Homework 7

Due Wednesday Oct 16, 2019

2019-10-16

For each assignment, turn in by the due date/time. Late assignments must be arranged prior to submission. In every case, assignments are to be typed neatly using proper English in Markdown.

This week, we spoke about parallelizing our R code. To do this homework, we will use ARC resources. I have added you to an "allocation" called arc-train4. If you go to ondemand.arc.vt.edu, use the Rstudio interactive app on Cascades, use the basic bio version of R, arc-train4 as the account, request 10 cores for 48 hours. The first time you do this, it will take 4-20 min to create the image being used, after that, it should be quick.

Problem 1

Create a new R Markdown file within your local GitHub repo folder (file–>new–>R Markdown–>save as).

The filename should be: HW7_lastname, i.e. for me it would be HW3_Settlage

You will use this new R Markdown file to solve the following problems.

Problem 2

Bootstrapping

Recall the sensory data from five operators:
http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat

Sometimes, you really want more data to do the desired analysis, but going back to the "field" is often not an option. An often used method is bootstrapping. Check out the second answer here for a really nice and detailed description of bootstrapping: https://stats.stackexchange.com/questions/316483/manually-bootstrapping-linear-regression-in-r.

What we want to do is bootstrap the Sensory data to get non-parametric estimates of the parameters. Assume that we can neglect item in the analysis such that we are really only interested in a linear model lm(y~operator).

Part a. First, the question asked in the stackexchange was why is the supplied code not working. This question was actually never answered. What is the problem with the code? If you want to duplicate the code to test it, use the quantreg package to get the data.

Answer

Part b. Bootstrap the analysis to get the parameter estimates using 100 bootstrapped samples. Make sure to use system.time to get total time for the analysis. You should probably make sure the samples are balanced across operators, ie each sample draws for each operator.

Answer

```
sensory_data <- read.csv("sensory_data.csv")[,2:4]
str(sensory_data)
```

```
## 'data.frame':    150 obs. of  3 variables:
##  $ item   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ operate: int  1 2 3 4 5 1 2 3 4 5 ...
##  $ value  : num  4.3 4.9 3.3 5.3 4.4 4.3 4.5 4 5.5 3.3 ...
```

Part c. Redo the last problem but run the bootstraps in parallel (`cl <- makeCluster(8)`), don't forget to `stopCluster(cl)`). Why can you do this? Make sure to use system.time to get total time for the analysis.

Create a single table summarizing the results and timing from part a and b. What are your thoughts?

Answer

Problem 3

Newton's method gives an answer for a root. To find multiple roots, you need to try different starting values. There is no guarantee for what start will give a specific root, so you simply need to try multiple. From the plot of the function in HW4, problem 8, how many roots are there?

Create a vector (`length.out=1000`) as a "grid" covering all the roots and extending +/-1 to either end.

Part a. Using one of the apply functions, find the roots noting the time it takes to run the apply function.

Part b. Repeat the apply command using the equivelant parApply command. Use 8 workers. `cl <- makeCluster(8)`.

Create a table summarizing the roots and timing from both parts a and b. What are your thoughts?

Problem 4

Gradient descent, like Newton's method, has "hyperparameters" that are determined outside the algorithm and there is no set rules for determing what settings to use. For gradient descent, you need to set a start value, a step size and tolerance. Using a step size of $1e^{-7}$ and tolerance of $1e^{-9}$, try 10000 different combinations of β_0 and β_1 across the range of possible β's +/-1 from true making sure to take advantages of parallel computing opportunities. In my try at this, I found starting close to true took 1.1M iterations, so set a stopping rule for 5M and only keep a rolling 1000 iterations for both β's. If this is confusing, see the solution to the last homework.

Part a. What if you were to change the stopping rule to include our knowledge of the true value? Is this a good way to run this algorithm? What is a potential problem?

Part c. Make a table of each starting value, the associated stopping value, and the number of iterations it took to get to that value. What fraction of starts ended prior to 5M? What are your thoughts on this algorithm?