Model Selection

Yu Yang

8/3/2020

1. step() method

Given a large linear model, select the best subset.

```
library(alr4)
data(BGSall)
m<-lm(WT18~Sex+WT2+HT2+WT9+HT9+LG9+ST9+HT18+LG18+ST18+Soma,data=BGSall)
m<-lm(WT18~.,data=BGSall) # same as above
summary(m)
```

1.1 forward, backward, both selection under AIC

```
# forward selection, start from the smallest model
step(lm(WT18~1,data=BGSall),
     scope=list(upper=~Sex+WT2+HT2+WT9+HT9+LG9+ST9+HT18+LG18+ST18+Soma,lower=~1),
     direction="forward")
# backward deletion
step(m, direction = 'backward')
# both direction
step(m,scope=list(upper=~.,lower=~1), direction = 'both')
```

1.2 selection under BIC

```
n <- length(BGSall$WT18)
step(m,k=log(n))
```

2. {leaps} package

For {leaps}, we need to first get the design matrix.

2.1 leaps()

For each dimension, it will search `nbest` best models.

```
library(leaps)
attach(BGSall)
# obtain design matrix
x<-cbind(Sex,WT2,HT2,WT9,HT9,LG9,ST9,HT18,LG18,ST18,Soma)

print(outs<-leaps(x,WT18, nbest = 3))
plot(outs$size, outs$Cp, log = "y", xlab = "p", ylab = expression(C[p]))
```

```
(outs <- leaps(x,WT18,nbest=1))
which.min(outs$Cp)
```

2.2 regsubsets()

- For each dimension, it will search the best model, by default.
- `method=c("exhaustive", "backward", "forward", "seqrep")`. Different method may give different results, and we can try all of these options in order to find the best one if time allows. Otherwise, we can just use the default option.

```
reg.all<-regsubsets(x,WT18,nvmax=11)

reg.summary<-summary(reg.all)

names(reg.summary)

reg.summary$which

# check which dimension we should choose
par(mfrow=c(2,2))
plot(reg.summary$rsq ,xlab="Model Size",ylab="R^2",
     type="l")
plot(reg.summary$adjr2 ,xlab="Model Size ",
     ylab="Adjusted R^2",type="l")
plot(reg.summary$cp ,xlab="Model Size",ylab="Cp",
     type="l")
plot(reg.summary$bic ,xlab="Model Size ",
     ylab="BIC",type="l")
par(mfrow=c(1,1))

which.min(reg.summary$bic)
which.min(reg.summary$cp)
which.max(reg.summary$adjr2)

# choose the optimal one

indi <- reg.summary$which[8,]
fit <- glm(WT18 ~ x[,indi[-1]])

summary(fit)
coef(reg.all,id=8)

# Calculate AIC
fit$aic
extractAIC(fit, k=2)

# Calculate AICc
npar <- fit$df.null - fit$df.residual + 1
n <- length(WT18)
extractAIC(fit3b,k=2)+2*npar*(npar+1)/(n-npar-1)

# Calculate BIC
extractAIC(fit3b,k=log(n))
```

```r
detach(BGSall)
```

LASSO and Ridge Regression

```r
library(glmnet)
library(ncvreg) # for SCAD and MCP
library(MASS)

data.generation<-function(n,p,intercept=0,beta,rho,sig){
  x<-matrix(0,n,p)
  Sigma <- matrix(rep(rho,p*p),byr=T,ncol=p) # set all predictors to have the same correlation
  diag(Sigma)<-rep(1,p) # compound symmatry correlation
  x<-mvrnorm(n,rep(0,p),Sigma)
  y<-intercept+x%*%beta+sig*rnorm(n)
  list(x=x,y=y)
}



p<-200
n<-50
rho<-0.5
sig<-1
beta<-c(rep(1,7),rep(0,193)) # only the first 7 are non-zero

dat<-data.generation(n,p,intercept=0,beta,rho,sig)
x=dat$x; y=dat$y

grid=10^seq(10,-3,length=100) # specify lambda values to be considered for

ridge.mod=glmnet(x,y,alpha=0,lambda=grid)  # alpha = 0 does ridge regression, alpha = 1 is lasso.

ridge.mod=glmnet(x,y,alpha=0)  # defaut choice for lambda
# plot(ridge.mod,type.coef="2norm")  #If type.coef="2norm" then a single curve per variable, else if ty
#   a coefficient plot per response. The command does not seem to work, with a few warnings.
plot(ridge.mod,xvar="norm") # the xlab should be L2 norm for ridge regression, but due to the package d
# xlab is L1 norm. Should pay attention to this.
# Ridge regression does no variable selection, so we could notice the number of variables remains 200 i
plot(ridge.mod,xvar="lambda",label=TRUE)   # choice for xvar: "norm", "lambda", "dev" (the percent devi
plot(ridge.mod)
cv.out=cv.glmnet(x,y,alpha=0)  # default is 10 fold. Can change using, e.g., nfold=5
cv.glmnet(x,y,alpha=0,nfold=5)
cv.out$lambda.min   # best lambda, one may also use lambda.1se (largest lambda within 1 sd of the small
cv.out$lambda.1se   # a larger cv would lead to a simpler model
plot(cv.out)    # all the 200 predictors are involved at the different lambda values
ridge.pred=predict(ridge.mod,s=cv.out$lambda.min,newx=x)
predict(ridge.mod,type="coefficients",s=cv.out$lambda.min) # Look at the estimated coefficients
predict(ridge.mod,s=cv.out$lambda.min,newx=x[1,,drop=FALSE]) # predict for case 1

lasso.mod=glmnet(x,y,alpha=1,lambda=grid)
plot(lasso.mod)    # the numbers on top indicate how many predictos have non-zero coefficients
plot(lasso.mod,xvar="lambda",label=TRUE)
```

```
lasso.mod=glmnet(x,y,alpha=1)
cv.out.l=cv.glmnet(x,y,alpha=1)
plot(cv.out.l)
cv.out.l$lambda.min
lasso.pred=predict(lasso.mod,s=cv.out.l$lambda.min,newx=x)
coeff.lasso=predict(lasso.mod,type="coefficients",s=cv.out.l$lambda.min)
which(abs(coeff.lasso)>0)

mcp.mod <- cv.ncvreg(x,y,family='gaussian',penalty='MCP',nfolds=5)
pre.mcp <- predict(mcp.mod,X=x[1,,drop=FALSE],lambda=mcp.mod$lambda.min)
mcp.fit=mcp.mod$fit
coeff.mcp=mcp.fit$beta[,mcp.mod$min]
which(abs(coeff.mcp)>0)

scad.mod <- cv.ncvreg(x,y,family='gaussian',penalty='SCAD',nfolds=5)
pre.scad <- predict(scad.mod,X=x[1,,drop=FALSE],lambda=scad.mod$lambda.min)
scad.fit=scad.mod$fit
coeff.scad=scad.fit$beta[,scad.mod$min]
which(abs(coeff.scad)>0)



cv.compare<-function(x,y,n2,r){
  n<-nrow(x)
  err.r<-rep(0,r)
  err.l<-rep(0,r)
  err.m<-rep(0,r)
  err.s<-rep(0,r)
  for (i in 1:r) {
    train=sample(1:n, n-n2)
    test=(-train)
    y.test=y[test]

    ridge.mod=glmnet(x[train,],y[train],alpha=0)
    cv.out=cv.glmnet(x[train,],y[train],alpha=0)
    ridge.pred=predict(ridge.mod,s=cv.out$lambda.min,newx=x[test,])
    err.r[i]=mean((ridge.pred-y.test)^2)

    lasso.mod=glmnet(x[train,],y[train],alpha=1)
    cv.out.l=cv.glmnet(x[train,],y[train],alpha=1)
    lasso.pred=predict(lasso.mod,s=cv.out.l$lambda.min,newx=x[test,])
    err.l[i]=mean((lasso.pred-y.test)^2)

    mcp.mod <- cv.ncvreg(x[train,],y[train],family='gaussian',penalty='MCP',nfolds=5)
    mcp.pred <- predict(mcp.mod,X=x[test,],lambda=mcp.mod$lambda.min)
    err.m[i]=mean((mcp.pred-y.test)^2)

    scad.mod <- cv.ncvreg(x[train,],y[train],family='gaussian',penalty='SCAD',nfolds=5)
    scad.pred <- predict(scad.mod,X=x[test,],lambda=scad.mod$lambda.min)
    err.s[i]=mean((scad.pred-y.test)^2)
  }
  par(mfrow=c(2,2))
  plot(err.r,err.l,xlab="Ridge MSPE",ylab="Lasso MSPE")
```

```
  abline(0,1)
  plot(err.m,err.l,xlab="MCP MSPE",ylab="Lasso MSPE")
  abline(0,1)
  plot(err.s,err.l,xlab="SCAD MSPE",ylab="Lasso MSPE")
  abline(0,1)
  plot(err.s,err.m,xlab="SCAD MSPE",ylab="MCP MSPE")
  abline(0,1)


  c(ridge.err=mean(err.r),ridge.sd=sd(err.r),lasso.err=mean(err.l),lasso.sd=sd(err.l), mcp.err=mean(err
}

cv.compare(x,y,15,50)

beta<-c(1/seq(1:15),rep(0,185))
dat<-data.generation(n,p,intercept=0,beta,rho,sig)
x=dat$x; y=dat$y

cv.compare(x,y,15,50)
```