MATH 3190 Homework 4

Focus: Notes 7 Part 1

Due March 9, 2024

Your homework should be completed in R Markdown or Quarto and Knitted to an html or pdf document. You will "turn in" this homework by uploading to your GitHub Math_3190_Assignment repository in the Homework directory.

Problem 1 (55 points)

Concrete is the most important material in civil engineering. The concrete compressive strength is an important attribute of the concrete and our goal is to predict the concrete compressive strength (in MPa) from the following variables:

- Cement (in kg/m^3)
- Blast furnace slag (in kg/m^3)
- Fly ash (in kg/m^3)
- Water (in kg/m^3)
- Superplasticizer (in kg/m^3)
- Coarse aggregate (in kg/m^3)
- Fine aggregate (in kg/m^3)
- Age of concrete (in days)

This dataset came from the UCI ML Repository and can be found on the Math3190_Sp24 GitHub page along with a Readme file.

Part a (3 points)

Read in the dataset. The data file is a .xls file, so you'll need to either convert it to a .csv or, preferably, use the `readxl` package to read in the Excel file. Once it is read in, change the names of the variables so they are shorter yet still descriptive.

Part b (5 points)

In the `GGally` library is the function `ggpairs`, which makes a nice scatterplot matrix in the `ggplot2` style. Create this scatterplot matrix for all of the variables in the dataset (they should all be plotted together in one plot).

Comment on the scatterplot matrix. Which variables seem to have a (at least somewhat) linear relationship with compressive strength? Does it seem like multicollinearity will be an issue here? Does it seem like the transformation of at least one variable is appropriate? There should be one (fairly) obvious variable that needs to be transformed.

Part c (8 points)

Fit a linear model (with the `lm()` function) predicting compressive strength using all other variables and include any transformations you thought were appropriate in part b.

Using `ggplot()`, make a QQ plot of the raw residuals. Include the QQ line as well. Comment on whether the residuals appear to be approximately normal.

Using `ggplot()`, make a plot of the jackknife residuals (obtained using the `rstudent()` function) on the y-axis and the fitted values of the model on the x-axis. Comment on whether this residual plot looks good. If it does not, indicate what the problem(s) is (are).

Part d (6 points)

Let's do some model selection to determine if any variables should be dropped from the model.

First, use the `step()` function on the model you fit in the previous part. This will select the variables using AIC.

Second, use `step()` with the option `k` equal to the log of the sample size. This will select the variables using BIC.

Third, use the `cv.glmnet()` function in the `glmnet` library (set a seed first) to fit a LASSO for variable selection. Note, the `model.matrix()` function will be helpful here to get a matrix to input for the `x` argument in the `cv.glmnet()` function. Use the "lambda.1se" value to select the variables and using that λ value, fit the LASSO model using `glmnet()`.

Finally, compare the variables that were selected by the three methods.

Part e (4 points)

Using the variables selected by the method that reduced the number of variables the most, fit a new ordinary least squares (OLS) model for predicting compressive strength using the `lm()` function.

Using `ggplot()`, make a QQ plot of the raw residuals. Include the QQ line as well. Comment on whether the residuals appear to be approximately normal and whether this plot looks better than the QQ plot in part c.

Using `ggplot()`, make a plot of the jackknife residuals (obtained using the `rstudent()` function) on the y-axis and the fitted values of the model on the x-axis. Comment on whether this residual plot looks good and whether this plot looks better than the residual plot in part c. If it does not, indicate what the problem(s) is (are).

Part f (10 points)

Since the residual plot still does not look good, let's try to use weighted least squares. Following slides 12-14 of Notes 7, create a vector of weights and then fit a model using weighted least squares.

Using `ggplot()`, make a plot of the jackknife residuals (obtained using the `rstudent()` function) on the y-axis and the fitted values of the weighted model on the x-axis. Comment on whether this residual plot looks good and whether this plot looks better than the residual plot in part e.

Part g (9 points)

Using the unweighted model from part e and the weighted model from part f, find and report both a confidence interval for the mean compressive strength and a prediction interval for the specific compressive strength for concrete that has a cement value of 300, a blast furnace slag of 90, a fly ash of 50, a water value of 200, a superplasticizer of 2.5, a coarse aggregate of 900, a fine aggregate of 600, and an age of 300 days. Note: some of those variables will not be used since you reduced the number of variables earlier. You can use the `predict()` function here and remember that you will need to find the specific weight for the given predictor variable values for the weighted intervals.

Comment on how these intervals differ. Does the change make sense given the value of the fitted value?

Part h (10 points)

Write a function called `predict_weighted` that takes three inputs: the **unweighted** model, the data frame containing the information about the value(s) of the predictor variables we are using to predict, and the interval type (either "confidence" or "prediction"). This function should return the predicted value and the interval bounds for the specified interval type for weighted least squares. So, this function should compute the weights, obtain the weighted least squares model, find the specific weight for the new value, and then get the prediction and the interval. This function should work for any model you give it, not just for this exact situation of this problem. So, you should not reference any data sets or variables specific to this concrete problem in the function.

Test this new function for the confidence and prediction intervals you made in part g.

Problem 2 (29 points)

An automobile consulting company wants to understand the factors on which the pricing of cars depends. The dataset `car_price_prediction.csv` in the GitHub data folder has information on the sales of 4340 vehicles.

Part a (3 points)

Read in the data file and take the log of all numeric variables. Then fit a linear model for predicting the log of selling price using all other variables except "name".

Part b (4 points)

Now fit some LASSO models for predicting log price using all but the "name" variable. Try the following values for the regularization parameter: $\lambda = 0, 0.01, 0.1$, and 1 and comment on how the coefficients of the model change.

Note: when $\lambda = 0$, the LASSO coefficients should equal the OLS model coefficients. However, they will actually be a bit off here. That is because the `glmnet()` function has a `thresh` argument that sets a threshold for convergence. It is, by default, set to `1e-07`. To make the parameters match, you can change that to `thresh = 1e-14` instead. This is not necessary, though.

Part c (4 points)

Now fit some ridge regression models for predicting log price using all but name. Try the following values for the regularization parameter: $\lambda = 0, 0.01, 0.1$, and 1 and comment on how the coefficients of the model change.

Part d (4 points)

Now fit some elastic net regression models for predicting log price using all but name. Try the following values for the regularization parameter: $\lambda = 0, 0.01, 0.1$, and 1 for $\alpha = 1/3$ and then comment on how the coefficients of the model change for each α from parts b, c, and d.

Part e (4 points)

Use the `cv.glmnet()` to find the "optimal" λ value (let's use the "lambda.1se") for LASSO, ridge, and elastic net and then fit models using the `glmnet()` function for all three using their respective "best" λ. **Set a seed** before running the `cv.glmnet()` each time.

Compare each model's variables and coefficients.

Part f (10 points)

Use bootstrapping with at least 1000 samples to estimate the standard errors of the coefficients of the ridge regression model. For each bootstrap sample, run the `cv.glmnet()` function to find the best λ and fit a model using that optimized λ. Use the "lambda.1se" for this. Then compare these bootstrapped standard errors to the standard errors for the OLS model you fit in part a. Are they larger or smaller? Does this make sense?

Problem 3 (16 points)

Sixty districts in California in 1990 were randomly selected. We want to predict the median house value for the district based on the district location (longitude and latitude), the median house age in the block group, the total number of rooms in the homes, the total number of bedrooms, the population in the district, the number of households, and the median income (in $10,000). The data are already in the .Rmd file.

Part a (2 points)

Fit a linear regression model predicting house value from the other variables.

Part b (4 points)

Fit a ridge regression using `cv.glmnet()` to choose the optimal λ. Set the seed before using`cv.glmnet()`.

Part c (6 points)

Compute the sum of squared errors for both the OLS model and the ridge regression model for the testing set. Remember to use the exact models you fit to the training sets. The `predict()` functions will be useful here. Compare the results and comment on why the ridge regression performs better here. Slides 26-29 of Notes 7 may be helpful.

Part d (4 points)

Use the `train()` function in the `caret()` package to find the optimal α, λ pair in this situation (using the training set and 5-fold cross validation). For λ, search for values between 0 and 10000 by 100 and for α, search for values between 0 and 1 by 0.05. Then fit a model using the optimal α and λ using `glmnet()` and compare the sum of squared errors using this optimized model to the two models in part c. Set a seed before running the `train()` function.