Lab 7: Ridge regression and lasso

Johnny Hong

Stat 154, Fall 2017

Introduction

In this lab we will use the dataset `Hitters` in the R packages `ISLR`. More specifically, you will use `Salary` as the response variable, and the rest of the variables in `Hitters` as the predictors. Our objective is to identify the model with the highest predictive power among a set of model candidates. The models in consideration are ordinary least squares (OLS), principal component regression (PCR), partial least squares regression (PLSR), ridge regression, and lasso.

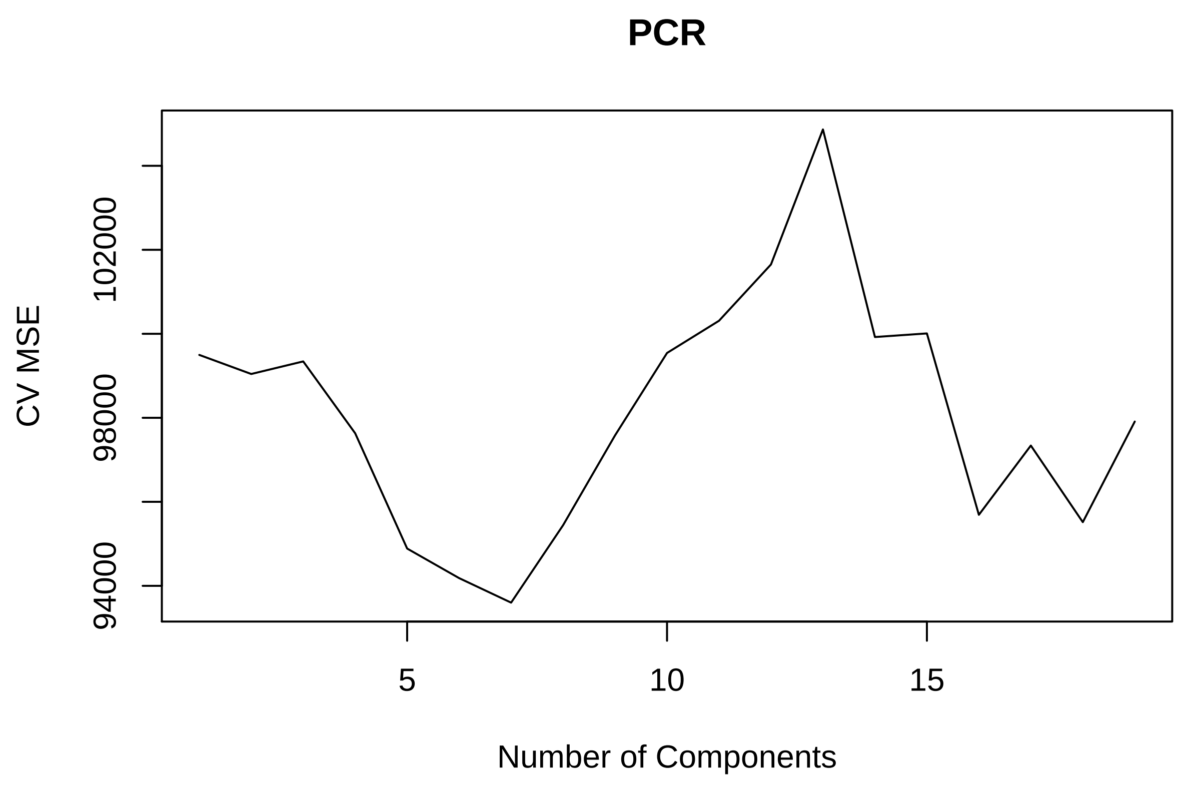
For simplicity, in this lab OLS means that we regress the response variable on all the other variables. Technically, choosing which variables to include is important, and there are many methods to do so such as forward stagewise selection and best subset selection, but we do not pursue these methods in this lab. Note that except for OLS, all the aforementioned methods require hyperparameter tuning. For PCR and PLSR, the hyperparameter is the number of components. For ridge regression and lasso, the hyperparameter is the regularization parameter. Traditionally hyperparameter tuning is performed via cross-validation with a grid search. For ridge regression and lasso, since the regularization parameter is a positive real number, grid search requires a discretization of the hyperparameter space.

Cross-validation for pcr() and plsr()

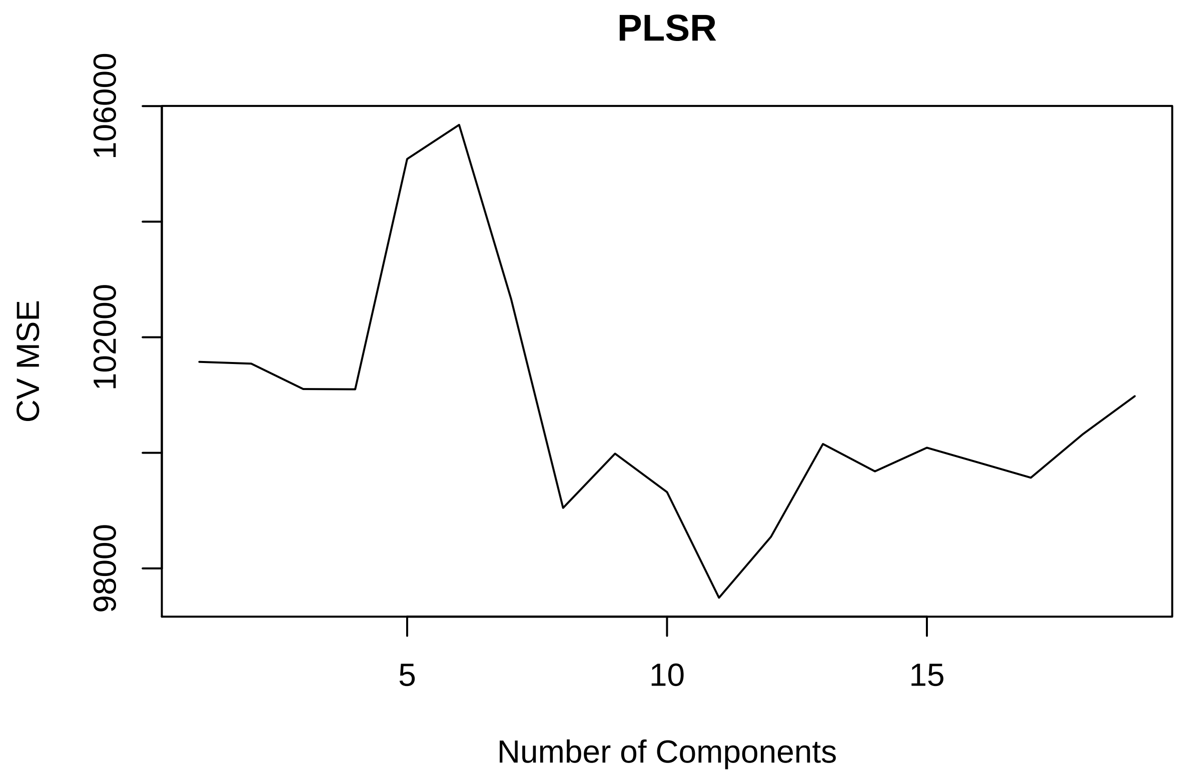
Cross validation is built in for `pcr()` and `plsr()`. For example, to do a 5-fold CV, use the argument `validation = "CV"` and `segments=5`. Here `segments` refers to the number of folds.

THe following shows how to do 10-fold CV for PCR and PLSR.

```r
n <- nrow(Hitters)
set.seed(100)
pcr_fit <- pcr(Salary ~ ., data = Hitters, scale = TRUE,
               validation = "CV", segments=10)
plot(pcr_fit$validation$PRESS[1, ] / n, type="l", main="PCR",
     xlab="Number of Components", ylab="CV MSE")
```



```r
set.seed(200)
plsr_fit <- plsr(Salary ~ ., data = Hitters, scale = TRUE,
                 validation = "CV", segments=10)
plot(plsr_fit$validation$PRESS[1, ] / n , type="l", main="PLSR",
     xlab="Number of Components", ylab="CV MSE")
```



- Based on the plots, how many components do you prefer for PCR? For PLSR?

Cross-validation for ridge regression and lasso

The function `glmnet()` in the R package `glmnet` is commonly used to fit ridge regression and lasso. However, `glmnet()` does not have an optional argument that allows us to do cross validation. We will use `cv.glmnet()` to perform CV for ridge regression and lasso.

`glmnet()` can fit a linear model with the *elastic net* penalty:

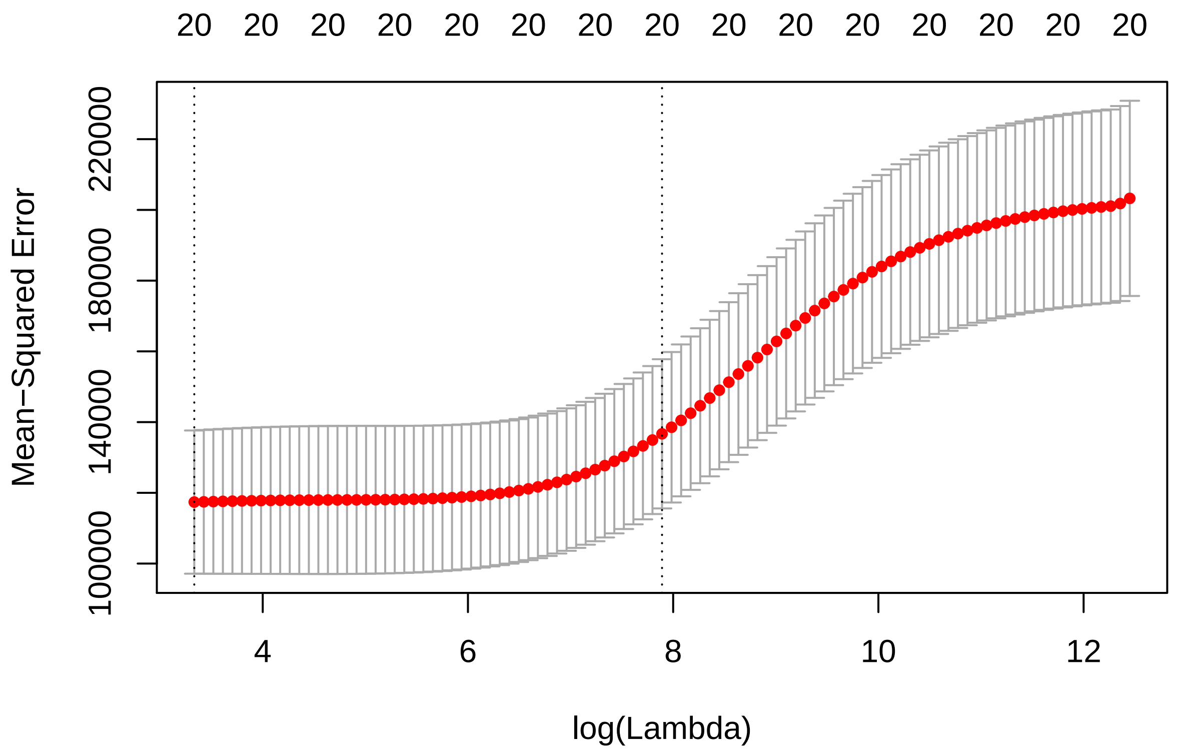
$$\frac{1-\alpha}{2}||\beta||_2^2 + \alpha||\beta||_1,$$

where $\alpha \in [0,1]$ is called the elastic net mixing parameter and β is the slope coefficient.

- If ridge regression is desired, what should be α?
- If lasso is desired, what should be α?
- Use `cv.glmnet()` to perform a 10-fold CV for ridge regression. Find out how to use `plot.cv.glmnet()` for visualization. Identify the optimal regularization parameter.
- Do the above for lasso.

```
set.seed(300)
# code for ridge regression CV
Hitters_noNA <- na.omit(Hitters)
X <- model.matrix(Salary ~ 0 + ., data=Hitters_noNA)
```

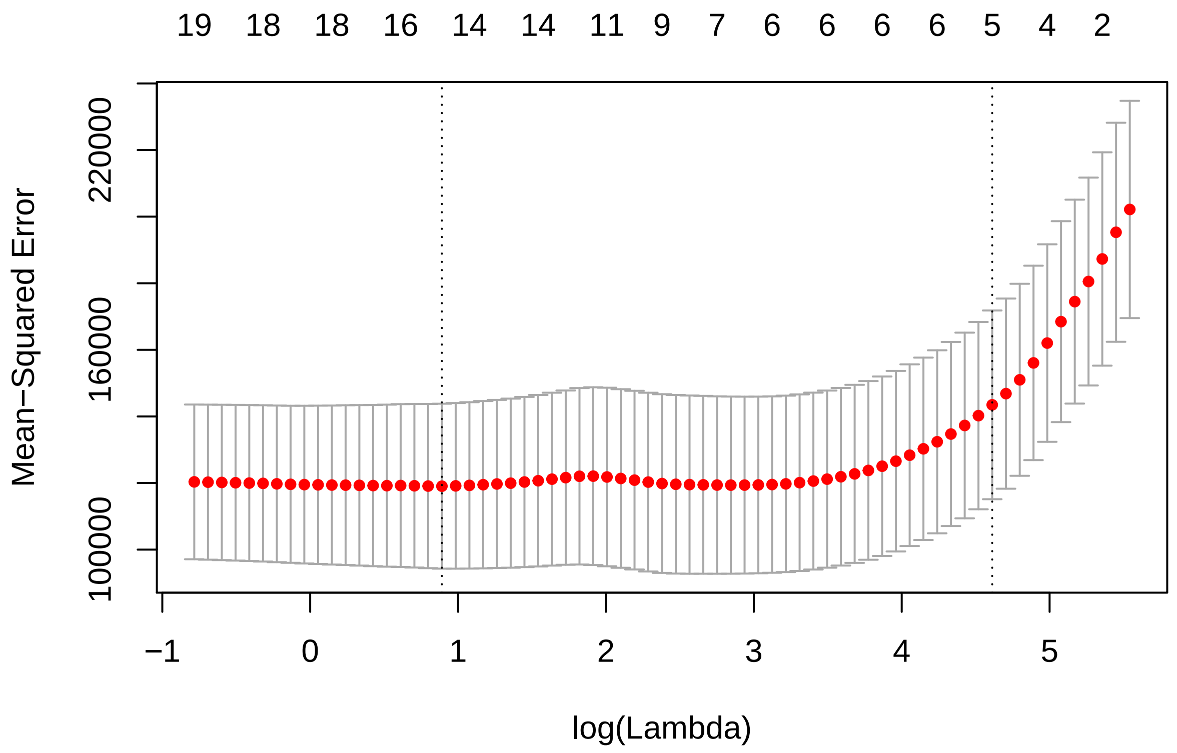
```
y <- Hitters_noNA$Salary
ridge_cv <- cv.glmnet(X, y, nfolds=10, alpha=0)
plot.cv.glmnet(ridge_cv)
```



```
ridge_cv$lambda.min
```

```
## [1] 28.01718
```

```
set.seed(400)
# code for lasso CV
lasso_cv <- cv.glmnet(X, y, nfolds=10, alpha=1)
plot.cv.glmnet(lasso_cv)
```



```r
lasso_cv$lambda.min
```

```
## [1] 2.436791
```

Nested Cross Validation

In order to make fair comparisons among different models, we should use a nested cross validation: an inner CV for parameter tuning and an outer CV for estimating the predictive power.

- Run a nested CV to estimate the predictive power for ordinary least squares (OLS), principal component regression (PCR), partial least squares regression (PLSR), ridge regression, and lasso. For hyperparameter tuning, use a 10-fold CV; for estimating predictive power, also use a 10-fold CV. As a reminder, OLS does not have any hyperparameter(s) for tuning in this lab assignment.
- Which model is the best?

```r
numOuterFolds <- 10
numInnerFolds <- 10
folds <- createFolds(Hitters_noNA$Salary, k=numOuterFolds)
test_MSEs <- matrix(nrow=5, ncol=numOuterFolds)
rownames(test_MSEs) <- c("OLS", "PCR", "PLS", "Ridge", "Lasso")
colnames(test_MSEs) <- paste0("Fold", 1:numOuterFolds)
for (i in 1:length(folds)) {
  fold <- folds[[i]]
  train_set <- Hitters_noNA[-fold, ]
  test_set <- Hitters_noNA[fold, ]
  # OLS
  ols_fit <- lm(Salary ~ ., data = train_set)
  ols_pred <- predict(ols_fit, test_set)
  # pcr
  pcr_fit <- pcr(Salary ~ ., data = train_set, scale = TRUE,
                 validation = "CV", segments=numInnerFolds)
  pcr_pred <- predict(pcr_fit, test_set, ncomp = which.min(pcr_fit$validation$PRESS))
  # pls
  pls_fit <- plsr(Salary ~ ., data = train_set, scale = TRUE,
                 validation = "CV", segments=numInnerFolds)
  pls_pred <- predict(pls_fit, test_set, ncomp = which.min(pls_fit$validation$PRESS))

  # ridge
  ridge_fit <- cv.glmnet(X, y, nfolds=numInnerFolds, alpha=0)
  ridge_pred <- predict(ridge_fit, newx=model.matrix(Salary ~ 0 + ., data=test_set),
                        s="lambda.min")
  # lasso
  lasso_fit <- cv.glmnet(X, y, nfolds=numInnerFolds, alpha=1)
  lasso_pred <- predict(lasso_fit, newx=model.matrix(Salary ~ 0 + ., data=test_set),
                        s="lambda.min")

  preds <- list(ols_pred, pcr_pred, pls_pred, ridge_pred, lasso_pred)

  test_MSEs[, i] <- sapply(preds, function(pred) {
     mean((test_set$Salary - pred)^2)
  })
}
test_MSEs
```

```
##           Fold1    Fold2    Fold3     Fold4    Fold5     Fold6    Fold7
## OLS   178815.82 59946.62 83572.54 114079.10 50017.48  94032.96 176201.1
## PCR   162907.65 68487.40 89780.43 163244.48 46729.85 106389.41 186396.1
## PLS   150371.09 59045.45 89107.95 166504.25 49663.74  85367.81 173978.8
## Ridge  97418.39 53791.04 76740.62 107355.78 33470.25  80741.88 142119.2
## Lasso  98135.91 61486.88 75279.61  89936.87 36085.68  73960.83 132900.3
##          Fold8    Fold9   Fold10
## OLS   69561.02 227600.5 171934.4
## PCR   70532.92 226579.3 160679.0
## PLS   67642.51 246605.2 158292.8
## Ridge 62382.71 221634.5 110959.1
## Lasso 62239.48 204696.2 119907.1
```

```r
cv_MSEs <- rowMeans(test_MSEs)
which.min(cv_MSEs)
```

```
## Lasso
##     5
```