. User may also need to specify a subset of features they need to include in the model. For example, suppose that variable `t` and `s` represents `time-to-event` and `status`, and you need to only include features $X1$, $X2$, and $X3$:

```
colnames(survData)[c(1,2)] <- c("t", "s")
fitBigSurvSGD <- bigSurvSGD(formula=Surv(time=t, status=s)~X1+X2+X3, data=survData,
                            parallel.flag=TRUE, num.cores = 2,
                            inference.method = "none")
```

Using standard Cox Model we have

```
fitCox <- coxph(formula=Surv(t, s)~X1+X2+X3, data=survData)
fitCox
#> Call:
#> coxph(formula = Surv(t, s) ~ X1 + X2 + X3, data = survData)
#>
#>       coef exp(coef) se(coef)      z      p
#> X1 0.37111   1.44934  0.03628 10.229 <2e-16
#> X2 0.32047   1.37778  0.03542  9.049 <2e-16
#> X3 0.34446   1.41123  0.03655  9.424 <2e-16
#>
#> Likelihood ratio test=252.2  on 3 df, p=< 2.2e-16
#> n= 1000, number of events= 823
```

Fitting large dataset of the hard drive

Package "bigSurvSGD" package can handle large datasets that do not fit the memory. For an example, dataset "survData" is very big and we access it through path "/home/arsh/bigSurvData.csv"

```
data("survData")
write.csv(survData, file = "/home/arash/bigSurvData.csv", row.names = F)
```

Suupose that our dataset "bigSurvData" is too large to fit the memory. In practice, this number is the maximum number of rows of data for which you can use R without lack of memory error. Suppose that memory can only handle up to 100 rows of "bigSurdData". Then we ask "bigSurvSGD" to use "bigmemory" package (by defining "bigmemory.flag=T") to read data chunk-by-chunk off the memory with chunk size equal to 100.

```
fitBigSurvSGD <- bigSurvSGD(formula=Surv(time=time, status=status)~.,
                            data="/home/arash/bigSurvData.csv",
                            bigmemory.flag = T, num.rows.chunk = 100,
                            inference.method = "none")
#> Warning in read.big.matrix(filename = data, sep = ",", skip = 0, header = T):
#> Because type was not specified, we chose double based on the first line of data.
fitBigSurvSGD
#> $coef
#>             [,1]
#>  [1,] 0.9945750
#>  [2,] 1.0404640
#>  [3,] 0.9148412
#>  [4,] 1.0547785
#>  [5,] 0.9510500
#>  [6,] 1.0238611
#>  [7,] 1.0218987
#>  [8,] 0.9823649
#>  [9,] 0.9147683
#> [10,] 0.9125228
#> attr(,"names")
#>  [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10"
#>
#> $coef.exp
#>             [,1]
#>  [1,] 2.703575
#>  [2,] 2.830530
#>  [3,] 2.496379
#>  [4,] 2.871339
#>  [5,] 2.588426
#>  [6,] 2.783923
#>  [7,] 2.778465
#>  [8,] 2.670765
#>  [9,] 2.496197
#> [10,] 2.490598
#> attr(,"names")
#>  [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10"
#>
#> $lambda
#> [1] 0
#>
#> $alpha
#> [1] 0
#>
#> $features.mean
#>          X1          X2          X3          X4          X5          X6
```

```
#> -0.02081123 -0.03844643 -0.01947003  0.02845033 -0.01993854  0.01782694
#>          X7          X8          X9         X10
#> -0.01202580  0.04699838 -0.00599775  0.01309362
#>
#> $features.sd
#>        X1        X2        X3        X4        X5        X6        X7        X8
#> 1.0170818 1.0161579 1.0041936 0.9956090 1.0160725 0.9824744 0.9993097 1.0000475
#>        X9       X10
#> 0.9905759 1.0180216
fitCox <- coxph(formula=Surv(time, status)~., data=survData)
fitCox
#> Call:
#> coxph(formula = Surv(time, status) ~ ., data = survData)
#>
#>        coef exp(coef) se(coef)     z      p
#> X1  0.98008   2.66467  0.04374 22.41 <2e-16
#> X2  1.01072   2.74757  0.04430 22.82 <2e-16
#> X3  0.90612   2.47471  0.04297 21.09 <2e-16
#> X4  1.01932   2.77131  0.04460 22.85 <2e-16
#> X5  0.95135   2.58920  0.04401 21.62 <2e-16
#> X6  1.00906   2.74303  0.04490 22.48 <2e-16
#> X7  0.98892   2.68834  0.04421 22.37 <2e-16
#> X8  0.97359   2.64743  0.04449 21.88 <2e-16
#> X9  0.91224   2.48990  0.04505 20.25 <2e-16
#> X10 0.90848   2.48054  0.04359 20.84 <2e-16
#>
#> Likelihood ratio test=1748  on 10 df, p=< 2.2e-16
#> n= 1000, number of events= 823
```

Note that the current package supports ".csv" file.

Fitting Cox Model with sparse data

Now we consider a dataset "sparseSurvData" for which number of features (columns) is larger than number of observations (rows). "bigSurvSGD" package handles sparse datasets. The following fits a regularized Cox Model with the elastic net penalty where the penalty coefficients of l_1 and l_2 norms are $alpha * lambda = 0.09$ and $(1 - alpha) * lambda = 0.01$, respectively.

```
data("sparseSurvData")
fitBigSurvSGD <- bigSurvSGD(formula=Surv(time=time, status=status)~.,data=sparseSurvData,
                            alpha = 0.9, lambda = 0.1)
fitBigSurvSGD$coef
#>                 [,1]
#>   [1,]  9.928989e-01
#>   [2,]  1.259818e+00
#>   [3,]  7.467543e-01
#>   [4,]  4.658558e-01
#>   [5,]  1.597831e+00
#>   [6,]  8.800695e-01
#>   [7,]  1.188276e+00
#>   [8,]  8.971378e-01
#>   [9,]  8.731069e-01
#>  [10,]  7.653917e-01
#>  [11,]  4.731574e-01
```

```
#>  [12,] -5.483423e-01
#>  [13,]  3.698323e-02
#>  [14,] -2.403789e-01
#>  [15,] -4.003957e-02
#>  [16,]  4.762976e-01
#>  [17,]  1.821405e-01
#>  [18,]  1.604856e-01
#>  [19,] -2.469305e-01
#>  [20,]  1.793316e-01
#>  [21,]  1.589051e-01
#>  [22,] -4.283980e-01
#>  [23,]  4.194410e-04
#>  [24,]  1.282891e-01
#>  [25,]  4.131862e-04
#>  [26,]  9.630760e-03
#>  [27,]  6.516638e-01
#>  [28,] -3.771082e-03
#>  [29,]  0.000000e+00
#>  [30,] -2.450621e-01
#>  [31,] -4.629220e-03
#>  [32,] -1.576861e-01
#>  [33,]  6.280828e-02
#>  [34,] -2.193999e-02
#>  [35,]  2.878986e-06
#>  [36,]  5.765236e-01
#>  [37,]  1.258553e-03
#>  [38,] -1.373528e-01
#>  [39,]  3.527126e-02
#>  [40,] -1.611678e-01
#>  [41,]  1.155423e-04
#>  [42,]  7.243471e-02
#>  [43,]  1.618874e-01
#>  [44,]  2.051811e-01
#>  [45,] -1.988390e-01
#>  [46,]  3.896617e-02
#>  [47,] -2.448202e-01
#>  [48,] -1.323041e-01
#>  [49,]  1.960592e-04
#>  [50,] -6.321139e-02
#>  [51,] -6.713778e-03
#>  [52,]  4.547419e-02
#>  [53,] -4.624725e-03
#>  [54,] -1.021385e-01
#>  [55,]  1.696980e-01
#>  [56,]  1.578617e-03
#>  [57,] -7.558558e-01
#>  [58,] -5.380855e-02
#>  [59,]  4.217356e-02
#>  [60,]  1.987118e-01
#>  [61,] -1.005209e-02
#>  [62,]  3.185053e-01
#>  [63,] -1.584379e-01
#>  [64,]  4.570937e-03
```

```
#>  [65,]  2.478202e-02
#>  [66,]  0.000000e+00
#>  [67,]  1.234833e-02
#>  [68,]  1.742496e-02
#>  [69,] -9.294535e-03
#>  [70,]  9.657681e-01
#>  [71,] -8.663477e-02
#>  [72,]  5.194683e-01
#>  [73,]  1.013651e-01
#>  [74,] -4.122003e-04
#>  [75,]  3.527588e-02
#>  [76,] -2.151554e-01
#>  [77,]  2.748415e-03
#>  [78,] -8.446816e-03
#>  [79,]  8.121915e-03
#>  [80,]  6.186264e-03
#>  [81,]  1.075227e-01
#>  [82,]  1.580812e-02
#>  [83,]  1.549543e-01
#>  [84,]  7.361880e-02
#>  [85,]  1.547408e-02
#>  [86,] -9.066009e-02
#>  [87,] -2.531165e-01
#>  [88,]  0.000000e+00
#>  [89,]  2.141111e-04
#>  [90,]  3.318511e-01
#>  [91,] -2.351045e-01
#>  [92,]  6.601868e-02
#>  [93,]  4.452988e-01
#>  [94,]  0.000000e+00
#>  [95,] -1.580770e-02
#>  [96,] -5.791580e-03
#>  [97,]  0.000000e+00
#>  [98,]  6.093165e-01
#>  [99,]  2.728164e-01
#> [100,]  1.671929e-02
#> [101,]  2.275540e-01
#> [102,]  3.431311e-04
#> [103,] -1.544311e-01
#> [104,] -4.908005e-02
#> [105,] -1.493334e-01
#> [106,] -1.787881e-04
#> [107,] -3.798232e-03
#> [108,]  5.156129e-01
#> [109,]  5.217735e-02
#> [110,]  4.280265e-03
#> [111,] -4.056833e-01
#> [112,] -5.737261e-02
#> [113,] -3.424954e-01
#> [114,]  4.425969e-01
#> [115,] -2.416808e-01
#> [116,] -7.355062e-02
#> [117,]  0.000000e+00
```

```
#> [118,] -4.370634e-01
#> [119,] -1.119551e-01
#> [120,] -2.666351e-02
#> [121,] -2.039230e-01
#> [122,] -2.792385e-03
#> [123,]  5.055440e-02
#> [124,] -2.857717e-02
#> [125,]  5.228218e-05
#> [126,]  1.403765e-01
#> [127,]  6.055160e-04
#> [128,] -2.683914e-01
#> [129,] -1.019644e-01
#> [130,]  0.000000e+00
#> [131,] -3.646887e-02
#> [132,] -1.231943e-02
#> [133,]  3.442848e-02
#> [134,]  7.850954e-02
#> [135,]  2.045016e-02
#> [136,]  0.000000e+00
#> [137,]  8.425978e-04
#> [138,]  1.471476e-01
#> [139,]  4.202453e-03
#> [140,] -2.240197e-02
#> [141,] -1.464044e-01
#> [142,] -4.922503e-02
#> [143,]  1.652050e-01
#> [144,] -7.246086e-02
#> [145,]  3.354146e-03
#> [146,]  1.727318e-02
#> [147,] -2.256599e-01
#> [148,]  2.019590e-01
#> [149,] -2.779190e-01
#> [150,]  2.532293e-02
#> attr(,"names")
#>   [1] "X1"   "X2"   "X3"   "X4"   "X5"   "X6"   "X7"   "X8"   "X9"   "X10"
#>  [11] "X11"  "X12"  "X13"  "X14"  "X15"  "X16"  "X17"  "X18"  "X19"  "X20"
#>  [21] "X21"  "X22"  "X23"  "X24"  "X25"  "X26"  "X27"  "X28"  "X29"  "X30"
#>  [31] "X31"  "X32"  "X33"  "X34"  "X35"  "X36"  "X37"  "X38"  "X39"  "X40"
#>  [41] "X41"  "X42"  "X43"  "X44"  "X45"  "X46"  "X47"  "X48"  "X49"  "X50"
#>  [51] "X51"  "X52"  "X53"  "X54"  "X55"  "X56"  "X57"  "X58"  "X59"  "X60"
#>  [61] "X61"  "X62"  "X63"  "X64"  "X65"  "X66"  "X67"  "X68"  "X69"  "X70"
#>  [71] "X71"  "X72"  "X73"  "X74"  "X75"  "X76"  "X77"  "X78"  "X79"  "X80"
#>  [81] "X81"  "X82"  "X83"  "X84"  "X85"  "X86"  "X87"  "X88"  "X89"  "X90"
#>  [91] "X91"  "X92"  "X93"  "X94"  "X95"  "X96"  "X97"  "X98"  "X99"  "X100"
#> [101] "X101" "X102" "X103" "X104" "X105" "X106" "X107" "X108" "X109" "X110"
#> [111] "X111" "X112" "X113" "X114" "X115" "X116" "X117" "X118" "X119" "X120"
#> [121] "X121" "X122" "X123" "X124" "X125" "X126" "X127" "X128" "X129" "X130"
#> [131] "X131" "X132" "X133" "X134" "X135" "X136" "X137" "X138" "X139" "X140"
#> [141] "X141" "X142" "X143" "X144" "X145" "X146" "X147" "X148" "X149" "X150"
```