ps5

Baoyue Liang

10/13/2018

Problem 1

Suppose that 1,..., n are the eigenvalues of A. Then the λs are also the roots of the characteristic polynomial, i.e.

$$det(A - \lambda I) = p(\lambda) = (1)^n(\lambda - \lambda_1)(\lambda - \lambda_2)...(\lambda - \lambda_n) = (\lambda_1 - \lambda)(\lambda_2 - \lambda)...(\lambda_n - \lambda)$$

The first equality follows from the factorization of a polynomial given its roots; the leading (highest degree) coefficient $(1)^n$ can be obtained by expanding the determinant along the diagonal.

Now, by setting λ to zero (simply because it is a variable) we get on the left side det(A), and on the right side $\lambda_1\lambda_2...\lambda_n$ that is, we indeed obtain the desired result.

$$det(A) = \lambda_1\lambda_2...\lambda_n$$

So the determinant of the matrix is equal to the product of its eigenvalues.

Problem 2

When z is large, $exp(z)$ can be large(with more than 16 digits) and $1 + exp(z)$ is rounded into the nearest possible value to $exp(z)$ and is thus is not accurate.

We could rearrange the function as follow to make it numerically stable.

$$expit(z) = \frac{1}{1 + exp(-z)}$$

Problem 3

```
set.seed(1)
z <- rnorm(10, 0, 1)
x <- z + 1e12
formatC(var(z), 20, format = 'f')
```

```
## [1] "0.60931443706111987346"
```

```
formatC(var(x), 20, format = 'f')
```

```
## [1] "0.60931216345893013386"
```

```
dg <- function(x, digits = 20) formatC(x, digits, format = 'f')
dg(z[2])
```

```
## [1] "0.18364332422208223816"
```

```
dg(x[2])
```

```
## [1] "1000000000000.18359375000000000000"
```

For x, since before the dot there is already 12-13 digits, only 3-4 digits after the dot is accurate. However, z has 16 digits of percision. When calculatin the variance, z has 16 digits percision, while X only has 3-4 figits percision. Thay explains why the two variance agree to only 5 digits.

Problem 4

(a)

If we divide into n individual column-wise computations, there are so many tasks and each one takes little time,the communication overhead of starting and stopping the tasks will reduce efficiency. Therefore it is better to divided the matrix into p blocks to improve efficiency.

(b)

Reference: I asked Chenxing Wu for her opinion on this quesion since I need clarification on the assuptions of this problem.

Assume each element in X, Y takes up 8 bytes.

(1)Approach 2 is better at minimizing memory use.

Approach 1: There are p copies of X and p blocks of Y which takes up $8(pn^2 + n^2)$ bytes.And the result takes $8n^2$ bytes. So the momory used should be $8n^2(p + 2)$ bytes.

Approach 2: There are p copies of one block of X and p blocks of Y at any single moment which takes up $16n^2$. And the result in a single moment takes up $8n^2/p$ bytes. So the memeory used shoud be $8n^2(2 + \frac{1}{p})$ bytes.

(2) Approach 1 is better at minimizing communication.

Approach 1: We need to transfer X p times, each of the p blocks of Y once and the result. Totally, the communication cost should be $n^2(p + 2)$.

Approach 2: we need to transfer each of the p blocks of X p times and each of the p blocks of Y p times and the result. Totally, the communication cost should bt $n^2(2p + 1)$