Penn State STAT 540
Homework #1, due Thursday, February 8, 2018

What you have to submit in a Canvas submission folder: (i) Your `R` code in a file titled PSUemailidHW1.R (e.g. muh10HW1.R), (ii) pdf file that contains a clear writeup for the questions below named PSUemailidHW1.pdf (e.g. muh10HW1.pdf). Note that your code should be readily usable, without any modifications.

1. Consider an $n \times n$ covariance matrix Σ with its (i, j)th element defined as follows: $\Sigma_{ij} = \exp(-|i - j|/\phi)$, with $\phi = 0.2$.

   (a) Write computer code to simulate $M = 100$ draws from a multivariate normal distribution with mean 0 (vector of 0s) and covariance matrix Σ as above for $n = 1000$. Write your own code to also evaluate the multivariate normal pdf at each simulated value. (Recall that you may use some ideas discussed in lecture.) Write pseudocode for both algorithms. You do not have to provide details for standard operations like eigendecompositions, you can simply write, for instance: "Perform a spectral decomposition of M."

   (b) Plot the pdf values for the samples. Report the wall time for the simulation algorithm and the evaluation of the pdf (jointly) using, say, `system.time`.

   (c) Now that you have successfully implemented this for $n = 1000$, repeat the exercise for $n = 2,000, n = 3,000$ and so on, going up to the largest value you are able to handle. Make plots of how the computational cost scales with increasing n.

   (d) Write out the computational complexity of your algorithm. Show the details of your calculation. If you have used an R function (e.g. eigen), you need to find out the cost of that algorithm by looking through the documentation.

   (e) Plot the theoretical cost of your algorithm as you increase n. Do this in terms of *flops* (floating point operations). How does the computation scale here when compared to the plot you produced above? (Try to make the plots easy to compare.) Do you have any explanation for differences between this plot and the above plot? Note: the discussion becomes more interesting as n gets large.

2. Monte Carlo methods for random matrices: Consider the following approach for generating random covariance matrices of dimension $m \times m$: Simulate m^2 $N(0, 1)$ random variates to obtain an $m \times m$ matrix R. Obtain the positive definite covariance matrix $\Sigma = RR'$. Note that this is a way to simulate a matrix with a particular Wishart distribution. Read through all the parts below before beginning to work on this problem.

   (a) Let $d_i = \lambda_i - \lambda_{i-1}$, where λ_i is the ith eigenvalue. What are the expected values of d_1 and d_2? Report Monte Carlo standard errors

and the sample size you used to approximate the expected value in each case. Also plot approximate density plots for d_1, d_2. Do this for $m = 100$, $m = 1000$, $m = 10,000$. Write a brief (1-2 sentence) summary of what you have learned.

(b) What is the expected value of the smallest eigenvalue of Σ? Plot approximate density plots for the smallest eigenvalue. Report Monte Carlo standard errors and the sample size you used in each case. Do this for $m = 100$ and $m = 1000$.

(c) Now study how the expected smallest eigenvalue changes as a function of m. (i) Plot approximate $E(\lambda_1)$, expected value of smallest eigenvalue, versus m. (ii) Report your grid of m values. (iii) What is the largest m value for which you approximated the expectation?

3. Matrix inversion: Consider matrices of the form $\Sigma = \sigma I + KK'$ where K is an $N \times M$ matrix of iid N(0,1) random variates and $\sigma = 0.2$. Fix $M = 10$. Compute the inverse of the matrix using two different algorithms: (i) directly by using the `solve` function and (ii) using the Sherman-Morrison-Woodbury identity discussed in class. I have provided example code for simulating the matrix and timing the solve function here `http://personal.psu.edu/muh10/540/Rcode/hw01.R` Report the following: (a) Plot the CPU time versus N for algorithm 1 and algorithm 2 (you will have to determine the grid and how large you can go with N); (b) a plot of floating point operations (flops) versus N for algorithm 1 and algorithm 2; (c) briefly summarize what you observe based on a comparison of the computational costs for the two algorithms using flops and using CPU time.

4. A simple simulation study: Let $X_1, \ldots, X_n$ iid with common distribution Poi(λ). Now consider constructing 95% confidence intervals for λ. Denote $\hat{\lambda} = \sum_{i=1}^{n} X_i/n$, and $\hat{\sigma}^2$ is the sample variance. Here are three confidence intervals:

   (i) $\hat{\lambda} \pm 1.96\hat{\lambda}/\sqrt{n}$, using the sample mean to directly estimate the variance/standard error.

   (ii) $\hat{\lambda} \pm 1.96\hat{\sigma}/\sqrt{n}$, using sample standard deviation to estimate the variance/standard error.

   (iii) Using the fact that we can calculate quantiles for the Poisson distribution (R command: qpois), construct an "exact" 95% interval for λ.

   Construct the above confidence intervals for two different sample sizes, $n = 10, 200$, and two different λ values, $\lambda = 2, \lambda = 50$. You should report a 4×3 table (rows are the different scenarios and the columns are the three different confidence intervals) that contains the estimated coverage rates along with Monte Carlo standard errors for each coverage rate. Write

a brief summary of what you have found, and which confidence interval approach you would recommend.

5. Let $X_1, \ldots, X_n$ iid from Gamma(α, β) distribution. Consider Bayesian inference for α, β with prior distributions $\alpha \sim N(0, 3)$, $\beta \sim N(0, 3)$. Use a Laplace approximation (as discussed in class) to approximate the posterior expectation of α, β. Show your work, and provide pseudocode for your algorithm. You will likely need to use the R function `optim` to carry out numerical optimization. The function is well documented. The data for this problem are available here: `http://personal.psu.edu/muh10/540/Rcode/hw1.dat`