Exercies 6 - Alon Goodman & Ran Hassid

Q1

a

$$y_i = 1 + \beta_1 e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}} + \varepsilon_i \quad , \quad E(\varepsilon_i) = 0 \quad Var(\varepsilon_i) =$$

$$\frac{\partial y_i}{\partial \beta_1} = e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}} + \frac{\beta_1 x_{1i}}{\beta_2 x_{2i}} e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}}$$

$$\frac{\partial y_i}{\partial \beta_2} = \beta_1 e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}} \cdot \frac{-x_{2i}}{(\beta_2 x_{2i})^2} = \frac{-\beta_1^2 x_{1i}}{\beta_2^2 x_{2i}} \cdot e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}}$$

$$W_{n\times 2} = \begin{pmatrix} \frac{\partial y_1}{\partial \beta_1}(x_{11}, x_{21}) & \frac{\partial y_1}{\partial \beta_2}(x_{11}, x_{21}) \\ \vdots & \vdots \\ \frac{\partial y_1}{\partial \beta_1}(x_{1n}, x_{2n}) & \frac{\partial y_1}{\partial \beta_2}(x_{1n}, x_{2n}) \end{pmatrix}_{n\times 2} = \begin{pmatrix} e^{\frac{\beta_1 x_{11}}{\beta_2 x_{21}}} (1 + \\ \vdots \\ e^{\frac{\beta_1 x_{1n}}{\beta_2 x_{2n}}} (1 + \end{pmatrix}$$

$$e_i^{(j)} = y_i - f(\beta_j, x_i) = y_i - \left(1 + \beta_1^{(j)} e^{\frac{\beta_1^{(j)} x_{1i}}{\beta_2^{(j)} x_{2i}}}\right)$$

$$\beta^{(j+1)} = \beta^{(j)} + \left[W^{(j)T} \cdot W^{(j)} \right]^{-1} \cdot W^{(j)T} \cdot e^{(j)}$$

```r
# ex6data1 <- read.csv("C:/Users/Alon/Desktop/Studies/Statistics/Statistical_Computing/Exercises/HW6/ex

ex6data1 <- read.csv("~/Desktop/Ran/D year/semester b/hishov statisti/exercies/HW6/ex6data1.csv")

graduent.func <- function(beta1,beta2,x1_vec,x2_vec){
  ### input: beta1,beta2. output: the derivative for beta0 and beta1
  beta1_derivative <- ( 1 + (beta1*x1_vec)/(beta2*x2_vec) ) * exp((beta1*x1_vec)/(beta2*x2_vec))
  beta2_derivative <- - ( (beta1*beta1*x1_vec)/(beta2*beta2*x2_vec) ) * exp((beta1*x1_vec)/(beta2*x2_ve
  return(matrix(data = c(beta1_derivative,beta2_derivative),ncol = 2))
}

beta1.j <- beta2.j <- 1 # first guess
y_vec.Q1 <- ex6data1$y
x1_vec.Q1 <- ex6data1$x1
x2_vec.Q1 <- ex6data1$x2

eps_for_break <- 10^-7

while (TRUE) {
  errors.j <- y_vec.Q1 - 1 - beta1.j*exp((beta1.j*x1_vec.Q1)/(beta2.j*x2_vec.Q1))
  betas.j <- matrix(data = c(beta1.j,beta2.j),nrow = 2,ncol = 1)
  W.j <- graduent.func(beta1.j,beta2.j,x1_vec.Q1,x2_vec.Q1)
  betas.j_1 <- betas.j + ( solve(t(W.j)%*%W.j) %*% t(W.j) ) %*% errors.j
  beta1.j <- betas.j_1[1]
  beta2.j <- betas.j_1[2]
  if (max(abs(betas.j_1 - betas.j)) < eps_for_break) {break}
}
```

```
## beta1 (GN) = 21.023
## beta2 (GN) = 44.157
```

b

$$S(\beta_1,\beta_2) = \sum_{i=1}^{n} \left(y_i - F(x_{1i}, x_{2i})\right)^2 = \sum_{i=1}^{n} (y_i - 1 - \beta_1$$

$$\frac{\partial S}{\partial \beta_1} = -2 \cdot \sum_{i=1}^{n} \left(y_i - 1 - \beta_1 e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}}\right) \cdot e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}} \left(1 + \frac{\beta_1 x}{\beta_2 x}\right.$$

$$\frac{\partial S}{\partial \beta_2} = 2 \cdot \sum_{i=1}^{n} \left(y_i - 1 - \beta_1 e^{\frac{\beta_1 x_{1i}}{\beta_2 x_{2i}}}\right) \frac{\beta_1^2 x_{1i}}{\beta_2^2 x_{2i}} \cdot e^{\frac{\beta_1 x}{\beta_2 x_2}}$$

```r
delta <- 10^-4

grad_s <- function(beta1,beta2,x1_vec,x2_vec,y_vec){
  d_s_d_beta1 <- - 2 * sum( ( y_vec - 1 - beta1*exp((beta1*x1_vec)/(beta2*x2_vec)) ) * exp((beta1*x1_ve

  d_s_d_beta2 <- 2 * sum( ( y_vec - 1 - beta1*exp((beta1*x1_vec)/(beta2*x2_vec)) ) * exp((beta1*x1_vec)/

    return(matrix(data = c(d_s_d_beta1,d_s_d_beta2),nrow = 1,ncol = 2))
}

hesian.Q1 <- function(beta1,beta2,x1_vec,x2_vec,y_vec,delta){
  y_d2_beta1 <- grad_s(beta1+delta,beta2,x1_vec,x2_vec,y_vec) - grad_s(beta1-delta,beta2,x1_vec,x2_vec,y

  y_d2_beta1 <- y_d2_beta1[1] / (2*delta)

  y_d2_beta2 <- grad_s(beta1,beta2+delta,x1_vec,x2_vec,y_vec) - grad_s(beta1,beta2-delta,x1_vec,x2_vec,y

  y_d2_beta2 <- y_d2_beta2[2] / (2*delta)

  y_d_beta1_d_beta2 <- grad_s(beta1,beta2+delta,x1_vec,x2_vec,y_vec) - grad_s(beta1,beta2-delta,x1_vec,x

  y_d_beta1_d_beta2 <- y_d_beta1_d_beta2[1] / (2*delta)

  return(matrix(data = c(y_d2_beta1,y_d_beta1_d_beta2,y_d_beta1_d_beta2,y_d2_beta2),nrow = 2,ncol = 2))
```

```
  }

beta1_h.j <- beta2_h.j <- 1 # first guess
y_vec.Q1 <- ex6data1$y
x1_vec.Q1 <- ex6data1$x1
x2_vec.Q1 <- ex6data1$x2

counter <- 0

eps_for_break <- 10^-5

while (TRUE) {
  counter <- counter+1
  betas_h.j <- matrix(c(beta1_h.j,beta2_h.j),ncol = 1)
  betas_h.j_1 <- betas_h.j - solve(hesian.Q1(beta1_h.j,beta2_h.j,x1_vec.Q1,x2_vec.Q1,y_vec.Q1,delta)) %

  beta1_h.j <- betas_h.j_1[1]
  beta2_h.j <- betas_h.j_1[2]
  if (max(abs(betas_h.j_1 - betas_h.j)) < eps_for_break) {break}
}
```

```
## beta1 (NR) = 21.023
## beta2 (NR) = 44.157
```

We can see that the both methods give us same beta1 & beta2.

Q2

a

[illegible] ; תלוי באירוע מסוים

$y_i | x_i \sim Pois(\lambda_i)$. X_i [illegible] שאנו

$$\ln(\lambda_i) = \beta_0 + \beta_1 X_i \Rightarrow \lambda_i = e^{\beta_0 + \beta_1 X_i}$$

$$\Rightarrow y_i | x_i \sim Pois\left(e^{\beta_0 + \beta_1 X_i}\right)$$

$$f_{y_i|x_i}(y) = \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!} = \frac{e^{-e^{\beta_0 + \beta_1 X_i}} \cdot e^{y_i(\beta_0 + \beta_1 X_i)}}{y_i!}$$

$$L(y, \beta_0, \beta_1) = \prod_{i=1}^{n} f_{y_i|x_i}(y) = \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!} = \prod_{i=1}^{n} -$$

$$= \frac{e^{-\sum_{i=1}^{n} e^{\beta_0 + \beta_1 X_i}} \cdot e^{\sum y_i(\beta_0 + \beta_1 x_i)}}{\prod_{i=1}^{n} y_i!}$$

b

$$\ell(y,\beta_0,\beta_1) = \ln(L(y,\beta_0,\beta_1)) = -\sum_{i=1}^{n} e^{\beta_0+\beta_1 x_i} + \sum_{i=1}^{n} y$$

$$S(\beta_0,\beta_1) = \begin{pmatrix} \frac{\partial \ell(y,\beta_0,\beta_1)}{\partial \beta_0} \\ \frac{\partial \ell(y,\beta_0,\beta_1)}{\partial \beta_1} \end{pmatrix} = \begin{pmatrix} -\sum_{i=1}^{n} e^{\beta_0+\beta_1 x_i} + \sum_{i=1}^{n} y \\ -\sum_{i=1}^{n} x_i \cdot e^{\beta_0+\beta_1 x_i} + \sum_{i=1} \end{pmatrix}$$

$\cdot \beta_1 \cdot x_i \quad \beta_0$

c + d

ההסיאן:

$$\frac{\partial^2 \ell(y,\beta_0,\beta_1)}{\partial^2 \beta_0} = \frac{\partial\left(-\sum_{i=1}^{n} e^{\beta_0+\beta_1 x_i} + \sum_{i=1}^{n} y_i\right)}{\partial \beta_0} = -\sum_{i=1}^{n} e^{\beta_0+\beta_1 x_i}$$

$$\frac{\partial^2 \ell(y,\beta_0,\beta_1)}{\partial \beta_0 \partial \beta_1} = \frac{\partial\left(-\sum_{i=1}^{n} e^{\beta_0+\beta_1 x_i} + \sum_{i=1}^{n} y_i\right)}{\partial \beta_1} = -\sum_{i=1}^{n} x_i e^{\beta_0+\beta_1 x_i}$$

$$\frac{\partial^2 \ell(y,\beta_0,\beta_1)}{\partial^2 \beta_1} = \frac{\partial\left(-\sum_{i=1}^{n} x_i \cdot e^{\beta_0+\beta_1 x_i} + \sum_{i=1}^{n} x_i y_i\right)}{\partial \beta_1} = -\sum x_i^2 e^{\beta_0+\beta_1 x_i}$$

$$H(\beta_0,\beta_1) = \begin{pmatrix} -\sum_{i=1}^{n} e^{\beta_0+\beta_1 x_i} & -\sum_{i=1}^{n} x_i e^{\beta_0+\beta_1 x_i} \\ -\sum_{i=1}^{n} x_i e^{\beta_0+\beta_1 x_i} & -\sum x_i^2 e^{\beta_0+\beta_1 x_i} \end{pmatrix}$$

הפתרון באיטרציה הבאה (j+1) נתון ע"י הנוסחה:

$$\begin{pmatrix} \beta_0^{(j+1)} \\ \beta_1^{(j+1)} \end{pmatrix} = \begin{pmatrix} \beta_0^{(j)} \\ \beta_1^{(j)} \end{pmatrix} - H(\beta_0^{(j)},\beta_1^{(j)})^{-1} \cdot S(\beta_0^{(j)},\beta_1^{(j)})^T =$$

$$= \begin{pmatrix} \beta_0^{(j)} \\ \beta_1^{(j)} \end{pmatrix}_{2\times 1} - \begin{pmatrix} -\sum_{i=1}^{n} e^{\beta_0^j+\beta_1^j x_i} & -\sum_{i=1}^{n} x_i e^{\beta_0^j+\beta_1^j x_i} \\ -\sum_{i=1}^{n} x_i e^{\beta_0^j+\beta_1^j x_i} & -\sum x_i^2 e^{\beta_0^j+\beta_1^j x_i} \end{pmatrix}^{-1}_{2\times 2} \cdot \begin{pmatrix} -\sum_{i=1}^{n} e \\ -\sum_{i=1}^{n} x_i \end{pmatrix}$$

```r
ex6_count_data <- read.csv("~/Desktop/Ran/D year/semester b/hishov statisti/exercies/HW6/count_data.csv

y_vec.Q2 <- ex6_count_data$y
x_vec.Q2 <- ex6_count_data$x
```

```r
grad_s.Q2 <- function(beta0,beta1,x_vec,y_vec){

  d_log_like_d_beta0 <- - sum(exp(beta0 + beta1 * x_vec)) + sum(y_vec)

  d_log_like_d_beta1 <- - sum(x_vec * (exp(beta0 + beta1 * x_vec))) +
    sum(x_vec * y_vec)

  return(matrix(data = c(d_log_like_d_beta0,d_log_like_d_beta1),
                nrow = 2,ncol = 1))

}
```

```r
hesian.Q2 <- function(beta0,beta1,x_vec){

  s_d2_beta0 <- - sum(exp(beta0 + beta1 * x_vec))

  s_d_beta0_d_beta1 <- - sum(x_vec * (exp(beta0 + beta1 * x_vec)))

  s_d2_beta1 <- - sum(x_vec^2 * (exp(beta0 + beta1 * x_vec)))

  return(matrix(data = c(s_d2_beta0,s_d_beta0_d_beta1,s_d_beta0_d_beta1,s_d2_beta1),
                 nrow = 2,ncol = 2))

}
```

```r
beta0_h.j <- beta1_h.j <- 1 # first guess

eps_for_break <- 10^-5

while (TRUE) {

  betas_h.j <- matrix(c(beta0_h.j,beta1_h.j),ncol = 1)

  betas_h.j_1 <- betas_h.j - solve(hesian.Q2(beta0_h.j,beta1_h.j,x_vec.Q2)) %*%
    (grad_s.Q2(beta0_h.j,beta1_h.j,x_vec.Q2,y_vec.Q2))

  beta0_h.j <- betas_h.j_1[1]
  beta1_h.j <- betas_h.j_1[2]

  if (max(abs(betas_h.j_1 - betas_h.j)) < eps_for_break) {break}
}
```

```
## beta0 (NR) = 0.74184
## beta1 (NR) = 0.21007
```

```r
model <- glm(formula = y_vec.Q2~x_vec.Q2, family = "poisson")
betas_glm <- summary(model)
```

```
beta0_glm <-  betas_glm$coefficients[1]
beta1_glm <-  betas_glm$coefficients[2]
```

```
## beta0 (glm) = 0.74184
## beta1 (glm) = 0.21007
```

We can see that the both methods give us same beta0 & beta1.

```
sd_beta0_glm <-  betas_glm$coefficients[3]
sd_beta1_glm <-  betas_glm$coefficients[4]
```

```
## beta0 sd (glm) = 0.04488
## beta1 sd (glm) = 0.02895
```

e

The Information matrix (-E[Hesian]) is not a function of a random variable so NR algorithm and FS algorithm are same.

Q5

a

$$X \sim \text{Gompertz} \Rightarrow f_X(x) = b\eta \cdot e^{\eta + bx - \eta e^{bx}}$$

$$L(x, \eta, b) = \prod_{i=1}^{n} f_X(x_i) = \prod_{i=1}^{n} b\eta \cdot e^{\eta + bx_i - \eta e^{bx_i}}$$

$$l(x, \eta, b) = \ln(L(x, \eta, b)) = n(\ln(b) + \ln(\eta)) + \sum(\eta$$

$$= n(\ln(b) + \ln(\eta)) + n\eta + b\sum x_i - \eta \sum e^{bx_i} =$$

$$= n\ln(b) + n\ln(\eta) + n\eta + b\sum x_i - \eta \sum e^{bx_i}$$

$$s(\eta, b) = \begin{pmatrix} \frac{\partial l(x,\eta,b)}{\partial \eta} \\ \frac{\partial l(x,\eta,b)}{\partial b} \end{pmatrix} = \begin{pmatrix} \frac{n}{\eta} + n - \sum e^{bx_i} \\ \frac{n}{b} + \sum x_i - \eta \sum x_i e^{bx_i} \end{pmatrix}$$

$$\frac{\partial^2 l(x,\eta,b)}{\partial^2 \eta} = \frac{\partial \frac{n}{\eta} + n - \sum e^{bx_i}}{\partial \eta} = -\frac{n}{\eta^2}$$

$$\frac{\partial^2 l(x,\eta,b)}{} \quad \partial \frac{n}{\eta} + n - \sum e^{bx_i} \quad \sum x_i e^{bx_i}$$

b

```
ex6data2 <- read.csv("~/Desktop/Ran/D year/semester b/hishov statisti/exercies/HW6/ex6data2.csv")

x_vec.Q5 <- ex6data2$x
```

```
grad_s.Q5 <- function(eta,b,x_vec){

  n = length(x_vec)

  d_log_like_d_eta <- n/eta + n - sum(exp(b * x_vec))

  d_log_like_d_b <- n/b + sum(x_vec) - eta * sum(x_vec * exp(b * x_vec))

  return(matrix(data = c(d_log_like_d_eta,d_log_like_d_b),
              nrow = 2,ncol = 1))

}
```

```
hesian.Q5 <- function(eta,b,x_vec){

  n = length(x_vec)

  s_d2_eta <- - (n / eta^2)

  s_d_eta_d_b <- - sum(x_vec * exp(b * x_vec))

  s_d2_b <- - (n / b^2) - eta * sum(x_vec^2 * exp(b * x_vec))

  return(matrix(data = c(s_d2_eta,s_d_eta_d_b,s_d_eta_d_b,s_d2_b),
                nrow = 2,ncol = 2))

}
```

```
NR.Q5 <- function(eta0,b0,x_vec,eps_for_break){

  eta_h.j <- eta0
  b_h.j <- b0

  while (TRUE) {

    eta_b_h.j <- matrix(c(eta_h.j,b_h.j),ncol = 1)

    eta_b_h.j_1 <- eta_b_h.j - solve(hesian.Q5(eta_h.j,b_h.j,x_vec)) %*%
      (grad_s.Q5(eta_h.j,b_h.j,x_vec))

    eta_h.j <- eta_b_h.j_1[1]
    b_h.j <- eta_b_h.j_1[2]

    if (max(abs(eta_b_h.j_1 - eta_b_h.j)) < eps_for_break) {break}
  }
```

```r
  return(cat(sprintf("The maximum likelihood estimator of eta (NR) = %s",
                     round(eta_h.j,5)),
    sprintf("The maximum likelihood estimator of b (NR) = %s",
            round(b_h.j,5)),
    sep = "\n"))
}
```

```r
NR.Q5(eta0 = 1,b0 = 1,x_vec = x_vec.Q5,eps_for_break = 10^-5 )
```

```
## The maximum likelihood estimator of eta (NR) = 5.07592
## The maximum likelihood estimator of b (NR) = 2.11706
```

Q6

b

b.1 - Gradient Descent

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

$$\frac{\partial f(x,y)}{\partial x} = -2(1-x) - 400x(y-x^2)$$

$$\frac{\partial f(x,y)}{\partial y} = 200(y-x^2)$$

$$\begin{pmatrix} x^{j+1} \\ y^{j+1} \end{pmatrix} = \begin{pmatrix} x^{j} \\ y^{j} \end{pmatrix} - \gamma^{j} \Bigg(\begin{matrix} -2(1-x^{j}) - 400 \cdot x \\ 200(y^{j} \end{matrix}$$

```r
f.Q6 <- function(x,y){
  return((1 - x)^2 + 100 * (y - (x^2))^2)
}
```

```r
grad_f.Q6 <- function(x,y){

  df_dx <-  400 * (x^3) - 400*x*y + 2*x -2
  df_fy <-  200*(y-(x^2))

  return(matrix(data=c(df_dx,df_fy),nrow = 2,ncol = 1))

}
```

```r
gamma.j.Q6 <- function(alpha,beta,gamma,x,y) {
  x_y.j_1 <- c(x,y) - gamma * grad_f.Q6(x,y) # x,y next value
  f.j_1 <- f.Q6(x_y.j_1[1],x_y.j_1[2]) # f.next
  f.j <- f.Q6(x,y) # f.now
    while (f.j_1 > f.j - alpha * gamma * t(grad_f.Q6(x,y))%*%(grad_f.Q6(x,y))) {
      gamma <- beta * gamma
      x_y.j_1 <- c(x,y) - gamma * grad_f.Q6(x,y) # x,y next value
      f.j_1 <- f.Q6(x_y.j_1[1],x_y.j_1[2]) # f.next
    }
  return(gamma)
}
```

```r
Gradient_Descent.Q6 <- function(x,y,alpha,beta,eps_for_break){

  x.j <- x
  y.j <- y
  counter <-  0


  while (TRUE) {

    counter <- counter + 1

    x_y.j <- matrix(c(x.j,y.j),ncol = 1) # the current solution

    f.j <- f.Q6(x.j,y.j) # the current function value
    gamma.j <- gamma.j.Q6(alpha,beta,gamma = 1,x.j,y.j) # compute the gamma value for the j iteration
    x_y.j_1 <- x_y.j - gamma.j * grad_f.Q6(x.j,y.j) # the next solution
    f.j_1 <- f.Q6(x_y.j_1[1],x_y.j_1[2]) # the next function value

    x.j <- x_y.j_1[1]
    y.j <- x_y.j_1[2]

    if(max(abs(f.j_1 - f.j)) < eps_for_break){break}
  }

  return(cat(sprintf("x (Gradient Descent) = %s", x.j),
    sprintf("y (Gradient Descent) = %s", y.j),
    sprintf("counter (Gradient Descent) = %s", counter),
    sep = "\n"))
}
```

```r
Gradient_Descent.Q6(x = 1.01, y = 1.01, alpha = 0.5, beta = 0.5, eps_for_break = 10^-8)
```

```
## x (Gradient Descent) = 1.00152697954938
## y (Gradient Descent) = 1.00306252228773
## counter (Gradient Descent) = 56
```

```r
Gradient_Descent.Q6(x = 2, y = 2, alpha = 0.5, beta = 0.5, eps_for_break = 10^-8)
```

```
## x (Gradient Descent) = 1.00210923493955
## y (Gradient Descent) = 1.0042312553076
## counter (Gradient Descent) = 2655
```

```r
Gradient_Descent.Q6(x = 0, y = 0, alpha = 0.5, beta = 0.5, eps_for_break = 10^-8)
```

```
## x (Gradient Descent) = 0.997341177892756
## y (Gradient Descent) = 0.994678936738671
## counter (Gradient Descent) = 1023
```

```r
Gradient_Descent.Q6(x = 10, y = 10, alpha = 0.5, beta = 0.5, eps_for_break = 10^-8)
```

```
## x (Gradient Descent) = 1.00177441897637
## y (Gradient Descent) = 1.00356318223124
## counter (Gradient Descent) = 17239
```

```r
Gradient_Descent.Q6(x = -105, y = 20, alpha = 0.5, beta = 0.5, eps_for_break = 10^-8)
```

```
## x (Gradient Descent) = 0.998819985844088
## y (Gradient Descent) = 0.997636421537645
## counter (Gradient Descent) = 27604
```

b.2 - NR

הלני

$$\frac{\partial^2 f(x,y)}{\partial^2 x} = 2 - 400(y - x^2) + 800x^2 = 2 - 400y + 1200\ x$$

$$\frac{\partial^2 f(x,y)}{\partial x \partial y} = -400x = \frac{\partial^2 f(x,y)}{\partial y \partial x}$$

$$\frac{\partial^2 f(x,y)}{\partial^2 y} = 200$$

$$\begin{pmatrix} x^{j+1} \\ y^{j+1} \end{pmatrix} = \begin{pmatrix} x^{j} \\ y^{j} \end{pmatrix} - \begin{pmatrix} 2 - 400y^j + 400x^{j^2} & -400x^j \\ -400x^j & 200 \end{pmatrix}^{-1} \cdot \Bigg($$

```
grad_f.Q6 <- function(x,y){

  df_dx <-  - 2*(1 - x) - 400*x*(y - x^2)
  df_fy <-  200*(y-x^2)

  return <- matrix(data=c(df_dx,df_fy),nrow = 2,ncol = 1)

}
```

```
grad_2_f.Q6 <- function(x,y){

  d2f_d2x <- 2 - 400 * y + 1200 * x^2
  d2f_dx_dy <- - 400 * x
  d2f_d2y <- 200
```

```r
  return(matrix(data = c(d2f_d2x,d2f_dx_dy,d2f_dx_dy,d2f_d2y),nrow = 2,ncol = 2))

}
```

```r
NR.Q6 <- function(x,y,eps_for_break){

  x.j <- x
  y.j <- y

  counter <-  0
  while (TRUE) {

  counter <- counter + 1

  x_y.j <- matrix(c(x.j,y.j),ncol = 1)

  x_y.j_1 <- x_y.j - solve(grad_2_f.Q6(x.j,y.j)) %*%  grad_f.Q6(x.j,y.j)
  x.j <- x_y.j_1[1]
  y.j <- x_y.j_1[2]

  if (abs(grad_f.Q6(x.j,y.j)) < eps_for_break) {break}
  }

  return(cat(sprintf("x (NR) = %s", x.j),
    sprintf("y (NR) = %s", y.j),
    sprintf("counter (NR) = %s", counter),
    sep = "\n"))
}
```

```r
NR.Q6 (x = 1.01, y = 1.01,eps_for_break = 10^-5)
```

```
## x (NR) = 1.00000012923304
## y (NR) = 1.00000025825566
## counter (NR) = 3
```

```r
NR.Q6 (x = 2, y = 2,eps_for_break = 10^-5)
```

```
## x (NR) = 1
## y (NR) = 1.00000000000001
## counter (NR) = 5
```

```r
NR.Q6 (x = 0, y = 0,eps_for_break = 10^-5)
```

```
## x (NR) = 1
## y (NR) = 1
## counter (NR) = 2
```

```r
NR.Q6 (x = 10, y = 10,eps_for_break = 10^-5)
```

```
## x (NR) = 1
## y (NR) = 1
## counter (NR) = 5
```

```r
NR.Q6 (-105,20,eps_for_break = 10^-5)
```

```
## x (NR) = 0.999999999999971
## y (NR) = 0.999999997565126
## counter (NR) = 4
```

We can see that the NR algorithm convergent to the optimum solution faster (really faster) then the Gradient Descent algorithm.

The Gradient Descent algorithm use a lot of "cheap" iterations while the NR algorithm use few "expensive" iterations.

In addition this function is a "not normal" function and it is a disadvantage for Gradient Descent algorithm.