Problem Set 3 – Ridge and Lasso Regression

2016-04-06

Simulating Data

To simulate data we must draw random variables with some prespecified distribution. In *R* for every distribution usually four functions are implemented which are useful for working with distributions and differ by their prefix: *r* (random), *d* (density), *p* (probability), and *q* (quantile). The prefix is combined with a name for the distribution, e.g. *norm* for the normal distribution: *dnorm* for the density of a normal distribution, *pnorm* for the probability, *qnorm* for the quantiles, and *rnorm* to draw from a normal distribution. (Check out the help page of the functions!)

Here we want to simulate a linear model of the form

$$y_i = x_i'\beta + \varepsilon_i, i = 1, \dots, n$$

with β a p-dimensional coefficient vector and x_i p-dimensional vector of regressors. In vector notation:

$$y = X\beta + \varepsilon$$

with y and ε n-dimensional vectors and X a $n \times p$-design matrix.

Here the task is to simulate from this model, where we assume that the coefficient vector β has s entries equal to one and all others are zero.

- Set $n = 100$, $p = 10$, $s = 3$
- Create the coefficient vector β. Useful functions: *c()*, *rep()*
- Simulate a design matrix and the error. Useful functions: *matrix()*, *rnorm*
- Construct the model from above. Useful function: $\% * \%$ for matrix multiplication

```r
set.seed(12345)
n <- 100
p <- 10
s <- 3

beta <- c(rep(1,s), rep(0, p-s))
X <- matrix(rnorm(n*p), ncol=p); eps <- rnorm(n)
y <- X%*%beta + eps
```

Ridge Regression I

- Estimate a ridge regression on simulated data from Exercise 1. Useful function: *glmnet* from the package *glmnet* with default $\alpha = 0$. Also check out the option *lambda* in *glmnet* and the function *cv.glmnet* to perform cross-validation to determine λ.
- Simulate new data from the same model and make predictions both in- and out-of-sample. Calculate the MSE for the predictions (also for the in-sample fit). Useful function: *predict*
- Repeat the previous steps with different settings on n, p, and s.
- Compare the results with ols regression!

```r
library(glmnet)
ridge1 <- glmnet(X, y, alpha=0)
plot(ridge1)
# lambda.grid <- seq(2,0, by=0.05)
# ridge2 <- glmnet(X, y, lambda=lambda.grid, alpha=1)
cv.out <- cv.glmnet(X, y, alpha=0)
plot(cv.out)
bestlam <- cv.out$lambda.min
ridge.pred <- predict(ridge1, s= bestlam, newx=X)
MSE.ridge.ins <- mean((y-ridge.pred)^2)
# out of sample (size n/4)
Xnew <- matrix(rnorm(n/4*p), ncol=p); epsnew <- rnorm(n/4)
ynew <- Xnew%*%beta + epsnew
ridge.pred.out <-  predict(ridge1, s= bestlam, newx=Xnew)
MSE.ridge.out <- mean((ynew-ridge.pred.out)^2)

## now for ols

df <- data.frame(y=y,X=X)
ols1 <- lm(y~X, data=df)
MSE.ols.ins <- mean((y-predict(ols1))^2)
yhatnew <- cbind(1,Xnew)%*%coef(ols1)
MSE.ols.out <- mean((ynew - yhatnew)^2)


# compare coefficients
Coefs <- cbind(coef(ols1), as.vector(predict(ridge1, s=bestlam, newx=X, type="coefficients")))
colnames(Coefs) <- c("OLS", "Ridge")
head(Coefs)
```

Lasso Estimation I

- Redo the calculations from above but with Lasso with varying n, p, and s. Hint: Set option α in *glmnet* to 1
- Compare the results! (In particular compare a "sparse" with a "dense" setting)
- The package *hdm* contains the function *rlasso* which determines the penalization parameter by some theoretical grounded method. Look up the function in the man pages and / or vignette and analyze now the data set using this function. Compare the results.

```r
lasso1 <- glmnet(X, y, alpha=1)
plot(lasso1)
# lambda.grid <- seq(2,0, by=0.05)
# ridge2 <- glmnet(X, y, lambda=lambda.grid, alpha=1)
cv.lasso.out <- cv.glmnet(X, y, alpha=1)
plot(cv.lasso.out)
bestlam <- cv.lasso.out$lambda.min
lasso.pred <- predict(lasso1, s= bestlam, newx=X)
MSE.lasso.ins <- mean((y-lasso.pred)^2)
lasso.pred.out <-  predict(lasso1, s= bestlam, newx=Xnew)
MSE.lasso.out <- mean((ynew-lasso.pred.out)^2)
```

```r
library(hdm)
lasso2 <- rlasso(y~X)
lasso2.pred.ins <- predict(lasso2)
MSE.lasso2.ins <-  mean((y-lasso2.pred.ins)^2)
lasso2.pred.out <- predict(lasso2, newdata=Xnew)
MSE.lasso2.out <-  mean((ynew-lasso2.pred.out)^2)
```

Bonus Excercise

- Write a loop around the programs above so that you can perform a small simulation study!