Lab 1 - Computational Statistics

Gustav Sternelöv

Friday, February 5, 2016

Assignment 1

1.1

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

As can be seen by the result obtained from the program, wrong conclusion is given.

This happened because the floating points of the different values may not be the exact values. This causes problems with the the round-off for the compared values. *x1* subtracted by *x2* should be the same as *1/12*, but since the exact values of *1/3* and/or *1/4* not is represented by *x1* and *x2* the subtraction of the values will not become exactly *1/12*.

1.2

There are some different options available for solving this problem. One alternative is to specify the number of decimals for the values that are compared. Another option is to specify the tolerance, meaning how far from *1/12* do we allow the subtraction of *x1* and *x2* to be.
In the code below the first option, specifying the number of decimals, is tried out and as can be seen the output from the program now generates the correct conclusion.

```
x1<-1/3
x2<-1/4
if (round(x1-x2,8)==round(1/12,8)){
  print("Teacher said true")
} else{
  print("Teacher lied")
}
```

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

Assignment 2

2.1

```
AssigTwo <- function(x, epsilon){
  f_prim <- ((x+epsilon) - x)/ epsilon
  return(f_prim)
}
```

2.2

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

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

2.3

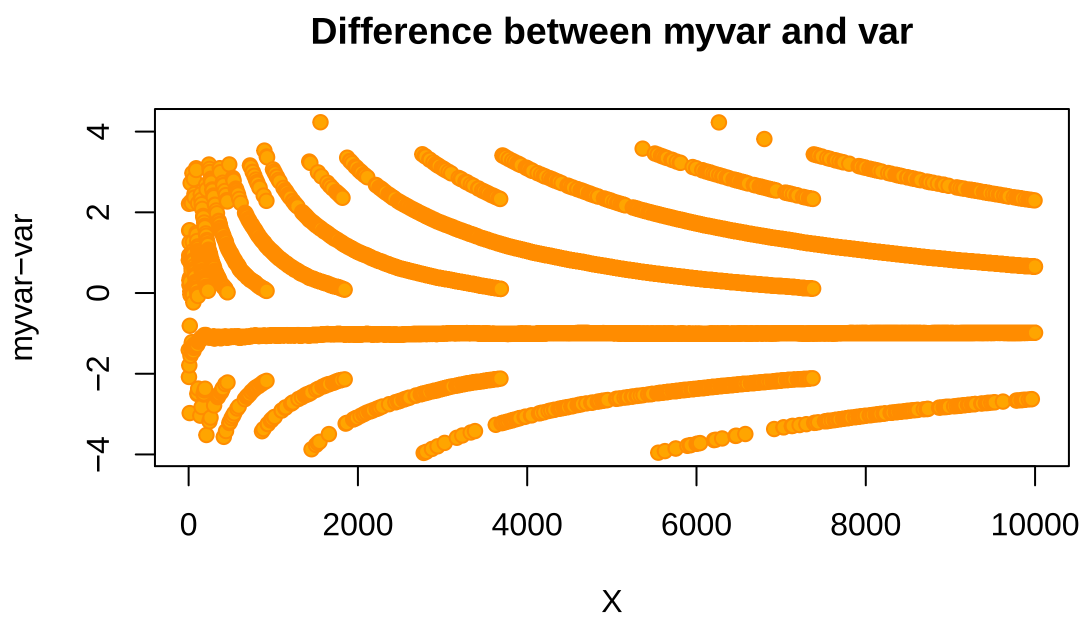
The obtained value is *0* but the real value of the derivative is *1*. The reason behind this error is connected to the different magnitudes. Here x is a very large value and *epsilon* is a very low value. Therefore the addition of x+*epsilon* practically becomes x which means that in the numerator x-x is the executed computation in practice. The value in the denominator is equal to the value represented by *epsilon* since it does not interfer with any magnitude problems. This leads to a division where the value of x-x is divided by *epsilon*, which of course becomes *0*.

Assignment 3

3.1-3.2

```
myvar <- function(x, n){
 varX <-  1/(n-1) * (sum(x^2) - (1/n)*(sum(x)^2))
 return(varX)
}
randVec <- rnorm(n = 10000, mean = 10^8, sd = 1)
```

3.3



There appear to be some differences between how the respective functions estimates the variance. Otherwise, if the functions were more similar, the differences plotted by the graph above should have been closer to zero. One possible explanation to this results is the problems that may occur when the sum of several values with the same sign and approximately same magnitude is computed. Probably *var* has a better solution to this problem than *myvar*, hence the differences between the estimated variances.

Assignment 4

4.1-4.3

```
tecat <- read.csv("C:/Users/Gustav/Documents/Machine-Learning/Lab 2/tecator.csv", sep = ";")
# Protein, column 103 is the target variable
Xvar <- data.frame(x1=1, tecat[,-c(1,103)])
A <- t(as.matrix(Xvar)) %*% as.matrix(Xvar)
b <- t(as.matrix(Xvar)) %*% as.matrix(tecat$Protein)
```

```
solve(A,b)
```

The result given is only a error message, so some numerical error seem to occur when running *solve* on the matrices:
"*Error in solve.default(A, b) :*
system is computationally singular: reciprocal condition number = 7.78804e-17"
Then, as has been noted, the *beta* coefficients are not recieved because of the numerical error. This error occurs because the matrix A is singular, hence the inverse cannot be computed.

4.4

The condition number for the matrix A is obtained by using the function *kappa*.

```
kappa(A)
```

```
## [1] 8.523517e+14
```

In the earlier step, *4.3*, it was concluded that the matrix A is singular and that this affects the linear system negatively since the inverse cannot be computed. The obtained condition number is considered to be quite large and this is is rather expected since the condition number measure the how big the error of the computations could be in the worst-case scenario. That the error might be large is understandable as it has been discovered that there might be some issues with the A matrix.

4.5

```
tecatSc <- data.frame(scale(x = tecat, center = TRUE, scale = TRUE))
XvarXc <- data.frame(x=1, tecatSc[,-c(1,103)])
ASc <- t(as.matrix(XvarXc)) %*% as.matrix(XvarXc)
bSc <- t(as.matrix(XvarXc)) %*% as.matrix(tecatSc$Protein)

# solve(ASc,bSc)
# The condition number is given by
# kappa(ASc)
```

When applying *solve* on the scaled matrices the *beta* coefficients for the respective x-variables are obtained. The problem that occured for the matrix A in *4.3* with non-scaled data does not occur for the scaled data. In the table below the first ten coefficients are shown.

```
##                   [,1]
## x        -1.872906e-12
## Channel1 -1.106420e+02
## Channel2 -2.211900e+02
## Channel3  3.780067e+02
## Channel4 -1.297038e+02
## Channel5  4.134180e+02
## Channel6 -7.975199e+01
## Channel7 -2.030060e+02
## Channel8  8.279496e+01
## Channel9 -1.323811e+02
```

The condition number for the scaled data is 4.9047152×10^{11}, hence the result has changed because of the scaling of data. This data set could be sensitive to scaling since there is a signficant difference between the magnitude of the values for the predictors, especially the variables *fat* and *moisture* has much higher values than the others. The new condition number is lower and an interpretation of this could be that the scaled linear system performs better than the non-scaled. However, this does not necesserily need to be true. The lower condition number could be due to the scaling and moreover does not a lower condition number guarantee that the new system is better. A lower number is preferable but it does not completely assures that itis better. For a linear system the condition provides a bound on the relative norms for the "correct" solution. Since it specifies the bound a lower value is preferable but a higher number does not have to mean that the linear system is worse.

Appendix

```
### Assignment 1 ###
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-x2,8)==round(1/12,8)){
  print("Teacher said true")
} else{
  print("Teacher lied")
}
AssigTwo <- function(x, epsilon){
  f_prim <- ((x+epsilon) - x)/ epsilon
  return(f_prim)
}
AssigTwo(x=100000, epsilon=10^-15)
myvar <- function(x, n){
 varX <-  1/(n-1) * (sum(x^2) - (1/n)*(sum(x)^2))
 return(varX)
}
randVec <- rnorm(n = 10000, mean = 10^8, sd = 1)
y <- 0
for(i in 1:10000){
  y[i] <- myvar(randVec[1:i], n=i) - var(randVec[1:i])
}
plot(y=y[2:10000], x=2:10000, pch=21, bg="orange", col="darkorange", ylab="myvar-var", xlab="X",
     main="Difference between myvar and var")
tecat <- read.csv("C:/Users/Gustav/Documents/Machine-Learning/Lab 2/tecator.csv", sep = ";")
# Protein, column 103 is the target variable
Xvar <- data.frame(x1=1, tecat[,-c(1,103)])
A <- t(as.matrix(Xvar)) %*% as.matrix(Xvar)
b <- t(as.matrix(Xvar)) %*% as.matrix(tecat$Protein)
## solve(A,b)
kappa(A)
tecatSc <- data.frame(scale(x = tecat, center = TRUE, scale = TRUE))
XvarXc <- data.frame(x=1, tecatSc[,-c(1,103)])
ASc <- t(as.matrix(XvarXc)) %*% as.matrix(XvarXc)
bSc <- t(as.matrix(XvarXc)) %*% as.matrix(tecatSc$Protein)

# solve(ASc,bSc)
# The condition number is given by
# kappa(ASc)
head(solve(ASc,bSc), 10)
##
```