ProblemSet4_4

Name: Zhen Qin, Uniqname: qinzhen, UMID: 48800866, Dept: Statistics

Datasets are from https://github.com/jensqin/Stat506 (https://github.com/jensqin/Stat506) .

a.

From the documentation, I know that I should use option family to indicate the distribution and link function.

```
# Install the package mgcv
# install.packages("mgcv")
library(mgcv)
library(doParallel)

# load a simulated data frame containing two columns
# setwd("E:/UM academy/stat 506/ProblemSet4")
load('ps4_q4.RData')

# use logistic regression to predict ‘y’
# link function is logit function
# distribution is binomial distribution
fit=gam(y~x,data=sample_data,family=binomial(link=logit))
yfit = fit$fitted.values
head(yfit)
```

```
## [1] 0.7031135 0.7024760 0.7047218 0.7025165 0.7055138 0.7049088
```

b/c.

Write a function xvalidate with arguments folds and cores to estimate the out-of-sample prediction error which is MSE.

```r
## Write a function xvalidate
xvalidate = function(folds, cores){
  load('ps4_q4.RData')
  ## Randomly shuffle the data
  n = nrow(sample_data)
  rands = sample_data[sample(n),]

  ## Create equally size folds
  kfolds = cut(seq(1, n),breaks=folds,labels=FALSE)

  # Perform k fold cross validation
  ymse = function(k){
    testIndexes <- which(kfolds==k,arr.ind=TRUE)
    testData <- sample_data[testIndexes, ]
    trainData <- sample_data[-testIndexes, ]

    ## fit the model
    fit=mgcv::gam(y~x,data=trainData,family=binomial(link=logit))
    ypred = predict(fit,newdata = testData)
    return(mean((ypred-testData$y)^2))
  }

  ## check if computing in parallel is needed
  if(cores%%1 == 0 & cores > 1){
    cl = makeCluster(cores)
    registerDoParallel(cl)

    #Perform k fold cross validation in parallel
    results = foreach(case = 1:folds) %dopar% {
      ##do do with case #
      ymse(case)
    }
    stopCluster(cl)
  }else if(cores == 1){
    results = list()

    #Perform k fold cross validation locally
    for(case in 1:folds){
      results[case]=ymse(case)
    }
  }else{
    return(0)
  }
  return(results)
}

# test the function
mean(unlist(xvalidate(10,1)))
```

```
## [1] 0.2355297
```

```r
mean(unlist(xvalidate(10,2)))
```

```
## [1] 0.2355297
```

d.

```
#!/bin/sh
##This names the job for the queueing system
#PBS -N fitmodel

##This denotes the queue that the job should be run in.
#PBS -A stats_flux
#PBS -l qos=flux,feature=haswell
#PBS -q flux

##This denotes the number of nodes and processors that the job should be run on.
#PBS -l nodes=1:ppn=1
##For embarrassingly parallel computing, instead select the number of processors
##PBS -l procs=8
##PBS -t 0

##This is the run time (hh:mm:ss) that your job will be allocated.
#PBS -l walltime=0:15:00

##Where does your program's STDOUT go? (Replace with your uniquemane)
#PBS -o /home/qinzhen/R

##Import the shell's environment
## This is important if you're using Environment Modules (i.e. module load ...)
#PBS -V

##In what circumstances should an email be sent regarding this job?  'a' is for aborted jobs,
## 'b' is when the job starts, and 'e' is when the job exits.
#PBS -m abe

##Where should email be sent when the job starts and stops? REPLACE with your uniquename.
##PBS -M qinzhen@umich.edu

#PBS -l pmem=4Gb

#PBS -j oe

#PBS -V

##code to be run
R CMD BATCH --vanilla /home/qinzhen/R/fit.R /home/qinzhen/R/cv.Rout
```

Set cores = 2. Submit the PBS script cv.pbs, then get cv.Rout. The MSE is 0.2355297. elapsed time is 5.816s.

> mean(unlist(xvalidate(10,2))) [1] 0.2355297
>
> proc.time()

> user system elapsed

> 1.225 0.096 5.816

e.

```
#!/bin/sh
##This names the job for the queueing system
#PBS -N fitmodel

##This denotes the queue that the job should be run in.
#PBS -A stats_flux
#PBS -l qos=flux,feature=haswell
#PBS -q flux

##This denotes the number of nodes and processors that the job should be run on.
#PBS -l nodes=1:ppn=1
##For embarrassingly parallel computing, instead select the number of processors
##PBS -l procs=8
##PBS -t 0

##This is the run time (hh:mm:ss) that your job will be allocated.
#PBS -l walltime=3:00:00

##Where does your program's STDOUT go? (Replace with your uniquemane)
#PBS -o /home/qinzhen/R

##Import the shell's environment
## This is important if you're using Environment Modules (i.e. module load ...)
#PBS -V

##In what circumstances should an email be sent regarding this job?  'a' is for aborted jobs,
## 'b' is when the job starts, and 'e' is when the job exits.
#PBS -m abe

##Where should email be sent when the job starts and stops? REPLACE with your uniquename.
##PBS -M qinzhen@umich.edu

#PBS -l pmem=4Gb

#PBS -j oe

#PBS -V

##code to be run
R CMD BATCH --vanilla /home/qinzhen/R/fit10.R /home/qinzhen/R/cv10.Rout
R CMD BATCH --vanilla /home/qinzhen/R/fit100.R /home/qinzhen/R/cv100.Rout
R CMD BATCH --vanilla /home/qinzhen/R/fit1000.R /home/qinzhen/R/cv1000.Rout
R CMD BATCH --vanilla /home/qinzhen/R/fit10000.R /home/qinzhen/R/cv10000.Rout
```

Submit the PBS script multicv.pbs, then get cv10.Rout, cv100.Rout, cv1000.Rout, cv10000.Rout.

For k = 10:

> mean(unlist(xvalidate(10,8))) [1] 0.2355297
>
> proc.time()

```
user system elapsed
```

```
1.263 0.119 14.027
```

For k = 100:

```
mean(unlist(xvalidate(100,8))) [1] 0.2352789

proc.time()
```

```
user system elapsed
```

```
1.422 0.117 23.044
```

For k = 1000:

```
mean(unlist(xvalidate(1000,8))) [1] 0.2353165

proc.time()
```

```
user system elapsed
```

```
1.792 0.180 105.663
```

For k = 10000:

```
mean(unlist(xvalidate(10000,8))) [1] 0.2353015

proc.time()
```

```
user system elapsed
```

```
5.610 0.560 917.207
```

In conclusion, the number of folds does not have a meaningful impact on the estimated prediction error. If the number of folds increases, the run time increases as well.