Lab04Learn

2024-03-20

Lab04 learning

This is an Learning page for lab04 cross validation

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(leaps)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
sgemm <-read.csv("./data/sgemm_product.csv")

set.seed(47)

#sample 500 index of rows
my_sample <- sample(1:nrow(sgemm),500)

#add a column name logrun1 which is log of run1_ms
#get rid of the run1_ms avoid the depulication and
#useless run2_ms,run3_ms,run4_ms
sgemm1 <- sgemm %>% janitor::clean_names() %>%
        mutate(logrun1 = log(run1_ms)) %>%
        select(-run1_ms,-run2_ms,-run3_ms,-run4_ms)


#get rows according the index vectors
sgemm1 <- sgemm1[my_sample,]

#construct a data frame with loggrun1 and all the intersections of x columns.
#using to find the possible relations between the 2 degree of intersections of x with the prediction
```

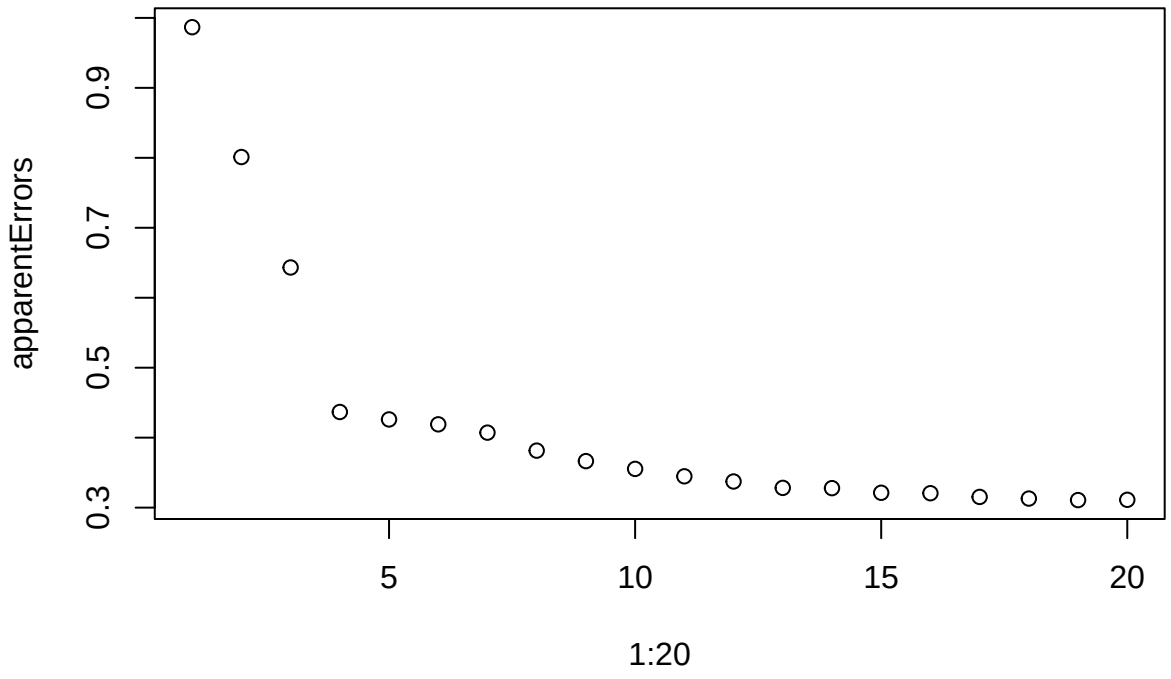
```r
mf <- model.frame(logrun1~.^2,data=sgemm1)

#construct a matrix for regsubset, without the Y.
X <- model.matrix(logrun1~.^2,mf)[,-1]
y <- sgemm1$logrun1

#stepwise regression and best subset selection
subset1 <- regsubsets(x=X,y=y,nvmax=20,method = 'backward')

subset1.summ <- summary(subset1)

apparentErrors <- subset1.summ$rss/(500-1:20)
plot(1:20,apparentErrors)
```




```r
allyhat<-function(xtrain, ytrain, xtest,lambdas,nvmax=50){

  n<-nrow(xtrain)
  yhat<-matrix(nrow=nrow(xtest),ncol=length(lambdas))

  search<-regsubsets(xtrain,ytrain, nvmax=nvmax, method="back")
  summ<-summary(search)
  for(i in 1:length(lambdas)){
    penMSE<- n*log(summ$rss)+lambdas[i]*(1:nvmax)
    best<-which.min(penMSE)  #lowest AIC
    betahat<-coef(search, best) #coefficients
    xinmodel<-cbind(1,xtest)[,summ$which[best,]] #predictors in that model
    yhat[,i]<-xinmodel%*%betahat
  }
  yhat
}
```

```r
lambdas <- c(2,4,6,8,10,12)
n <- nrow(X)
folds <- sample(rep(1:10, length.out=n))
fitted <- matrix(nrow=n,ncol=length(lambdas))
for(i in 1:10){
  train <- (1:n)[folds != i] # indices for train
  test <- (1:n)[folds == i] # indices for test
  fitted[test,] <- allyhat(X[train,], y[train],
                           X[test,], lambdas)
}
mspe_cv <- colMeans((y-fitted)^2)

#picking \lambdas = 8

#task 4
logrun2 <- log(sgemm[my_sample,]$Run2..ms.)
best_lambda <- 8

search <- regsubsets(X, y, nvmax = 20, method = 'backward')
summ <- summary(search)
# penalised_rss
penalised_rss <- 500*log(summ$rss) + best_lambda*(1:20)
best_mod <- which.min(penalised_rss)

best_mod
```

```
## [1] 13
```

```r
# picking the best (i.e. min penalised_rss)
beta_hat <- coef(search, best_mod)
beta_hat # coefficients
```

```
##   (Intercept)           mwg           nwg           vwn       mwg:nwg
##  3.5730413590  0.0184889650  0.0104711240 -0.2081554660  0.0001425587
##     mwg:mdimc     mwg:ndimc     nwg:mdimc     nwg:ndimc       nwg:vwn
## -0.0005652439 -0.0006172698 -0.0005026225 -0.0005794572  0.0015073559
##     kwg:ndimc   mdimc:ndimc      ndimc:sa       strm:sb
##  0.0007558822  0.0024539939 -0.0132534560 -0.2501747383
```

```r
# organise X matrix for prediction
# the matrix includes an intercept term 1 and only the variables selected by the best model, which can
Xpred <- cbind(1, X)[, summ$which[best_mod,]]
colnames(Xpred) # varibles been picked = beta_hat names
```

```
##  [1] ""            "mwg"         "nwg"         "vwn"         "mwg:nwg"
##  [6] "mwg:mdimc"   "mwg:ndimc"   "nwg:mdimc"   "nwg:ndimc"   "nwg:vwn"
## [11] "kwg:ndimc"   "mdimc:ndimc" "ndimc:sa"    "strm:sb"
```

```r
# generate prediction, i.e. y_hat
y_hat <- Xpred %*% beta_hat # y_hat = X * beta_hat

# calculate mspe_sample
mspe_sample <- sum((logrun2 - y_hat)^2)/length(y_hat)
mspe_sample
```

```
## [1] 0.3190249
```

```
sgemm <- sgemm %>% janitor::clean_names() %>%
        mutate(logrun2 = log(run2_ms)) %>%
        select(-run1_ms,-run2_ms,-run3_ms,-run4_ms)

mf <- model.frame(logrun2~.^2,data=sgemm)

#construct a matrix for regsubset, without the Y.
X <- model.matrix(logrun2~.^2,mf)[,-1]
y <- sgemm$logrun2


fullPred <- cbind(1, X)[, summ$which[best_mod,]]
colnames(fullPred)
```

```
##  [1] ""            "mwg"         "nwg"         "vwn"         "mwg:nwg"
##  [6] "mwg:mdimc"   "mwg:ndimc"   "nwg:mdimc"   "nwg:ndimc"   "nwg:vwn"
## [11] "kwg:ndimc"   "mdimc:ndimc" "ndimc:sa"    "strm:sb"
```

```
y_hat_full <- fullPred %*% beta_hat # y_hat = X * beta_hat

mspe_sample <- sum((y - y_hat_full)^2)/length(y_hat_full)
mspe_sample
```

```
## [1] 0.3600138
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.