RGallery: A Package for 3 questions in Stochastic Average Gradient Project

Eric Xin Zhou

February 19, 2015

1 3 Problems

Q1(easy) : Use glmnet[1] to fit an L2-regularized logistic regression model. Use the system.time function to record how much time it takes for several data set sizes, and make a plot that shows how execution time depends on the data set size.

Q2(medium) : create a simple R package with one function and one documentation file, and upload it to your GitHub account.

Q3(hard) : Harder: Write an R package which uses .C to interface C code.

1.1 Data simulation setup for L2

Given that test 1 need us provide different data set of different size to record how much time **glmnet** takes for different data size.

I generate Gaussian data with N observation and p predictors. with each pair of predictors $X_j, X_{j'}$ has the same population correlation ρ. If N and ρ are determined. We generate the observed data Y by adding several gaussian noise.

$$Y = \sum_{j=1}^{p} X_j \beta_j + kZ \tag{1}$$

If Y is a $N \times 1$ column vector, then $X_j, X_{j'}$ are all $N \times 1$ column vectors, so $\mathbf{X}$ is a $N \times p$ matrix and β is a $p \times 1$ column vector.

Z represents noise of observation, and k is chosen so that we can control signal-to-noise ratio to 3.0.

In generation model, we also should simulate the coefficient vector β, we define that

$$\beta_j = (-1)^j \exp\left(\frac{-2(j-1)}{20}\right) \tag{2}$$

This guarantee that the coefficients are constructed to have alternating signs and to be exponential descreasing.

And in logistical regression model, what we observation is $\mathcal{G} = \{1, 2\}$, therefore, the logistic regression model represents the probability we observed $\{1, 2\}$.

$$\begin{aligned} Pr(G = 1|x) &= \frac{1}{1 + e^{-(\beta_0 x_0 + ... + \beta_p x_p)}} \\ Pr(G = 2|x) &= \frac{e^{-(\beta_0 x_0 + ... + \beta_p x_p)}}{1 + e^{-(\beta_0 x_0 + ... + \beta_p x_p)}} \end{aligned} \tag{3}$$

So we can get the column $\log\left(\frac{Pr(G=1|X)}{Pr(G=2|X)}\right) = \sum_{j=1}^{p} x_j \beta_j$. In this model, if $Pr(G = 1|x) > Pr(G = 2|x)$, then response is actual 1, otherwise responese is 2.

However, in real observation, noise will be introduced into this regression model. As the result, we should add an $Z \sim \mathcal{N}(0, 1)$ into the original LR model.

So, the response Y we generate comes from Eqn.(4)

$$y = \sum_{j=1}^{p} X_j \beta_j + kZ \tag{4}$$

And we also can tell that $y \to \beta_0 x_0 + ... + \beta_p x_p$, therefore, we can determine $Pr(G = 1|x) = \frac{1}{1+e^{-y}}$. And then we will define our observation Y by binomial model which generate Y with $Pr(Y = 1) = p$ and $Pr(Y = 2) = 1 - p$.

2 Build a package for data set generation and time record

Since **Q2** require us develop a package. Therefore, I develop a simple package for LR2's data simulation. And record all of the time on different data set.

And then, to record glmnet time of different data set size, therefore I create a package RGallery to solve this question.

To solve **Q1**, if we fixed feature dimension to 300. then we execute **glmnet()** on different N size data set. And we let $N \in [100, 500, 1000, 2000, 5000, 10000, 100000]$, in addition, p is fixed $p = 300$.

```
require(RGallery)
N <- c(100, 500, 700, 1000, 1500, 2000)
glmnet_time <- sapply(N, function(n){
                          Gbenchmark(sim_dat_LR, N = n, p = 300)
                         }
                      )
colnames(glmnet_time) <- N
```

	user_time/sec	sys_time/sec	elapesd/sec
100	0.028	0.000	0.029
500	0.213	0.001	0.215
700	0.254	0.001	0.255
1000	0.343	0.001	0.345
1500	0.472	0.001	0.472
2000	0.679	0.002	0.683

Even though that glmnet analysis time changed slightly with data set size increasing, function **mvrnorm()** has become too slow when data size was up to 2000×300.

As the result, I should rewrite our **mvrnorm()** by a C++ function, so that the time consuming in data simulation can be compressed.

3 Rcpp accerlation

In **Q3**, mentor give us an example of writing package by **.C** and **.Call**. And in this package, I prefer Rcpp, which provides an elegant way to handle Cpp code.

In this Rcpp source file, I apply the Choleski decomposition to execute the function **cmvrnorm()**. And then, with the help of **Rcpp**(use .Call in RcppExports.R). We can quickly caculate out $15,000 \times 300$ data set.

```
M <- c(100, 500, 700, 1000, 1500, 2000, 4000, 8000, 10000, 15000)
glmnet_time_c <- sapply(N, function(n){
                          Gbenchmark(fast_LR_sim, N = n, p = 300)
                        }
                    )
colnames(glmnet_time_c) <- M
```

	user_time/sec	sys_time/sec	elapesd/sec
100	0.032	0.003	0.039
500	0.200	0.002	0.203
700	0.263	0.001	0.263
1000	0.358	0.000	0.359
1500	0.495	0.001	0.497
2000	0.699	0.001	0.701
4000	1.414	0.002	1.415
8000	2.193	0.005	2.229
10000	2.717	0.021	2.746
15000	3.585	0.005	3.588

In addition, we also can draw a relationship map for different data set size and glmnet's elasped time.

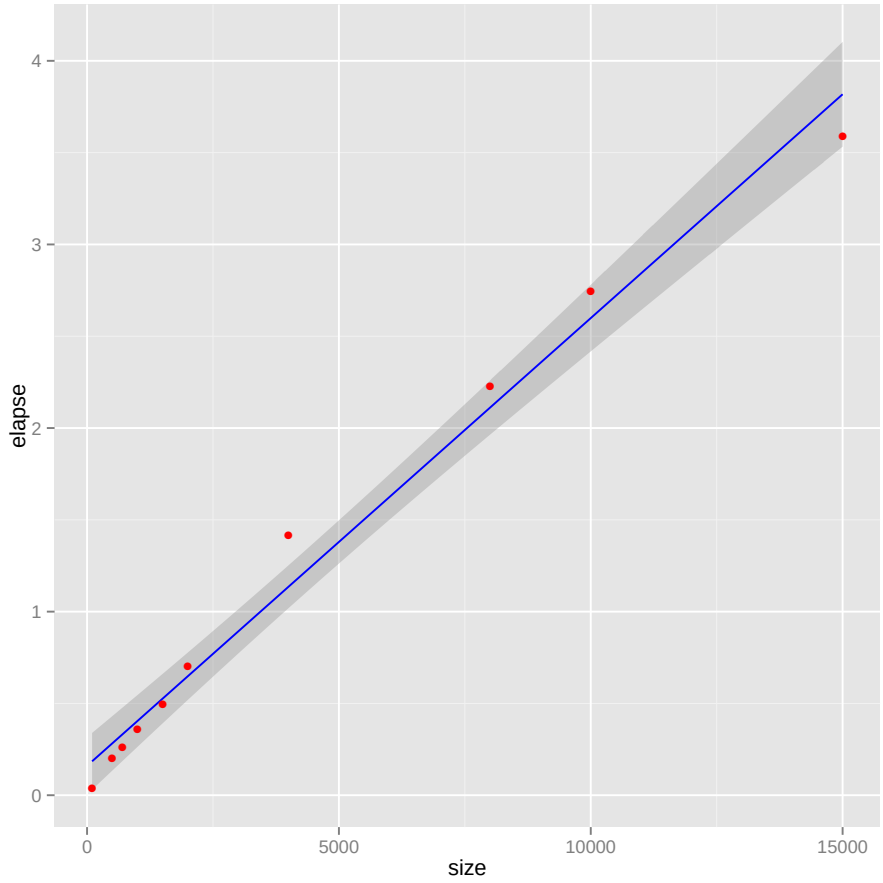

Figure 1: glment time　data size

4 Conclusion

Improvement next, RGallery package can be improvement, my plan is to create a cuda version for the function **cmvrnorm**, given that matrix multiplication perform better on GPU, I wish **cudamvrnorm** could run faster than other ***mvrnorm** functions.

In this RGallery package, I have completed the three questions given on SAG project's homepage. Therefore, I think I can meet mentors requirements and I wish I could work with you for the SAG project!

References

[1] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.