ridge_lasso_glmnet.R

vikaskamath

2021-03-20

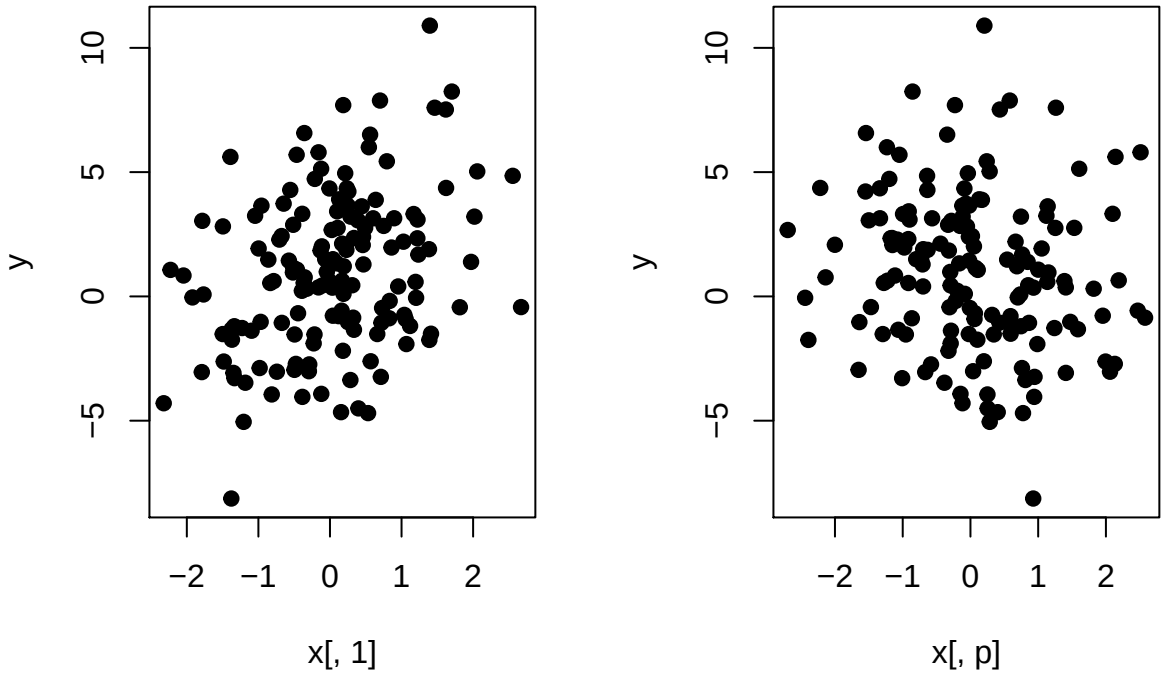
```
### The most prominent package to perform ridge, lasso, and elastic net is glmnet.
### This package is very fast and reliable and is written by the creators of lasso
###  and elastic net.

# install.packages("glmnet")
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

```
set.seed(1000)

### We are going to have 50 predictors of which the last z have zero for coefficients
###  (so they are not really predictor) and n rows of data.  However, we will use about
###  m = n/2 of the data for assessing prediction accuracy.  A better way is to cross validate but
###  putting a separate test data aside is more convenient for demo purposes

p = 50
n = 150
m = floor(n/2)
z = 20

### Sigma is the amount of "noise" in data.

sigma = 1

### X is our data matrix.  Data are generated from a Gaussian
###  distribution with mean 0 and sd = sigma

x = matrix( rnorm(p*n, sd = sigma), nrow = n, ncol = p)

### I just generate the coefficients at random from uniform(0,1).
### Note z of them will be set to zero.

b = c(runif(p-z), rep(0,z))

### Here are our y's. The +1 adds an intercept of 1.

y = x %*% b + rnorm(n) + 1
```

```
### Let's just look at y versus first predictor and last one:

par(mfrow=c(1,2), pch = 19)
plot(y ~ x[,1])
plot(y ~ x[,p])
```




```
par(mfrow=c(1,1))

### Split the data into train and test

train_index = 1:m
train   = x[ train_index,]
test    = x[-train_index,]
y_train = y[train_index]
y_test  = y[-train_index]

###
### Next, we create three models: ridge, lasso, and ols.
### For now I will just use the defaults.
### cv.glmnet automatically performs cross validation to
###  to pick the lamda.  It does the scaling automatically too.

### Extracting the coef from the ridge and lasso requires specifying
###  the value of lambda at which the lowest cv happened.
```

```
ridge_model = cv.glmnet(x = train, y = y_train, alpha = 0)
ridge_model$lambda.min
```

```
## [1] 0.1536589
```

```
ridge_coef  = coef(ridge_model, s = ridge_model$lambda.min)

lasso_model = cv.glmnet(x = train, y = y_train, alpha = 1)
lasso_model$lambda.min
```

```
## [1] 0.03388379
```

```
lasso_coef  = coef(lasso_model, s = lasso_model$lambda.min)

ols_model   = lm(y_train ~ train)
ols_coef    = coef(ols_model)

### Let's look at the results:

results = as.matrix(cbind(c(1,b), ols_coef, as.matrix(ridge_coef), as.matrix(lasso_coef)))
colnames(results) = c("true","ols", "ridge", "lasso")

head(round(results,4))
```

```
##               true    ols  ridge  lasso
## (Intercept) 1.0000 1.0290 0.9730 0.9472
## train1      0.6675 0.7273 0.7068 0.7104
## train2      0.7350 0.5215 0.4873 0.4422
## train3      0.7381 0.9485 0.8819 0.9045
## train4      0.3385 0.6426 0.5169 0.4732
## train5      0.4234 0.8325 0.7155 0.6421
```

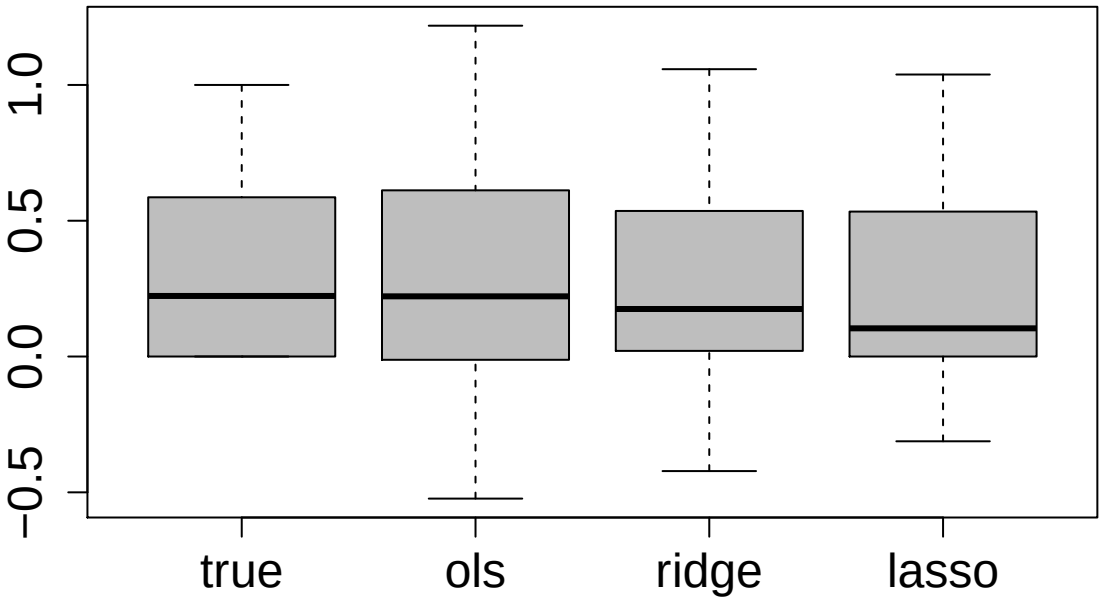
```
tail(round(results,4))
```

```
##         true     ols   ridge   lasso
## train45    0  0.1664  0.1402  0.0782
## train46    0  0.0745  0.0630  0.0000
## train47    0 -0.1908 -0.0952 -0.0231
## train48    0  0.1991  0.1932  0.1452
## train49    0 -0.0476 -0.0185 -0.0176
## train50    0 -0.0133 -0.0486 -0.0069
```

```
summary(results)
```

```
##       true             ols                ridge              lasso        
##  Min.   :0.0000   Min.   :-0.52307   Min.   :-0.42174   Min.   :-0.3121  
##  1st Qu.:0.0000   1st Qu.:-0.01207   1st Qu.: 0.02076   1st Qu.: 0.0000  
##  Median :0.2230   Median : 0.22165   Median : 0.17501   Median : 0.1039  
##  Mean   :0.3030   Mean   : 0.31283   Mean   : 0.28982   Mean   : 0.2761  
##  3rd Qu.:0.5863   3rd Qu.: 0.61189   3rd Qu.: 0.53605   3rd Qu.: 0.5337  
##  Max.   :1.0000   Max.   : 1.21809   Max.   : 1.05798   Max.   : 1.0383
```

```
boxplot(results, pch = 19, cex.lab = 1.5, cex.axis = 1.5, col = "grey" )
```




```
### Predict on the test data next.
### By the way, you could use predict for lm but it expects a data frame and not a matrix.

rmse = function(a,b) { sqrt(mean((a-b)^2)) }

ols_rmse   = rmse(y_test , cbind(1,test) %*% ols_coef )
ridge_rmse = rmse(y_test , predict(ridge_model, newx = test, s = ridge_model$lambda.min))
lasso_rmse = rmse(y_test , predict(lasso_model, newx = test, s = lasso_model$lambda.min))

ols_rmse
```

```
## [1] 1.802231
```

```
ridge_rmse
```

```
## [1] 1.547671
```

```
lasso_rmse
```

```
## [1] 1.46193
```

```
### What I did favored lasso
```