ps5

Weijie Yuan

10/17/2018

Problem 1

According to the question, A is a symmetric matrix. Suppose the dimension of A is n*n. Now, we have

$$A = \Gamma\Lambda\Gamma^T$$

Then, what we need to prove is that the determinant of A is equal to the product of the eigenvalues.

$$\begin{aligned}
|A| &= |\Gamma\Lambda\Gamma^T| \\
&= |\Gamma||\Lambda||\Gamma^T| \\
&= |\Gamma||\Gamma^T||\Lambda| \\
&= |\Gamma\Gamma^T||\Lambda|
\end{aligned}$$

Because Γ is an orthogonal matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues. Therefore, we have

$$\begin{aligned}
\Gamma\Gamma^T &= I \\
|A| &= |I||\Lambda| \\
&= |\Lambda| \\
&= \prod_{i=1}^{n} \lambda_i
\end{aligned}$$

This supports the conclusion.

Problem 2

```r
z=1e10
expit<-function(z){
  return(exp(z)/(1+exp(z)))
}
expit(z)
```

```
## [1] NaN
```

It returns NaN, which means that it is not computable in R when z is large.

```r
z=1e10
exp(z)
```

```
## [1] Inf
```

Because when z is large and exp(z) is difficult to represent, R returns 'Inf'. As we all know, Inf/(Inf+1) is obviously not computable in R.

To fix that, we can implement the following transformation:

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

Now even z becomes large, $exp(-z) \to 0$. $\frac{1}{0+1}$ is still computable in R. We can validate it using the following code:

```
z=1e10
expit_trans<-function(z){
  return(1/(1+exp(-z)))
}
expit_trans(z)
```

```
## [1] 1
```

This shows that the numerical issue has been successfully fixed.

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"
```

Firstly, we claim that $Vaz(X) = Var(Z+a) = Var(Z)$ where a is a fixed number. i.e, the variance of z and x mentioned above should be same mathematically.

Then we look into x and z vector to get some insights about what happens:

```
formatC(z[4], 20, format = 'f')
```

```
## [1] "1.59528080213779155372"
```

```
formatC(x[4], 20, format = 'f')
```

```
## [1] "1000000000001.59533691406250000000"
```

It seems that the accurancy of difference between corresponding element of x and z limited to 10^{-3}. Theoretically, x is around 10^12. Then the absolute error in representing it is $x\epsilon \approx 2 * 10^{-4}$. In other words, the machine epsilon between numbers around 10^12 and the next significant number is quite big compared with the base number(the element of z).

This means that we have accurancy of variance to the order 10^{-4}, which explains these two estimates agree to only 4 or 5 digits.

```
set.seed(12)
z <- rnorm(10, 0, 1)
x <- z + 1e12
formatC(var(z), 20, format = 'f')
```

```
## [1] "1.00131389554051986046"
```

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

```
## [1] "1.00132647818989228838"
```

Problem 4

(a)

Since we have exactly p cores to do the computation, even if break up Y into n individual column-wise, we can only carry out p processes at tha same time, which makes the former division meaningless.

(b)

- Approach A

For each task, we have two input matrices with dimensions of $m * n$ and $n * n$. Besides, we have calculated matrix with dimension of $m * n$. As a result, we need $2 * m * n + n * n$ of memory for each task. Since we have p tasks carried out at the mean time, we need $2n^2 + n^2 p$ of memory. Then the total communication cost is $2n^2 + n^2 p$.

- Approach B

For each task, we have two input matrices with dimensions of $m * n$ and $n * m$. Besides, we have calculated matrix with dimension of $m * m$. As a result, we need $2 * m * n + m * m$ of memory for each task. Since we have p tasks carried out at the mean time, we need $2n^2 + m^2 p = 2n^2 + mn$ of memory at the same time. Then the total communication cost is $(2n^2 + mn) * p = 2n^2 p + n^2$.

In conclusion, approach B is better for minimizing memory use and approach A is better for minimizing communication cost.