Gradient Bootstrapping

Anushree Shivarudrappa

June 15, 2016

1. Pre-Processing

```
library(data.table)
library(ggplot2)
library(dplyr)
library(scales)
library(RColorBrewer)
library(tidyr)
library(caTools)
library(rpart)
library(rpart.plot)
library(ROCR)
library(randomForest)
library(tree)
library(caret)
library(e1071)
library(gbm)
```

2. Data Loading

```
Death_US <- fread("DeathRecords.csv", header = T)
```

3. Selecting dataset for model

```
# separates natural death
Death_US_natural <- Death_US[Death_US$MannerOfDeath == 7, ]
```

Select required variables

```
require(MASS)
require(dplyr)
natural_sub <- Death_US_natural %>% dplyr::select(Education2003Revision, Sex, Age,
                      InfantAgeRecode22,
                      PlaceOfDeathAndDecedentsStatus, MaritalStatus, InjuryAtWork,
                      MannerOfDeath,
                      Autopsy, ActivityCode, PlaceOfInjury, Icd10Code,CauseRecode358,
                      CauseRecode113, InfantCauseRecode130,CauseRecode39,
                      NumberOfEntityAxisConditions,NumberOfRecordAxisConditions,Race)
```

Converting Character variable into Integer variable

```
natural_sub$Sex <- as.integer(as.factor(natural_sub$Sex))
natural_sub$MaritalStatus <- as.integer(as.factor(natural_sub$MaritalStatus))
natural_sub$InjuryAtWork <- as.integer(as.factor(natural_sub$InjuryAtWork))
natural_sub$Autopsy <- gsub("n", "N", natural_sub$Autopsy)
natural_sub$Autopsy <- as.integer(as.factor(natural_sub$Autopsy))
natural_sub$Icd10Code <- as.integer(as.factor(natural_sub$Icd10Code))
```

As we analyzed the feature variables are "Age + InfantAgeRecode22 + PlaceOfDeathAndDecedentsStatus + MaritalStatus + ActivityCode + PlaceOfInjury + NumberOfRecordAxisConditions + NumberOfEntityAxisConditions"

```
# Since the decision tree support till 32 levels removing 7 levels which has less entries
table(factor(natural_sub$CauseRecode39))
```

```
##
##      1      2      3      5      6      7      8      9     10     11
##    366     37   5619   9053  43839  33847 133412  34621  23359  23422
##     12     13     14     15     16     17     20     21     22     23
##  25734  17116  19671 133276  63721  75552  37415 310848 175752  23704
##     24     25     26     27     28     29     30     31     32     33
## 111664   5426  16551  45801 125752   2519  31595  41369   1000   9930
##     34     35     36     37     38     39     40     41     42
##   8110    414  23035 433081    212  13088      8      5      9
```

```
CauseExtraRemove <- natural_sub[, natural_sub$CauseRecode39 %in% c(2, 40, 41, 42, 38, 35, 1)]
table(CauseExtraRemove)
```

```
## CauseExtraRemove
##   FALSE    TRUE
## 2058882    1051
```

```
# remove the 7 factors levels from Death_US_natural dataset
natural_sub <- natural_sub[!(CauseExtraRemove)]
nrow(natural_sub)
```

```
## [1] 2058882
```

```
# model data
modeldata <- natural_sub

# We will do a random 70:30 split in our data set (70% will be for training models,
# 30% to evaluate them)
set.seed(111)
# randomly pick 70% of the number of observations
index <- sample.split(modeldata$CauseRecode39, SplitRatio = 0.7)
# subset data to include only the elements in the index
train <- subset(modeldata, index==T)
nrow(train)
```

```
## [1] 1441215
```

```r
# subset data to include all but the elements in the index
test <- subset(modeldata, index==F)
nrow(test)
```

```
## [1] 617667
```

```r
# take a copy of ICD10Code of test set and remove the variable from test set
Cause39 <- test$CauseRecode39
test$CauseRecode39 <- NULL
```

Model 3:: Gradient Bootstrapping

```r
gbm2 <- gbm(as.factor(CauseRecode39) ~ Age + InfantAgeRecode22 +
                     PlaceOfDeathAndDecedentsStatus + MaritalStatus + ActivityCode +
                       PlaceOfInjury + NumberOfRecordAxisConditions +
                     NumberOfEntityAxisConditions,
                  data = train,
            var.monotone=c(0,0,0,0,0,0,0,0),
                  # +1: monotone increase,
                  #  0: no monotone restrictions
                  distribution="gaussian",     # bernoulli, adaboost, gaussian,
                  # poisson, coxph, and quantile available
                  n.trees=3000,                # number of trees
                  shrinkage=0.005,             # shrinkage or learning rate,
                  # 0.001 to 0.1 usually work
                  interaction.depth=3,         # 1: additive model, 2: two-way interactions, etc.
                  bag.fraction = 0.5,          # subsampling fraction, 0.5 is probably best
                  n.minobsinnode = 10,         # minimum total weight needed in each node
                  cv.folds = 5,                # do 5-fold cross-validation
                  keep.data=TRUE,              # keep a copy of the dataset with the object
                  verbose=T )
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1       75.6015             nan     0.0050    0.0400
##      2       75.5615             nan     0.0050    0.0397
##      3       75.5224             nan     0.0050    0.0393
##      4       75.4835             nan     0.0050    0.0388
##      5       75.4448             nan     0.0050    0.0385
##      6       75.4066             nan     0.0050    0.0381
##      7       75.3689             nan     0.0050    0.0377
##      8       75.3313             nan     0.0050    0.0374
##      9       75.2943             nan     0.0050    0.0370
##     10       75.2577             nan     0.0050    0.0366
##     20       74.9083             nan     0.0050    0.0337
##     40       74.2900             nan     0.0050    0.0288
##     60       73.7363             nan     0.0050    0.0264
##     80       73.2566             nan     0.0050    0.0234
##    100       72.8224             nan     0.0050    0.0192
```

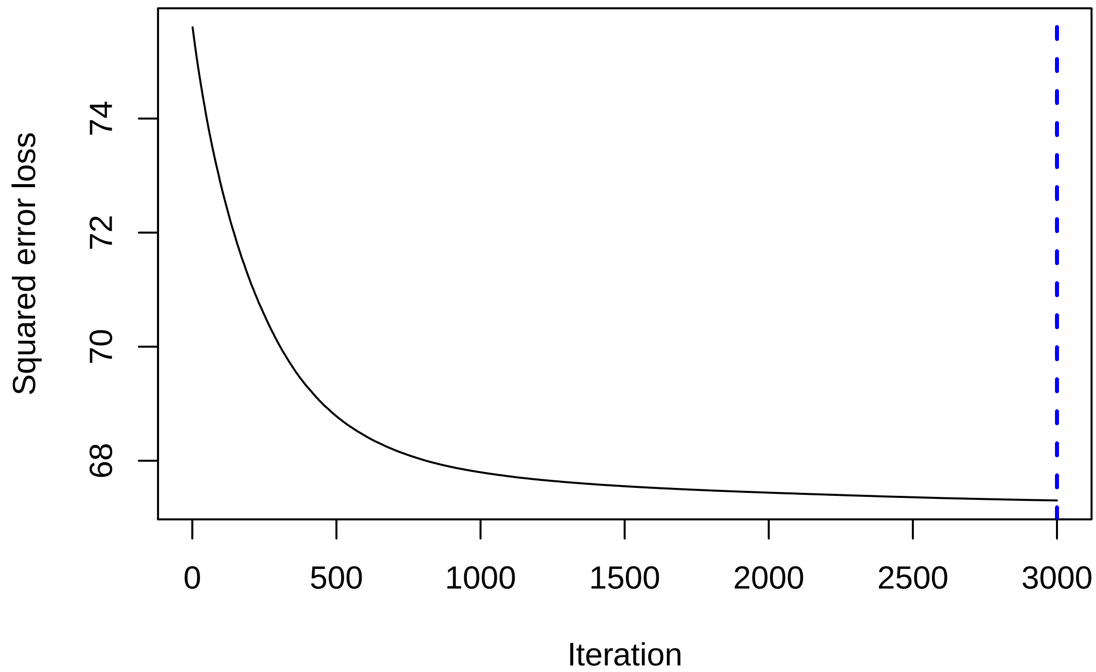
```
##    120        72.4316             nan     0.0050    0.0174
##    140        72.0683             nan     0.0050    0.0157
##    160        71.7419             nan     0.0050    0.0141
##    180        71.4435             nan     0.0050    0.0139
##    200        71.1583             nan     0.0050    0.0147
##    220        70.9027             nan     0.0050    0.0107
##    240        70.6670             nan     0.0050    0.0124
##    260        70.4454             nan     0.0050    0.0096
##    280        70.2388             nan     0.0050    0.0083
##    300        70.0459             nan     0.0050    0.0077
##    320        69.8689             nan     0.0050    0.0096
##    340        69.7060             nan     0.0050    0.0070
##    360        69.5513             nan     0.0050    0.0075
##    380        69.4165             nan     0.0050    0.0055
##    400        69.2881             nan     0.0050    0.0051
##    420        69.1719             nan     0.0050    0.0058
##    440        69.0600             nan     0.0050    0.0062
##    460        68.9597             nan     0.0050    0.0039
##    480        68.8681             nan     0.0050    0.0053
##    500        68.7815             nan     0.0050    0.0034
##    520        68.7014             nan     0.0050    0.0032
##    540        68.6258             nan     0.0050    0.0030
##    560        68.5593             nan     0.0050    0.0039
##    580        68.4960             nan     0.0050    0.0031
##    600        68.4365             nan     0.0050    0.0034
##    620        68.3792             nan     0.0050    0.0032
##    640        68.3287             nan     0.0050    0.0021
##    660        68.2821             nan     0.0050    0.0019
##    680        68.2350             nan     0.0050    0.0019
##    700        68.1918             nan     0.0050    0.0021
##    720        68.1525             nan     0.0050    0.0017
##    740        68.1167             nan     0.0050    0.0016
##    760        68.0812             nan     0.0050    0.0015
##    780        68.0480             nan     0.0050    0.0016
##    800        68.0167             nan     0.0050    0.0016
##    820        67.9875             nan     0.0050    0.0011
##    840        67.9613             nan     0.0050    0.0012
##    860        67.9357             nan     0.0050    0.0012
##    880        67.9116             nan     0.0050    0.0013
##    900        67.8900             nan     0.0050    0.0011
##    920        67.8688             nan     0.0050    0.0011
##    940        67.8505             nan     0.0050    0.0008
##    960        67.8318             nan     0.0050    0.0008
##    980        67.8143             nan     0.0050    0.0007
##   1000        67.7978             nan     0.0050    0.0007
##   1020        67.7821             nan     0.0050    0.0009
##   1040        67.7674             nan     0.0050    0.0006
##   1060        67.7533             nan     0.0050    0.0008
##   1080        67.7399             nan     0.0050    0.0005
##   1100        67.7267             nan     0.0050    0.0007
##   1120        67.7132             nan     0.0050    0.0007
##   1140        67.7016             nan     0.0050    0.0004
##   1160        67.6909             nan     0.0050    0.0004
##   1180        67.6793             nan     0.0050    0.0005
```

```
##    1200        67.6696             nan     0.0050    0.0006
##    1220        67.6598             nan     0.0050    0.0004
##    1240        67.6504             nan     0.0050    0.0003
##    1260        67.6417             nan     0.0050    0.0003
##    1280        67.6327             nan     0.0050    0.0004
##    1300        67.6241             nan     0.0050    0.0003
##    1320        67.6158             nan     0.0050    0.0003
##    1340        67.6081             nan     0.0050    0.0003
##    1360        67.6004             nan     0.0050    0.0005
##    1380        67.5933             nan     0.0050    0.0005
##    1400        67.5857             nan     0.0050    0.0003
##    1420        67.5786             nan     0.0050    0.0002
##    1440        67.5720             nan     0.0050    0.0003
##    1460        67.5659             nan     0.0050    0.0004
##    1480        67.5598             nan     0.0050    0.0002
##    1500        67.5533             nan     0.0050    0.0003
##    1520        67.5472             nan     0.0050    0.0003
##    1540        67.5418             nan     0.0050    0.0002
##    1560        67.5361             nan     0.0050    0.0002
##    1580        67.5305             nan     0.0050    0.0003
##    1600        67.5252             nan     0.0050    0.0002
##    1620        67.5199             nan     0.0050    0.0003
##    1640        67.5152             nan     0.0050    0.0002
##    1660        67.5106             nan     0.0050    0.0002
##    1680        67.5056             nan     0.0050    0.0003
##    1700        67.5008             nan     0.0050    0.0002
##    1720        67.4962             nan     0.0050    0.0002
##    1740        67.4921             nan     0.0050    0.0002
##    1760        67.4875             nan     0.0050    0.0003
##    1780        67.4831             nan     0.0050    0.0002
##    1800        67.4785             nan     0.0050    0.0003
##    1820        67.4748             nan     0.0050    0.0001
##    1840        67.4708             nan     0.0050    0.0002
##    1860        67.4666             nan     0.0050    0.0003
##    1880        67.4632             nan     0.0050    0.0001
##    1900        67.4588             nan     0.0050    0.0002
##    1920        67.4551             nan     0.0050    0.0002
##    1940        67.4518             nan     0.0050    0.0001
##    1960        67.4479             nan     0.0050    0.0002
##    1980        67.4444             nan     0.0050    0.0002
##    2000        67.4406             nan     0.0050    0.0002
##    2020        67.4368             nan     0.0050    0.0001
##    2040        67.4333             nan     0.0050    0.0001
##    2060        67.4301             nan     0.0050    0.0001
##    2080        67.4273             nan     0.0050    0.0001
##    2100        67.4234             nan     0.0050    0.0002
##    2120        67.4200             nan     0.0050    0.0001
##    2140        67.4164             nan     0.0050    0.0001
##    2160        67.4136             nan     0.0050    0.0001
##    2180        67.4101             nan     0.0050    0.0001
##    2200        67.4068             nan     0.0050    0.0001
##    2220        67.4035             nan     0.0050    0.0001
##    2240        67.4003             nan     0.0050    0.0002
##    2260        67.3964             nan     0.0050    0.0001
```

```
## 2280 67.3934 nan 0.0050 0.0003
## 2300 67.3902 nan 0.0050 0.0001
## 2320 67.3872 nan 0.0050 0.0003
## 2340 67.3842 nan 0.0050 0.0002
## 2360 67.3813 nan 0.0050 0.0001
## 2380 67.3778 nan 0.0050 0.0002
## 2400 67.3741 nan 0.0050 0.0001
## 2420 67.3713 nan 0.0050 0.0002
## 2440 67.3685 nan 0.0050 0.0001
## 2460 67.3660 nan 0.0050 0.0001
## 2480 67.3631 nan 0.0050 0.0001
## 2500 67.3604 nan 0.0050 0.0001
## 2520 67.3573 nan 0.0050 0.0000
## 2540 67.3545 nan 0.0050 0.0001
## 2560 67.3519 nan 0.0050 0.0001
## 2580 67.3487 nan 0.0050 0.0002
## 2600 67.3456 nan 0.0050 0.0002
## 2620 67.3432 nan 0.0050 0.0001
## 2640 67.3407 nan 0.0050 0.0001
## 2660 67.3389 nan 0.0050 0.0001
## 2680 67.3363 nan 0.0050 0.0002
## 2700 67.3341 nan 0.0050 0.0001
## 2720 67.3319 nan 0.0050 0.0001
## 2740 67.3296 nan 0.0050 0.0001
## 2760 67.3273 nan 0.0050 0.0001
## 2780 67.3252 nan 0.0050 0.0000
## 2800 67.3231 nan 0.0050 0.0001
## 2820 67.3206 nan 0.0050 0.0001
## 2840 67.3188 nan 0.0050 0.0000
## 2860 67.3168 nan 0.0050 0.0000
## 2880 67.3148 nan 0.0050 0.0001
## 2900 67.3125 nan 0.0050 0.0000
## 2920 67.3107 nan 0.0050 0.0001
## 2940 67.3094 nan 0.0050 0.0000
## 2960 67.3077 nan 0.0050 0.0001
## 2980 67.3057 nan 0.0050 0.0000
## 3000 67.3040 nan 0.0050 0.0001
```

```r
# check performance using an out-of-bag estimator
# OOB underestimates the optimal number of iterations
best.iter <- gbm.perf(gbm2,method="OOB")
```

```
## Warning in gbm.perf(gbm2, method = "OOB"): OOB generally underestimates the
## optimal number of iterations although predictive performance is reasonably
## competitive. Using cv.folds>0 when calling gbm usually results in improved
## predictive performance.
```




```r
print(best.iter)
```

```
## [1] 3000
```

```r
data.predict = predict(gbm2, n.trees = best.iter, newdata = test)
# Confusion matrix
conf_matrix <- table(data.predict, Cause39)
```

Accuracy of model and SSE

```r
#Accuracy
sum(diag(conf_matrix)) / nrow(test)
```

```
## [1] 8.094977e-05
```

```r
# SSE
SSE = sum((Cause39 - data.predict)^2)
print(SSE)
```

```
## [1] 65365605
```