Homework 5: Sandwiches, Propagation, Lasso

36-462/662, Spring 2022

Due at 6 pm on Thursday, 24 February 2022

Agenda: Practice using "the usual asymptotics", specifically to calculate robust standard errors; learning a useful trick for propagating uncertainty, specifically from parameter estimates into predictions; trying our hand at actually doing regularized estimation.

1. (10) **Reading questions** are, as usual, online, and due on Monday at 6 pm.

2. **The "sandwich variance" for linear regression** Suppose that our data consists of IID pairs (X_i, Y_i), and that both X_i and Y_i are centered, one-dimensional random variables, so $\mathbb{E}[X_i] = \mathbb{E}[Y_i] = 0$. We want to estimate a linear regression of Y on X by least squares, so we would ideally like to find the β which minimizes $r(b) = \mathbb{E}\left[(Y - bX)^2\right]$. We do *not* assume that the true relationship between Y and X is linear.

   a. (4) It is known (e.g., from Lecture 2) that the optimal $\beta = \mathrm{Cov}[X, Y]/\mathrm{Var}[X]$. Use this to show that $\mathbb{E}[Y - \beta X] = 0$ and that $\mathrm{Cov}[Y - \beta X, X] = 0$.

   b. (4) Show that $r(b) = \mathrm{Var}[Y] + b^2\mathrm{Var}[X] - 2b\mathrm{Cov}[Y, X]$.

   c. (4) Show that the second derivative of $r(b)$ (with respect to b) is $r''(b) = 2\mathrm{Var}[X]$.

   d. (4) With finite data, we approximate $r(b)$ by $\hat{r}(b) = n^{-1}\sum_{i=1}^{n}(Y_i - bX_i)^2$. We'll call the minimizer of this $\hat{\beta}$. Define the residual for the i^{th} observation as $D_i(b) \equiv Y_i - bX_i$. Show that the first derivative of $\hat{r}(b)$ is

   $$\hat{r}'(b) = \frac{-2}{n}\sum_{i=1}^{n} D_i(b)X_i \tag{1}$$

   e. (4) Explain why it's reasonable, under our assumptions, to estimate $\mathrm{Var}[\hat{r}'(\beta)]$ by

   $$\hat{J}_n \equiv \frac{4}{n^2}\sum_{i=1}^{n} D_i^2(\hat{\beta})X_i^2 \tag{2}$$

   "Reasonable" here means you don't need to give a formal proof, but you should give reasons to explain why this $\hat{J}_n$ is connected to $\mathrm{Var}[\hat{r}'(\beta)]$. *Hints*: (i) What're the expectations of the summands in the definition of $\hat{J}_n$? (ii) Use Q2a.

   f. (5) Give an estimator for the variance of $\hat{\beta}$. Your answer should involve both $\hat{J}_n$ and the sample variance of X (and possibly other things, all of which can be calculated from data). *Hint*: Lecture 9.

   g. (4) Find an estimator for the standard error[1] of $\hat{\beta}$.

[1] Recall that every estimator is a random quantity. The standard error of an estimator is, as you'll remember from your mathematical statistics and linear models classes, *defined* as simply that estimator's standard deviation. The standard error tells us about how uncertain our estimates are, by saying how far the estimator typically is from its expected value. (This is proportional to the expected difference in estimates between two repetitions of the same experiment or random sample.) Every estimator has a standard error; you are used to seeing those for regression slopes in R's output after you run `lm()`. One example of a standard error is "the standard error of the mean", $\sigma/\sqrt{n}$, but standard errors for other estimators do not look like that.

3. **Simplifying Q2 when the model is correct** (3) Make all the assumptions from Q2, but *also* assume that $Y = \beta X + \epsilon$ where ϵ is IID with mean 0 and variance σ^2. (That is, assume the true regression function is linear, with constant-variance noise around that line.) Show that your expression for the standard error from Q2g will converge on $\sigma/\sqrt{n\text{Var}[X]}$ for large n.

4. **Propagation of error and uncertainty in predictions**. The following technique, called "propagation of error", "the delta method", or "propagation of uncertainty", is often useful in simplifying complicated calculations about the variances of functions.

   a. (3) Suppose that $R = f(T)$, where the random variable T has expectation μ and variance σ^2. Use a Taylor expansion of f to explain why $\text{Var}[R] \approx (f'(\mu))^2 \sigma^2$, at least when σ^2 is small.

   b. (3) Now suppose that $R = f(T_1, T_2, \ldots T_d)$, where T_i has expectation μ_i and variance σ_i^2. Assume the T_i are uncorrelated with each other. Assuming all the σ_i^2 are small, explain why

$$\text{Var}[R] \approx \sum_{i=1}^{d} \left(\frac{\partial f}{\partial t_i}(\mu_1, \ldots \mu_d)\right)^2 \sigma_i^2 \tag{3}$$

   c. (4) Suppose the situation is as in Q4b, but that $\text{Cov}[T_i, T_j] = \rho_{ij}$, not necessarily equal to 0. Explain why

$$\text{Var}[R] \approx \sum_{i=1}^{d} \left(\frac{\partial f}{\partial t_i}(\mu_1, \ldots \mu_d)\right)^2 \sigma_i^2 + 2\sum_{i=1}^{d-1}\sum_{j=i+1}^{d} \left(\frac{\partial f}{\partial t_i}(\mu_1, \ldots \mu_d)\right)\left(\frac{\partial f}{\partial t_j}(\mu_1, \ldots \mu_d)\right)\rho_{ij} \tag{4}$$

   d. (4) Define $\mathbf{\Sigma}$ as the matrix with diagonal entries $\sigma_1^2, \ldots \sigma_d^2$, and off-diagonal entries ρ_{ij}. Define $\vec{g}$ as the d-dimensional vector $\nabla f(\mu_1, \ldots \mu_d)$. (If you like, you can think of this as a $d \times 1$ matrix $\mathbf{g}$.) Is

$$\text{Var}[R] \approx \vec{g} \cdot \mathbf{\Sigma}\vec{g} = \mathbf{g}^T \mathbf{\Sigma} \mathbf{g} \; ? \tag{5}$$

   If so, explain why; if not, explain why not, and give a correct expression if possible.

   e. (5) Now suppose that our model / strategy / prediction rule makes the prediction $s(x;\theta)$ on information x when the p-dimensional parameters vector is $\theta = (\theta_1, \ldots \theta_p)$. We have a $p \times p$ variance-covariance matrix $\mathbf{c}$ for our estimated parameter vector $\hat{\theta}$ (perhaps from the "usual asymptotics" of Lecture 9, or from something like Q2, or perhaps from the Oracle). Explain, in words, how we could use $\mathbf{c}$, and the earlier parts of this problem, to get a variance for $s(x;\hat{\theta})$, our prediction at $X = x$. What, if anything, would we need to calculate, beyond $\mathbf{c}$?

5. **Lasso and spam** Re-load the `spam` data set from Homework 2, and divide it into a training and testing set as before. We'll work through using the lasso (L_1) penalty to regularize, and do variable selection, on fitting a logistic regression for spam detection. Load (and if need be install) the package `glmnet`. (The last page of this assignment gives some usage examples you may find helpful.)

   a. (3) The `glmnet` package's main function, also called `glmnet()`, is not quite as clever about formulas as `glm()`, so it requires two arguments, `x` and `y`, which need to both be matrices. Run `glmnet` with `y=` the `spam` column of the data set, and `x=` a matrix consisting of all the other columns. Be sure to set `family="binomial"` (so it knows to do a logistic regression) and `alpha=1` (which does what?). By default, `glmnet()` automatically generates a range of λ values and fits the model at each one. Run `plot()` on the model you created. This shows a picture of the estimated coefficients at different values of λ. Include the plot. Which side of the figure corresponds to large values of λ, and which to small values? How can you tell?

   b. (2) The element `lambda` in the object which `glmnet()` returns contains a vector of the actual values of λ it considered when fitting the model. How many values of λ did it try on this data? What were the biggest and smallest values tried?

c. (4) For each value of λ, calculate the average log loss on the training data. The easiest way to do this is to use the `predict()` function, which for `glmnet` requires a fitted model, a `newx` argument which is a matrix of values at which you want predictions, and a `type` argument which you find convenient to set to `"response"`. You will get back an array of predictions, one for each row of `newx` at each value of λ. Make a plot showing the average log loss on the training data versus λ. Describe the shape of the curve. *Hints*: look at examples of calculating average log loss for logistic regression in earlier homeworks; do the same thing to each column.

d. (4) For each value of λ, classify the rows in the training data as either `spam` or `email`. Plot the error rate as a function of λ, and describe the shape of the curve. *Hints*: Use `predict()` again.

e. (5) Repeat Q5c, but now make predictions for each row of the testing set, plotting the average log loss against λ. How does this curve differ from that in Q5c? *Hints*: You may find it helpful to have both curves in one plot (with different colors or line-types, etc.), and to use a log scale for the λ axis.

f. (5) Repeat Q5d, but, again, now classify the rows of the testing set, plotting error rate against λ. How does the shape of this curve differ from that in Q5d? *Hints*: See hint for Q5e.

g. (5) Based on what you have done so far, what value of λ would you recommend using? Explain your reasoning.

6. (1) Roughly how much time did you spend on this assignment?

Presentation rubric (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. Plots are carefully labeled, with informative and legible titles, axis labels, and (if called for) sub-titles and legends; they are placed near the text of the corresponding problem. All plots and tables are generated by code included in the R Markdown file. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. All parts of all problems are answered with actual coherent sentences, and raw computer code or output are only shown when explicitly asked for.

A few code examples, to get you started on Q5

```
# Make up some data
n <- 200
p <- 10
df <- data.frame(array(rnorm(n*p), dim=c(n,p)),
                 country=sample(c("Iran", "Turan"), replace=TRUE, size=n))

# Use glmnet() to fit logistic regression with lasso regularization, at many
# different values of lambda
library(glmnet)
lasso.fits <- glmnet(y=df$country,
                     x=as.matrix(df[,-(p+1)]),
                     family="binomial",
                     alpha=1)
# Plot the coefficients as lambda changes
plot(lasso.fits)

# Get predictions on the original data
predictions.in.sample <- predict(lasso.fits, newx=as.matrix(df[,-(p+1)]),
                                 type="response")
# Predictions like this will be a matrix, because we have many data points
# and many values of lambda
dim(predictions.in.sample)
# Compare the first few classes to the first few predictions --- how do you
# think the predictions work here?
head(predictions.in.sample)
head(df$country)

# Make up data to serve as test cases
new.data.points <- matrix(rnorm(n=47*p), ncol=p)

# Get predictions on a different data set
dim(predict(lasso.fits, newx=new.data.points, type="response"))

# Get predictions on a different data set at a particular value of lambda
  # Why they decided to call this argument s and not lambda, I do not know
dim(predict(lasso.fits, newx=new.data.points, s=lasso.fits$lambda[1],
            type="response"))
# Look at the first few of those
head(predict(lasso.fits, newx=new.data.points, s=lasso.fits$lambda[1],
            type="response"))

# Get predictions on a different data set on the log-odds scale at a particular
# value of lambda
head(predict(lasso.fits, newx=new.data.points, s=lasso.fits$lambda[1],
             type="link"))
```