Lab 5: Comparing Ridge, Lasso, and OLS

Getting started

Click the Github Assignment link on Moodle and open the repository in Posit Cloud.

Warm up

Let's warm up with some simple exercises. Open the `lab5.qmd` Quarto file. Update the YAML of your Quarto file with your information, knit, commit, and push your changes. Make sure to commit with a meaningful commit message.

Please make sure to keep committing throughout while working on the assignment!

Packages

You will need the `leaps` and `glmnet` packages. We strongly recommend that you use the `tidyverse` suite of packages for data visualization and data wrangling. The `patchwork` package may also come in useful.

```
# load packages here
```

Data

For this lab, we will once again be working with the 2022 data about soccer players from Major League Soccer that you used for homework 3. Please look at the codebook in the `README` page of the `data` folder of your repository for a reminder of what the different variables are.

Import the data as a data frame called `mls22`.

```
#Import data here
```

Exercises

Data Preparation

Q1

Recreate the changes from HW3:

- Recreate the `usa` variable
- Recreate the `GK`, `DF`, `MF`, and `FW` variables
- Make a `base_salary_log` variable that is the logged (base e) base salary
- Drop `NA`'s (save as a new object)

Comparing Predictive Performance

When doing machine learning, we are often concerned with predicting the outcome on previously unseen data. In previous labs and assignments, we have seen how we can compare different modeling specifications. We often try to find the best modeling approach to do so. The best approach may be different for different datasets and different questions. This means that a common part of the machine learning workflow is to compare different modeling approaches.

With best subsets and regularization/penalization expansions of linear regression, we are moving into a somewhat more complicated part of machine learning.

We can think of the optimal number of variables and the optimal λ as *tuning* parameters. As the book and example code show, the particular values of these tuning parameters impacts how well the model performs. But the model fitting process itself requires us to specify these ahead of time. So how do we make sure that we fit the best model that we can (at least based on prediction, which is what most machine learning focuses on)? And how do we compare different modeling approaches that all involve tuning parameters?

We want to find them before we can compare the performance of these different modeling approaches … but to find them, we need to do model comparison!

We can do this in three main ways:

1. Split the data into three sets:

- Training set: portion of data that will always be used to fit the model using a particular value(s) of the tuning parameter(s).
- Validation set: portion of data that will always be used to assess the performance of the model for the particular value(s) of the tuning parameter(s) (compared to other values of the tuning parameter[s])
- Test set: portion of the data that will always be used to assess the overall performance of the model (compared to other modeling approaches)

2. Split the data into two sets:

- Cross-validation set: portion of the data that will be used to do cross-validation. We perform cross-validation for a particular value(s) of the tuning parameter(s) and use the average across the folds to assess performance for the particular value(s) of the tuning parameter(s)
- Test set: portion of the data that will always be used to assess the overall performance of the model (compared to other modeling approaches)

3. Use nested cross-validation:

- Split the data into K folds
- For each fold k, use cross-validation on all folds *except* k to find the optimal tuning parameter value(s), then estimate test error on fold k
- Average across folds to estimate overall test error, which can be used to compare modeling approaches.

These approaches require successively more model fits (and so more computation time). More tuning parameters implies more model fits as well. More complex models usually have more tuning parameters to fit and so the process will also be more computationally expensive.

In addition, in the third approach, it is not clear how to conceive of the optimal value of tuning parameters, as different tuning parameters may be found in different sets of the data.

We will be using the **second** approach in this section: using cross-validation on the training set to choose the optimal tuning parameters and then comparing models on a test set.

For this section we will be regressing `base_salary_log` on `usa`, `GK`, `DF`, `MF`, `FW`, `Age`, `MP`, `Starts`, `Min`, `Gls`, `Ast`, `PK`, `PKatt`, `CrdY`, `CrdR`, `xG`, `npxG`, `xAG`, `PrgC`, `PrgP`, and `PrgR` (the same ones we used for the *all* model in homework 3).

Q2

Split the data into a 70/30 CV/test split. Set the seed to 16 beforehand. Create two separate data frames for these two sets called `mls22_cv` and `mls22_test`.

Q3

Use best subsets with cross-validation (like in the example code for this week) to find the best subset of variables from the ones use in the previous section. Plot the CV MSE against the size of the model. For cross-validation use 10 folds; set the seed to 17 before doing fold assignment. What does the optimum number of variables seem to be?

Then calculate the test set MSE (using `mls22_test`). You can do this using a `for` loop or using a more efficient method.

Note 1: We have to do cross-validation by hand because there is no built-in cross-validation function in the `leaps` package (unlike for ridge and Lasso). To do so, remember to first create a vector called `fold` that assigns a fold ID to each observation in `mls22_cv` (you can use in chunk `cross-validation` in this week's example code as an example).

Note 2: We will need a version of the `prediction.regsubsets()` method you first saw in the example code for this week.[1]

```
predict.regsubsets <- function(object, newdata, id, ...) {

   if(is.symbol(object$call[[2]])){
     i <- 2
     evals_form <- function(x){
       !rlang::is_formula(eval(x), scoped = TRUE)
     }
     pos_evals_form <- possibly(evals_form, otherwise = FALSE)
     while(pos_evals_form(object$call[[i]])){
      i <- i + 1
     }
     tt <- eval(object$call[[i]])
   } else {
     tt <- as.formula(object$call[[2]])
   }
   mat <- model.matrix(tt, newdata)
   coefj <- coef(object, id = id)
   xvars <- names(coefj)
   mat[, xvars] %*% coefj
}
```

[1] The difference between this method and the one defined in the book is that this method allows you to pass an object containing a formula to `regsubsets()`. Otherwise, the method would only work if a formula was directly passed to `regsubsets()`. Feel free to ask Dr. Hoellerbauer if you would like to know more.

Q4

Use `cv.glmnet()` to do 10-fold cross-validation to find the optimal λ for *ridge* regression. Set the seed to 18 beforehand. Use a λ sequence (check the help file for `cv.glmnet()` to figure out which argument to use) of `10^seq(10, -2, length = 100)`. Remember that you need to create an `x` matrix and `y` vector for `cv.glmnet()`, like in the example code.

What is the optimal λ? What is the test set MSE?

Q5

Use `cv.glmnet()` to do cross-validation to find the optimal λ for *Lasso* regression. Set the seed to 18 beforehand. Use all the same parameter as in Q4.

What is the optimal λ? What is the test set MSE? Which of the three approaches has the lowest test MSE?

Comparing Variable Selection Approaches

Q6

Based on the optimal Lasso model, which player characteristics seem to be important for predicting 2022 MLS player salaries? Remember to refit the model on the whole data with the optimal λ found in the previous section.

Q7

Based on the optimal best subsets model, which player characteristics seem to be important for predicting 2022 MLS player salaries? Remember to refit the model on the whole data with the optimal number of variables found in the previous section.

Q8

Compare the variables selected by Lasso and by best subsets approach. Are they the same? Different? Does this match what you would expect, based on what you know about soccer and salaries (which doesn't have to be a lot, don't worry!).

Final Steps

When you have made your final edits, remember to render, stage the changed files, then commit, and push to the repo.

Don't forget to submit your pdf on Canvas.

Grading

Please put your names on your answer document and add the date. You won't get points for doing this, but you will *lose* points for not doing so.

Remember to comment your code. You don't get points for commenting, but you will *lose* points for not commenting your code.

For clarity's sake, you should label your code chunks (this helps make debugging easier). You will *lose* points for not naming chunks.

If an exercise asks a question, you should answer it in narrative form and not just rely on the code. You will be *penalized* otherwise.

Please suppress warnings and messages. You will *lose* points if you do not suppress warnings and messages.

If you are asked to make a visualization, make sure that it is presentable, professional in appearance, has a title, and has properly labeled axes. You will *lose* points if you do not do this.

Points Breakdown

The different components of this lab are worth the following points:

- Q1: 2 pts
- Q2: 1 pt
- Q3: 6 pts (4 pts for CV including test MSE, 2 pts for plot)
- Q4: 3 pts
- Q5: 3 pts
- Q6: 3 pts
- Q7: 3 pts
- Q8: 2 pts
- Workflow (includes making sufficient commits): 4 pts

Total: 27 pts