46-927, Fall 2017: Homework #1

Due 5:20PM, Monday, January 22, 2017.

Homework should be submitted electronically on canvas before the deadline. Please submit your homework as a single file (pdf or html). We recommend using Rmarkdown or jupyter notebooks to prepare your homework.

Note: For this homework, try to do all problems in python.

Please post any questions on the Piazza discussion board.

1. *Asymmetric loss and the regression function.* In class, we saw that the best possible estimator of Y when minimizing expected squared error in regression (if you know the distribution of X, Y) is the conditional mean, $\mathbb{E}(Y|X)$. This is called the *regression function*. In this problem, we will see how this result can change when you have a loss function other than squared error.

   Define the loss function

   $$\mathcal{L}(Y, f(X)) = b\left(e^{a(Y-f(X))} - a(Y - f(X)) - 1\right),$$

   with $a, b > 0$ and $f(X)$ representing the estimate of Y.

   (a) Plot the loss function against $z = Y - f(X)$ for $z \in [-2, 2]$, $a = 1.1$, $b = 2$. Describe what you see. Why might you use a loss function like this?

   (b) Find the optimal rule, $f(x)$, for minimizing the expected loss. Assume you can use the expectation of any function of Y conditional on X.

   Hints:

   - Follow the basic approach of the derivation from class: Get a nested conditional expectation of X, pass it through as many terms as possible, and take a derivative of the inside of the outer expectation with respect to $f(x)$.
   - Expect something that reminds you of a moment generating function.

(c) Suppose that you know the conditional distribution $Y|X = x = N(\beta \cdot x, \sigma^2)$, for known scalars $\beta, \sigma \in (0, \infty)$. What is your optimal estimator $f(x)$?

Useful fact: You know from your probability course that for a Gaussian random variable $Z \sim N(\mu, \sigma^2)$, the Moment Generating Function is:

$$\mathbb{E}(e^{tZ}) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}$$

for any $t \in \mathbb{R}$.

(d) There is code on blackboard (`asymm_loss.py`) for this problem. Change the body of the `f_yours` function to correspond to your estimator from the previous part. The code will run simulations to compare the average loss of your estimator to the average loss of the conditional mean (βx).

What is your average loss in the simulation? Do you have a lower average loss than the conditional mean? Why do you think this is, thinking back to the plot of the loss function and the form of your estimator.

2. Given a response vector $y \in \mathbb{R}^n$, predictor matrix $X \in \mathbb{R}^{n \times p}$, and tuning parameter $\lambda \geq 0$, recall the ridge regression estimate

$$\widehat{\beta}^{\text{ridge}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \; \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2.$$

(a) Show that $\widehat{\beta}^{\text{ridge}}$ is simply the vector of linear regression coefficients from regressing the response $\tilde{y} = \begin{bmatrix} y \\ 0 \end{bmatrix} \in \mathbb{R}^{n+p}$ onto the predictor matrix $\tilde{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+p) \times p}$, where here $0 \in \mathbb{R}^p$, and $I \in \mathbb{R}^{p \times p}$ is the identity matrix.

(b) Show that when $\lambda > 0$, the matrix $\tilde{X}$ always has full column-rank, i.e., its columns are always linearly independent, regardless of the columns of X. Hence argue that the ridge regression estimate is always unique, for any matrix of predictors X.

(c) Write out an explicit formula for $\widehat{\beta}^{\text{ridge}}$ involving X, y, λ. Conclude that for any $a \in \mathbb{R}^p$, the estimate $a^T \widehat{\beta}^{\text{ridge}}$ is a linear function of y. **Hint:** Take advantage of part (a).

3. In this problem, we will explore ideas related to a recent paper, Preis et al. (2013), which attempts to model DJIA returns based on Google Trends data.

In `trends_train.csv` and `trends_test.csv`, you will find training and test data sets of matching column format. For now we'll work with the training data set.

The data sets consist of the following columns:

- The date of the Sunday that *ends* the week of google trends data, which is aggregated Monday - Sunday. The volume differences in the other columns start one week before this date and end on this date.
- Google trends data for 96 words, with words selected by the methodology in the paper. The time series for each word reflects *change in* the volume of search queries that week, compared to the previous week. The original time series (before taking differences) were normalized to have a maximum of 100 over time; you can find these in *trend_timeseries.csv* just for fun.
- The weekly DJIA log returns for the week *after* the google trends data (Monday - Monday adjusted closing price).

You are welcome to do this problem in either R or python, but I highly recommend R. In python, I think you would have to write your own 1 s.e. rule. (All the information to do so is available in the model though.)

(a) Load the training data and see what you have. Fit the lasso to predict the weekly log return from the weekly changes in search volume. Use the defaut range of penalization parameters. Plot the coefficients as a function of $\log(\lambda)$.

(b) Carry out cross-validation for the lasso (using the built in procedure). Plot the cross-validation curve with its standard errors.

Note: For the purpose of this problem, we're going to carry out cross-validation as if the samples were i.i.d., neglecting the temporal structure. In reality, this would be something to worry about.

(c) Look at the models that you obtain through each selection rule (minimization vs. 1 s.e.). How many nonzero coefficients does each model have (other than the intercept)? What does this suggest to you about the amount of signal in the problem?

Note: The output of cross-validation depends on the random splits that you select. In this problem, you should get different values for λ_{min} and $\lambda_{\text{1s.e.}}$ to make it interesting. If by some chance you select the same model both ways, just rerun cross-validation again.

(d) Let's try the λ_{min} model out. Make predictions on your training data by using the predict function and passing your training X in as the `newx` parameter. The `s` parameter is the λ value you want to fit at. Make a scatterplot of your fitted values against the true values; is there any pattern?

(e) We could carry out a simple (and overly simplified) trading strategy based on your model, as outlined in the paper. Each Sunday, we look at the prediction from your model. If the log returns for the next week are predicted to be positive, we buy on the Monday and sell on the following Monday. If the log returns are predicted to be negative, we do the opposite. This lets us realize log returns for each week of

$$\text{(sign of our prediction)} \cdot \text{(DJIA log return)}$$

which is easy to work with. Compute your return for this strategy on the training data, using your λ_{min} model. Compare it to the return you would have realized if you bought on the first day of the data set and sold on the last day.

(f) Now apply your λ_{min} model, trained on the training set, to your test data. Again, you just use the predict function, changing the `newx` parameter. Plot your fitted values as before.

Carry out the same simple strategy on the test data. How well do you do? Again, compare to the overall return for the DJIA in this period. In the end, would you have been better off with the $\lambda_{\text{1s.e.}}$ model?

Preis, T., Moat, H. S., & Stanley, H. E. (2013). Quantifying trading behavior in financial markets using Google Trends. Scientific reports, 3, srep01684.