Peer review for Kai Liu

Su Chen

September 18, 2016

I reviewed exercise 02 for Kai Liu. Overall, he is doing a great job writing code that's very easy to follow. He also created some nice looking and informative plots to compare results. Here are what I learned in his work.

- consistent and descriptive naming convention. When I look at my own code, I see a lot of variables with a single letter names because I was too lazy:(This is definitely not good for other people trying to understand my code. I will use more descriptive names next time.
- you wrote a function to sample data so that the batch stochastic gradient descent is looping over all the data points before going into the next round. Good modular code practice! I also learned the operators %/% and %% in r, which I have not seen before.
- I like your plot comparing different step sizes for the constant step size stochastic gradient descent. It is showing the clear message that a step size too large makes the likelihood function jumping around while a step size too small will make it converging too slowly.

I also found two issues:

- for the exponentially weighted average

```
# tracking exponentially weighted average
        l.single = l.beta.single(x.sample, y.sample, new.beta)
        if (count > 1000 & count <= max.iter) {
            l.weighed = lambda * l.single + (1 - lambda) * l.weighted
        }
        else {
            l.weighed = l.single
        }
        l.weighted.tracking = append(l.weighted.tracking, l.weighed)
```

I think the reason why the exponentially weighted average likelihood is not working is the if/else statement here. R is very picky for the syntax so you will have to put if/else as ONE line, otherwise it will throw an error message at you.

```
if (test_expression) {
   statement1
} else {
   statement2
}
```

- for the running time average

```
        # tracking average l(beta)
        # l.average = l.sum / count
        l.average = (l.single + 2*l.average)/count
        l.average.tracking = append(l.average.tracking, l.average)
```

I think here you meant:

```
        l.average = (l.single + (count-1)*l.average)/count
```

that's why your running time average likelihood is converging to 0.

- I think for the likelihood function evaluated at a single data point, you can scale it by N (the total number of data points, so treat it as if you identical data in the whole data set). This does not make any difference theoretically, but it is easier to see the convergence trend for the likelihood value that's not too small, and it will be more visually comparable to the likelihood function evaluated at the whole data set.
- Also a reminder of what I learned from Jennifer in class. She mentioned for r, it is much faster to save a variable in fixed length/dimension than appending it in each iteration.