Lab 6

Statistical Computing, 36-350

Friday October 9, 2015

Today's agenda: Least squares regression and least absolute deviations regression using grid search.

General instructions for labs. Upload an R Markdown file, named .Rmd", to Blackboard. You will give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. Include the name of your lab partner at the top of the file.

R Markdown setup. Open a new R Markdown file; set the output to HTML mode and click "Knit HTML". This should produce a web page with the knitting procedure executing your code blocks. You can edit this new file to produce your homework submission. Alternatively, you can start from the lab's R Markdown file posted on the course website, as a template.

Background. We'll examine two types of regresssion (Linear least squares regression and least absolute deviations regression), and code a simple version of the two regressions ourselves. First, Gaussian linear regression can be thought of as finding the best fit linear function to the data, but assuming that the data contains some independent Gaussian noise

$$f(x) = a + bx, \quad y = f(x) + \mathcal{N}(0, \sigma^2).$$

We learned in Homework 2 that one solution was entirely made possible by a simple linear algebra formulation. We will now demonstrate finding the solution with an alternate approach – using a simple grid-search to find the minimizer. The data we will use comes from Boston housing data, and has 506 rows and 14 columns.

```
library(MASS)
data(Boston)
#?Boston # Uncomment this to learn about the Boston housing data
y = log(Boston$medv)
x = log(Boston$lstat)
```

We will isolate our attention to two variables; median value of owner-occupied homes in $1,000s, and lower status of the population (percent). Specifically, we will find a linear regression between the response variable y = log(medv) and x = log(lstat).

Part I: Least Squares Linear Regression

1. First, notice there are some median housing values that are equal to $50,000; these are probably artificial *caps* (anything larger than 50,000 is replaced with the cap 50,000 while recording data), so they disrupt our analysis. Identify these points, and assign to `y` the log median house values and to `x` the log crime rate *after* having eliminated these points. Having eliminated these points, our model is only suitable for predictions reasonably within the data range. Use these for the remainder of the lab.

2. Write a loss function called `squareloss()` that takes in two arguments (scalar values `y`,`u`) and returns the square of the difference:

   $$(y - u)^2$$

   Fixing `u = 0`, evaluate the loss function over a set of equally distant values between -2 and 2 of `y` and plot it.

3. Write a function `meanerror()` that takes in five arguments (scalar values `beta0` and `beta1`, and vectors `x` and `y`, and a loss function `lossfun`) and returns the following result. This gives a quantification of how 'off' our function $\beta_0 + \beta_1 x$ is, in the training data. For instance, using square loss, this becomes

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

and using absolute loss, this becomes

$$\sum_{i=1}^{n} |y_i - (\beta_0 + \beta_1 x_i)|$$

which may be helpful for part II.

4. Construct a function `lm.gridsearch()` that takes in four inputs – the explanatory variable `x`, the response variable `y`, a numeric (scalar) value `B`, and a function `lossfun` whose default value of `squareloss` – and returns a 2-length vector which contains the best combination of β_0 and β_1 in a 2D grid constructed with candidate values each between `-B` and `B`. The candidate values should be, by default 1000 equally spaced points between `-B` and `B`. Use it on your data with `B=10`.

5. Congratulations, now you know **two** ways to obtain a linear regression. Compare your current solution to the solution you had obtained from homework 2, which used $(X^T X)^{-1} X^T y$ (revisit HW2 for details). Plot the best fit linear regression line ($y = \beta_0 + \beta_1 x$) obtained using the two methods but with different colors, and overlayed with the data.

6. Fit a linear model using `lm()` learned in class, and extract the coefficients using `coef()`. Confirm that they agree with the results of your grid search.

7. Examine the normality residuals of your linear model using a quantile-quantile plot (QQ plot). Explain your plot; do the residuals look normal (Gaussian)?

Part II: Least Absolute Deviations Regression

Least absolute deviations (LAD) regression is a different flavor of least squares regression a similar goal; it finds a best-fit linear function to the data, but using a different loss function. Unfortunately, there does not exist a clean and easy closed form solution (like $(X^T X)^{-1} X^T y$ in least squares), so we will use the grid search we've learned in part 1.

8. Write a loss function called `absloss()` that takes in four arguments (scalar values `y`,`u`) and calculates the following value.

$$|y - u|$$

As in problem 1, fixing `u = 0`, evaluate the loss function over a set of equally distant values between -2 and 2 of `y` and plot it. Describe how it is different from square loss.

9. Now, simply apply `lm.gridsearch()` with the loss function `absloss` and `B=10` to find the LAD regression coefficients β_0 and β_1.

10. Lastly, plot the two regression lines (least squares and LAD) in the same plot, the least squares model in red, and the quantile model in black – along with the data points. If all went correctly, the least square regression should be slightly higher.)

11. (Bonus, optional) Let's add a couple of misleading points to the dataset, and run the above procedure again. Add the resulting lines to the plot you made in 10 in the same respective colors as before, but in dotted lines.

```
y = c(y,-4,-5)
x = c(x,3,4)
```

If the least squares regression changed noticeably, while the LAD regression is very similar, you've done the right thing. This is a strength of least absolute deviations regression; the model estimation is *robust* against some influential or outlying points. Why though? For full bonus credit, provide a sensible explanation.