BIOST 534: Homework 1

Allen Roberts

2020-03-30

1. As mentioned in the recorded lecture, using logarithms can help reduce numerical errors when working with large values that may not be easily represented by a computer. In this example, I calculated the log of the determinant of a matrix by summing the log of the eigenvalues of the matrix. This operation is expressed in the `logdet` function shown below:

```
logdet <- function(mat) {

  return(sum(log(eigen(mat, only.values = TRUE)$values)))

}

mat <- matrix(c(2, -1, 0, -1, 2, -1, 0, -1, 2), nrow = 3, ncol = 3)

print(paste("My function's result:", logdet(mat)))
```

```
## [1] "My function's result: 1.38629436111989"
```

```
print(paste("Result using R's log(det(mat)):", log(det(mat))))
```

```
## [1] "Result using R's log(det(mat)): 1.38629436111989"
```

Using a simple 3x3 positive definite matrix, the `logdet` function returns the same answer as taking the log of the built-in `det` function in `R`. To better understand why the `logdet` function is a useful approach, I created a "bad" version of the function that takes the logarithm of the product of the eigenvalues of the matrix, as shown below:

```
logdet_bad <- function(mat) {

  return(log(prod(eigen(mat, only.values = TRUE)$values)))

}
print(logdet_bad(mat))
```

```
## [1] 1.386294
```

As the matrix dimension increases, the product of the eigenvalues will increase very rapidly, so this function may result in an overflow error for larger matrices. I created a large positive definite matrix using this method as a test case to compare different methods for calculating the log determinant.

```
set.seed(503)
n <- 1000
p <- qr.Q(qr(matrix(rnorm(n^2), n)))
large_matrix <- crossprod(p, p*(n:1))

print(paste("My function's result:", logdet(large_matrix)))
```

```
## [1] "My function's result: 5912.12817848816"
```

```
print(paste("My bad function's result:", logdet_bad(large_matrix)))
```

```
## [1] "My bad function's result: Inf"
```

```r
print(paste("Result using R's log(det(large_matrix)):",
            log(det(large_matrix))))
```

```
## [1] "Result using R's log(det(large_matrix)): Inf"
```

```r
  print(paste("Result using R's determinant(large_matrix, logarithm = TRUE):",
              determinant(large_matrix, logarithm = TRUE)$modulus[1]))
```

```
## [1] "Result using R's determinant(large_matrix, logarithm = TRUE): 5912.12817848816"
```

As shown above, my `logdet` function returns the correct determinant, as verified by using a built-in R function that also calculates the determinant on the log scale. In comparison, the version of my function that works with the product of the eigenvalues returns `Inf`, indicating that an overflow error has occurred. This also occurs if I take the log of the built-in `det` function in R.

2. Using the equation given for the marginal likelihood $p(D_1, D_A|[1|A])$, the function below calculates the log marginal likelhood for any $A \subset \{2, ..., p\}$. The function works on the additive scale using the logs of the terms in the right-hand side of the equation provided for $p(D_1, D_A|[1|A])$, thus avoiding potential numerical instability that might result when instead calculating the product of the terms. Note the function uses the `logdet` function defined in problem 1 to calculate the log determinant. The log marginal likelihood calculation specified in the problem is calculated at the end.

```r
logmarglik <- function(data, A) {

  n <- nrow(data)

  D_1 <- as.matrix(data[, 1], nrow = n)
  D_A <- as.matrix(data[, A], nrow = n)
  M_A <- diag(length(A)) + t(D_A)%*%(D_A)

  lml <- lgamma((n + length(A) + 2)/2) - lgamma((length(A) + 2)/2) -
    (1/2)*logdet(M_A) - ((n + length(A) + 2)/2) *
    log((1 + t(D_1)%*%D_1 - t(D_1)%*%D_A%*%solve(M_A)%*%t(D_A)%*%D_1))

  return(as.numeric(lml))

}

erdata <- read.table("erdata.txt")
print(paste("logmarglik(data, c(2,5,10)) = ", logmarglik(erdata, c(2,5,10))))
```

```
## [1] "logmarglik(data, c(2,5,10)) =  -59.978932648833"
```