P8106 Data Science Homework 1

Emil Hafeez

Contents

Section 1: Setup **1**
Libraries and Options 1
Prompt 1
Questions 2

Section 2: Implementation and Results **2**
Problem A) 2
Problem B) 3
Problem C) 4
Section (D) 5

Section 3: Comparison **7**
Problem E 7

Section 1: Setup

Libraries and Options

```
train_data = read_csv("./data/solubility_train.csv")
test_data = read_csv("./data/solubility_test.csv")
train_data = na.omit(train_data)
test_data = na.omit(test_data)
set.seed(1993)
```

Prompt

In this exercise, we will predict solubility of compounds using their chemical structures.

The training data are in the file "solubility train.csv" and the test data are in "solubility test.csv". Among the 228 predictors, 208 are binary variables that indicate the presence or absence of a particular chemical substructure, 16 are count features, such as the number of bonds or the number of bromine atoms, and 4 are continuous features, such as molecular weight or surface area. The response is in the column "Solubility" (the last column).

Questions

(A) Fit a linear model using least squares on the training data and calculate the mean squared error using the test data.

(B) Fit a ridge regression model on the training data, with chosen by cross-validation. Report the test error.

(C) Fit a lasso model on the training data, with chosen by cross-validation. Report the test error and the number of non-zero coefficient estimates in your model.

(D) Fit a principle component regression model on the training data, with M chosen by cross-validation. Report the test error and the value of M selected by cross-validation.

(E) Which model will you choose for predicting solubility?

Section 2: Implementation and Results

Results of MSE comparison appear at the end.

Problem A)

The dataframe consists of 229 variables. There are a large number of predictors and potential collinearity (implying large variance), but a linear model is implemented to start. While it would therefore be computationally unwise to compute all or best subsets and choose that way, this approach is included for sake of completeness.

Note that in performing cross-validation, you want to apply CV both when finding the predictors with the largest correlation with the response; and, you want to do it when fitting the model using only those predictors.

```
# summary(train_data)
```

```
regsubsets_obj = regsubsets(Solubility ~ ., data = train_data, method = "backward", nbest = 1)
# summary(regsubsetsObj)
plot(regsubsets_obj, scale = "bic")
```

We train the model using cross-validation.

```
set.seed(1993)
control_cv <- trainControl(method = "repeatedcv", number = 20, repeats = 5)

# lm(Solubility ~ ., data = train_data)

fit.lm <- train(Solubility~.,
          data = train_data,
          method = "lm",
          trControl = control_cv)

# fit.lm$finalModel
# I am aware of this error: thus far TAs are justifiedly unsure. Posted to discussion board.
```

Then, we apply the model to the test data and obtain the MSE for evaluation purposes.

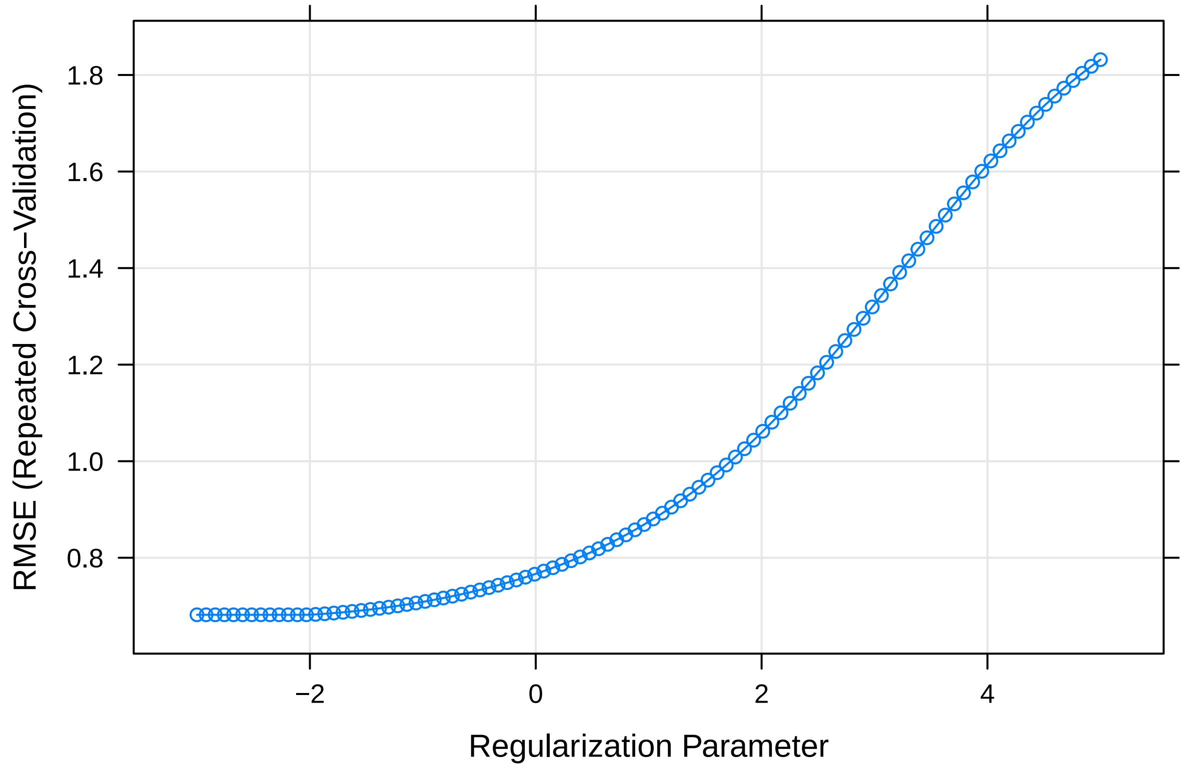
```
#Fitting training model on test set
pred.lm = predict(fit.lm, newdata = test_data)
rmse_lm = RMSE(pred.lm, test_data$Solubility)
rmse_lm
```

```
## [1] 0.7455802
```

Problem B)

Since regularization methods like ridge regression is not scale-invariant like LS methods (since ridge regression minimizes the objective function which includes a penalty term) so we must standardize our predictors first.

```
train_data_model_matrix <- model.matrix(Solubility ~ ., train_data)[,-1] #note that we drop the interce

control_object <- trainControl(method = "repeatedcv", number = 20, repeats = 5)

set.seed(1993)
ridge.fit <- train(train_data_model_matrix, train_data$Solubility,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(5, -3, length=100))),
                   # preProc = c(scale), glmnet does this by default
                   trControl = control_object)

plot(ridge.fit, xTrans = log) #note that this is on the log scale so the U shape is not visible
```



```r
#ridge.fit$bestTune
#coef(ridge.fit$finalModel, s = ridge.fit$bestTune$lambda)
```

The optimal value of lambda chosen by repeated cross-validation is 0.1312957.

Problem C)

(C) Fit a lasso model on the training data, with lambda chosen by cross-validation. Report the test error and the number of non-zero coefficient estimates in your model.

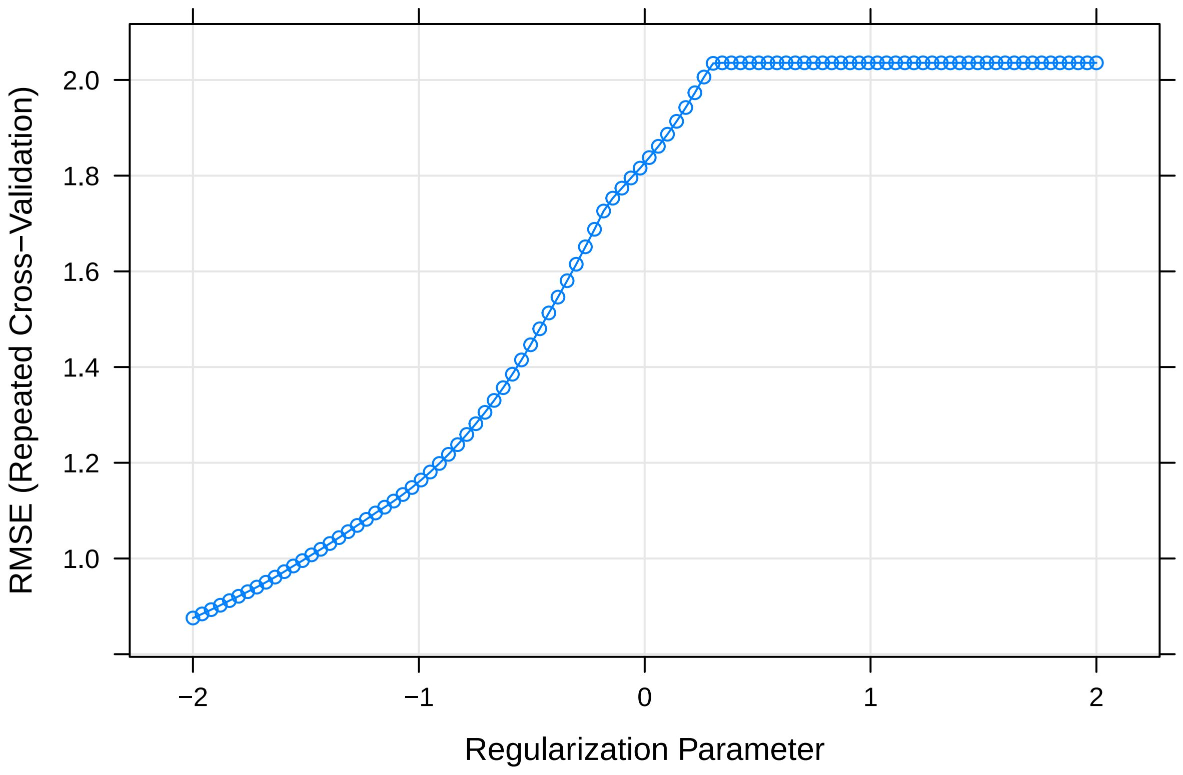
Given such a complex model, using a method like LASSO will provide useful variable selection properties that help with interpretability. The L1 penalty has the effect of forcing some of the coefficient estimates to be zero when lambda is sufficiently large. "Lasso is somewhat indifferent to very correlated predictors, and will tend to pick one and ignore the rest."

We must also remember to standardize the predictors here.

```r
set.seed(1993)
lasso.fit <- train(train_data_model_matrix, train_data$Solubility,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(2, -2, length=100))),
                   # preProc = c(scale), glmnet does this by default
                   trControl = control_object)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
plot(lasso.fit, xTrans = log)
```



```
#lasso.fit$bestTune$lambda
#lasso.fit$bestTune

non_zero_coefficients_lasso = coef(lasso.fit$finalModel, s = lasso.fit$bestTune$lambda)
non_zero_coefficients_lasso = as.data.frame(as.matrix(non_zero_coefficients_lasso))
length(which(non_zero_coefficients_lasso != 0))
```

```
## [1] 21
```

The optimal value of lambda chosen by repeated cross-validation is 0.1353353. The number of nonzero coefficient estimates is 21.

Section (D)

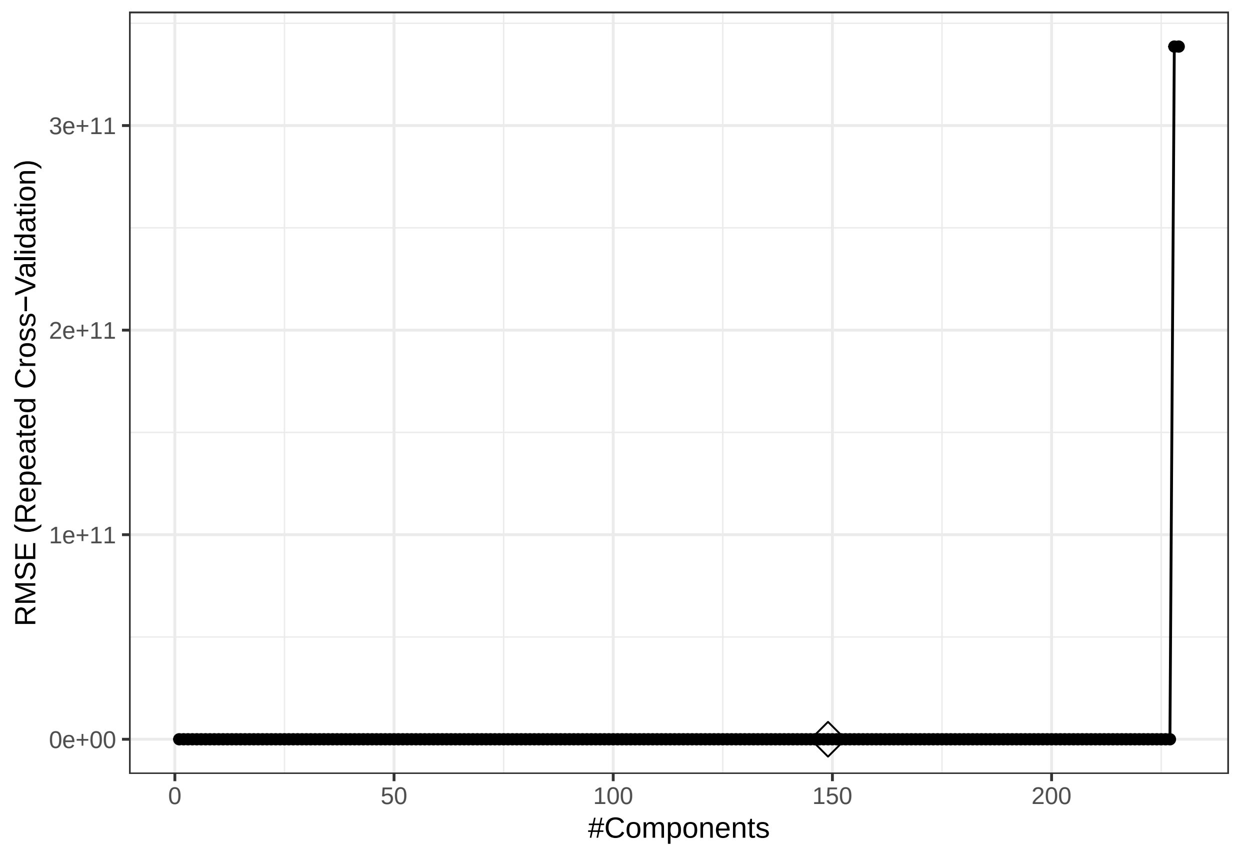
Fit a principle component regression model on the training data, with M chosen by cross-validation. Report the test error and the value of M selected by cross-validation.

M can be viewed as a tuning parameter, which we select using cross-validation. Since deriving the Z_m values us not dependent on the response variable, dimension reduction is unsupervised. We replace correlated original variables with the first M principal components that capture the joint variation. While the criteria is that each component contains as much information as possible while being uncorrelated with previous components, there is thus no guarantee that Z_m are the best linear combinations of X_j in predicting the response.

```r
control_pcr <- trainControl(method = "repeatedcv", number = 20, repeats = 5,
                            selectionFunction = "best")

set.seed(1993)

pcr.fit <- train(train_data_model_matrix, train_data$Solubility,
                 method = "pcr",
                 tuneGrid  = data.frame(ncomp = 1:229), #fine grid, but I want to test my computing pow
                 trControl = control_pcr,
                 preProcess = c("center", "scale"))

# test data
x2 <- model.matrix(Solubility~.,test_data)[,-1]
y2 <- test_data$Solubility

predy2.pcr2 <- predict.train(pcr.fit, newdata = x2) #Remember that this predict.train function automati
mean((y2 - predy2.pcr2)^2)
```

```
## [1] 0.540555
```

```r
ggplot(pcr.fit, highlight = TRUE) + theme_bw()
```



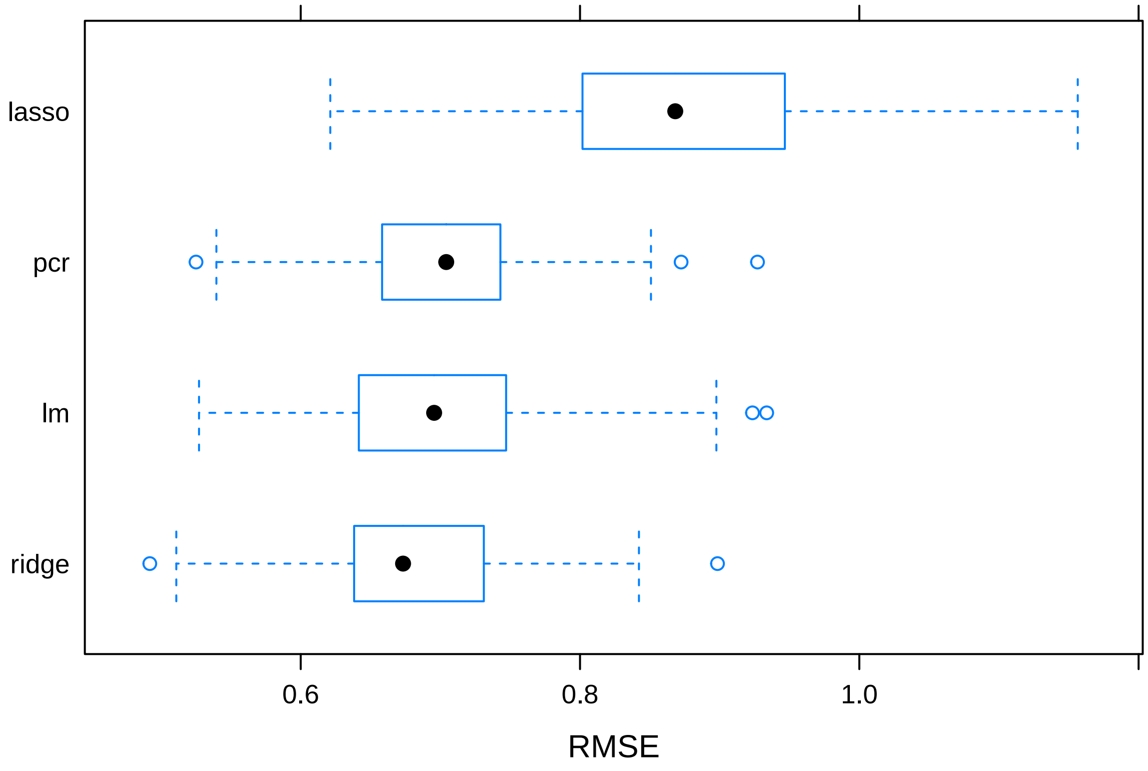
Section 3: Comparison

Problem E

```
resamp <- resamples(list(lm = fit.lm,
                         lasso = lasso.fit,
                         ridge = ridge.fit,
                         pcr = pcr.fit
                         ))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, lasso, ridge, pcr
## Number of resamples: 100
##
## MAE
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm    0.3931721 0.4823755 0.5170480 0.5215378 0.5642053 0.6771354    0
## lasso 0.4518713 0.6140475 0.6602448 0.6673720 0.7211136 0.8725327    0
## ridge 0.3896549 0.4801795 0.5219150 0.5211480 0.5520519 0.6694557    0
## pcr   0.3935424 0.4992509 0.5391444 0.5394799 0.5757574 0.7117268    0
##
## RMSE
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm    0.5271840 0.6420618 0.6955348 0.6981010 0.7469520 0.9336851    0
## lasso 0.6211677 0.8024573 0.8681973 0.8756455 0.9457939 1.1564303    0
## ridge 0.4919181 0.6397579 0.6732917 0.6819174 0.7310466 0.8984494    0
## pcr   0.5250311 0.6605466 0.7042044 0.7020385 0.7428360 0.9270920    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm    0.8054358 0.8633349 0.8881008 0.8841857 0.9089782 0.9454272    0
## lasso 0.7004428 0.8123465 0.8339057 0.8343658 0.8688120 0.9181735    0
## ridge 0.8012583 0.8726602 0.8912119 0.8890621 0.9079720 0.9464386    0
## pcr   0.8184997 0.8622085 0.8841218 0.8827658 0.9015702 0.9419949    0
```

```
bwplot(resamp, metric = "RMSE")
```



Based on these results for RMSE, I would choose the ridge regression method for predicting solubility.