COMPUTER LAB 1 Group report

Andrea Bruzzone,Thomas Zhang

2016 M02 3

Assignment 1

The result of the program wrote by a student is:

```r
x1 <- 1/3
x2 <- 1/4
if(x1-x2 == 1/12){
  print("Teacher said true")
}else{
  print("Teacher lied")
}
```

```
## [1] "Teacher lied"
```

However, the result should be "Teacher said true", since 1/3 - 1/4 = 1/12. I imagine this is due to the finite nature of the significand of floating point numbers.

In order to get a correct result, we should round both the difference $x1 - x2$ and 1/12 with the same decimal places, it seems that using less or equal than 16 decimal places the result is correct. We chose to round with 4 decimal places.

The correct code should be like:

```r
x1 <- 1/3
x2 <- 1/4
if(round(x1-x2, 4)==round(1/12, 4)){
  print("Teacher said true")
}else{
  print("Teacher lied")
}
```

```
## [1] "Teacher said true"
```

Assignment 2

Using:

$$f'(x) = \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

we compute the derivative of f(x) = x.

For `x = 100000` and `epsilon = 10^-15`,we implement this in R below.

```r
derivative <- function(x,epsilon){
  res <- ((x + epsilon) - x) / epsilon
  return(res)
}

derivative(x = 100000, epsilon = 10^-15)
```

```
## [1] 0
```

The obtained value is zero, while the real value is one. We have a wrong value because ϵ is much smaller than x, so $x + \epsilon$ give a result which is rounded to x in floating-point arithmetics and so the numerator is 0, resulting in the derivative to be 0.

When we try a larger epsilon (a smaller difference in magnitude between $x + \epsilon$ and x) the obtained value will be one. The numerical breakdown happens when order difference is around 16.

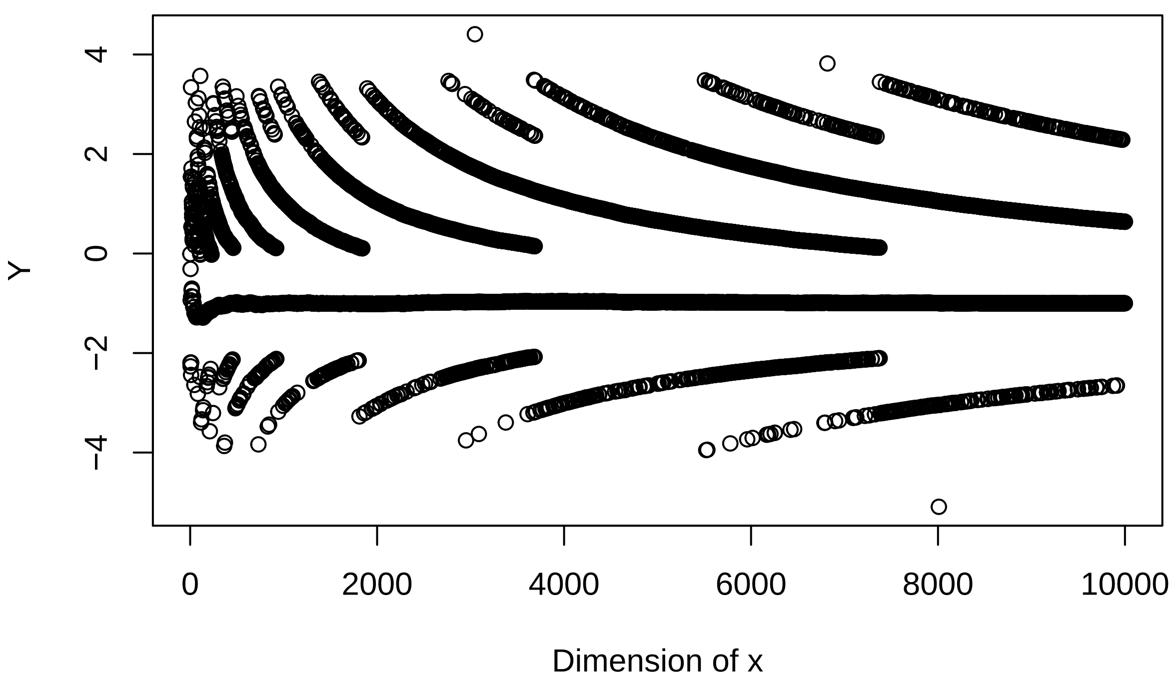
```r
derivative(x = 100000, epsilon = 10^-5)
```

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

Assignment 3

We write a function myvar estimating the variance, generate a vector $x = (x_1, .., x_{10000})$ with 10000 random numbers, normally distributed with mean 10^8 and variance 1. For each subset $X_i = (x_1, .., x_i)$ with $i = 1...10000$ we compute the difference Y_i between our function and the R function var.

We plot this difference against the dimension of the subset:



From the plot we can see that the difference is never zero so `myvar` and the function `var` never give the same value. The reason could be catastrophic cancellation, since we are adding two numbers with almost equal magnitude but opposite sign, this brings a high probability of roundoff error, considering the narrowness of the standard deviation of the $x_i s$, (They are $\mathcal{N}(10^8, 1)$ distributed).

Moreover, if we print the results for the subset $X_i = (x_1, .., x_i)$ with $i = 1...10$ for the `myvar` function and the `var` function respectively we see that our function works quite bad and this could be due to the catastrophic cancelltion problem.

`myvar`:

```
## [1] 0
## [1] -2
## [1] 0
## [1] -2
## [1] 0
## [1] -1.333333
## [1] 2.285714
## [1] 4
## [1] -1.777778
```

`var`:

```
## [1] 0.007680213
## [1] 0.1947559
## [1] 0.3099219
## [1] 0.2682
## [1] 0.9394953
## [1] 0.8549428
## [1] 0.7438822
## [1] 0.658902
## [1] 0.6620352
```

Assignment 4

We want to solve, in the sense of least squares the linear regression system $A\beta = b$ where $A = X'X$ are derived from predictors from the `tecator.xls` data sheet and $b = X'Y$ is derived from one column chosen as response. We will try to use the built-in `solve` function.

```
X <- as.matrix(data[,-c(1,102)])
#head(X)
Y <- as.matrix(data[,102,drop=FALSE])
A <- t(X) %*% X
b <- t(X) %*% Y
solveres <- solve(A,b) # can not solve the system is ill-conditioned
rcond(A) # rcond returns 7.25 * 10^(-17)
```

Due to at least two columns in the X matrix being almost linearly dependent, the condition number, the inverse of the number produced by `rcond`, of `A` is large. Since the `solve` function seems not to work because the condition number of `A` is too large, we will try to scale the data in hope it will reduce condition number and make `solve` work.

```
datascaled <- scale(data)

#head(data)
Xscaled <- as.matrix(datascaled[,-c(1,102)])
#head(X)
```

```
Yscaled<- as.matrix(datascaled[,102,drop=FALSE])
A <- t(Xscaled) %*% Xscaled
b <- t(Xscaled) %*% Yscaled
solveres <- solve(A,b) #now it works. scaling appears to improve rcond a little
rcond(A) # 7.0 * 10^(-14)
solveres
```

The condition number is smaller now and we are lucky that it works now. Generally we should probably decompose the matrix `A` using QR or SVD or Cholesky decomposition/factorization.

Appendix

R code

```
x1 <- 1/3
x2 <- 1/4
if(x1-x2 == 1/12){
  print("Teacher said true")
}else{
  print("Teacher lied")
}
x1 <- 1/3
x2 <- 1/4
if(round(x1-x2, 4)==round(1/12, 4)){
  print("Teacher said true")
}else{
  print("Teacher lied")
}
derivative <- function(x,epsilon){
  res <- ((x + epsilon) - x) / epsilon
  return(res)
}

derivative(x = 100000, epsilon = 10^-15)
derivative(x = 100000, epsilon = 10^-5)
set.seed(12345)
x <- rnorm(10000, mean = 10^8, sd = 1)
myvar <- function(x){
  v <- 1/(length(x) - 1) * (sum(x^2) - 1/length(x) * (sum(x)^2))
  return(v)
}

y <- c()
for(i in 1:10000){
  sub <- x[1:i]
  y[i] <- myvar(sub) - var(sub)
}

plot(1:10000, y, xlab = "Dimension of x", ylab = "Y")
for(i in 2:10){
print(myvar(x[1:i]))
```

```
}
for(i in 2:10){
print(var(x[1:i]))
}
## X <- as.matrix(data[,-c(1,102)])
## #head(X)
## Y <- as.matrix(data[,102,drop=FALSE])
## A <- t(X) %*% X
## b <- t(X) %*% Y
## solveres <- solve(A,b) # can not solve the system is ill-conditioned
## rcond(A) # rcond returns 7.25 * 10^(-17)
## datascaled <- scale(data)
##
## #head(data)
## Xscaled <- as.matrix(datascaled[,-c(1,102)])
## #head(X)
## Yscaled<- as.matrix(datascaled[,102,drop=FALSE])
## A <- t(Xscaled) %*% Xscaled
## b <- t(Xscaled) %*% Yscaled
## solveres <- solve(A,b) #now it works. scaling appears to improve rcond a little
## rcond(A) # 7.0 * 10^(-14)
## solveres
## NA
```