ST340 Assignment 3

Xintian Han 1909780, Runze Wang 1907544, Jingyuan Chen 2029628

14/03/2023

Q1 Gradient descent

There is a function that does gradient descent with a fixed number of iterations to find local minimum:

```
gradient.descent <- function(f, gradf, x0, iterations=1000, eta=0.2) {
  x<-x0
for (i in 1:iterations) {
  cat(i,"/",iterations,": ",x," ",f(x),"\n")
  x<-x-eta*gradf(x)
}
x
}
```

Example:

```
 f <-function(x) { sum(x^2) }
gradf<-function(x) { 2*x }
gradient.descent(f,gradf,c(10,20),10,0.2)
```

```
## 1 / 10 :  10 20   500
## 2 / 10 :  6 12   180
## 3 / 10 :  3.6 7.2   64.8
## 4 / 10 :  2.16 4.32   23.328
## 5 / 10 :  1.296 2.592   8.39808
## 6 / 10 :  0.7776 1.5552   3.023309
## 7 / 10 :  0.46656 0.93312   1.088391
## 8 / 10 :  0.279936 0.559872   0.3918208
## 9 / 10 :  0.1679616 0.3359232   0.1410555
## 10 / 10 :  0.100777 0.2015539   0.05077998
```

```
## [1] 0.06046618 0.12093235
```

(a) Write a short function that uses gradient.descent to find a local maximum.

Gradient descent is opposite to gradient ascent as one is finding minimum and another is finding maximum, we could adjust the function by taking eta to be -eta, so if -eta=0.2, then we will have x+0.2*gradf(), we have obtained an increasing function and the direction is increasing most rapidly, which is the same as gradient ascent.

```r
gradient.ascent <- function(f, df, x0, iterations=1000, eta=0.2) {
  # change eta to -eta to find the direction increasing the most
  gradient.descent(f,df,x0,iterations,-eta)
}

f <-function(x) { (1+x^2)^(-1) }
gradf<-function(x) { -2*x*(1+x^2)^(-2) }
gradient.ascent(f,gradf,3,40,0.5)
```

```
## 1 / 40 :  3   0.1
## 2 / 40 :  2.97   0.1018237
## 3 / 40 :  2.939207   0.1037459
## 4 / 40 :  2.907572   0.1057756
## 5 / 40 :  2.87504   0.1079231
## 6 / 40 :  2.841554   0.1101998
## 7 / 40 :  2.807046   0.1126189
## 8 / 40 :  2.771444   0.1151954
## 9 / 40 :  2.734667   0.1179467
## 10 / 40 :  2.696624   0.120893
## 11 / 40 :  2.657212   0.1240575
## 12 / 40 :  2.616317   0.1274679
## 13 / 40 :  2.573807   0.1311564
## 14 / 40 :  2.529532   0.1351619
## 15 / 40 :  2.483321   0.1395307
## 16 / 40 :  2.434974   0.144319
## 17 / 40 :  2.384258   0.1495956
## 18 / 40 :  2.330901   0.155446
## 19 / 40 :  2.274579   0.1619772
## 20 / 40 :  2.214901   0.1693254
## 21 / 40 :  2.151398   0.1776669
## 22 / 40 :  2.083488   0.1872336
## 23 / 40 :  2.010448   0.1983379
## 24 / 40 :  1.931361   0.2114095
## 25 / 40 :  1.845041   0.2270572
## 26 / 40 :  1.74992   0.2461708
## 27 / 40 :  1.643875   0.2701006
## 28 / 40 :  1.523947   0.300986
## 29 / 40 :  1.385889   0.3423852
## 30 / 40 :  1.223424   0.400518
## 31 / 40 :  1.027169   0.4866
## 32 / 40 :  0.7839563   0.6193532
## 33 / 40 :  0.4832319   0.8106927
## 34 / 40 :  0.165641   0.9732957
## 35 / 40 :  0.008728518   0.9999238
## 36 / 40 :  1.329848e-06   1
## 37 / 40 :  4.703997e-18   1
## 38 / 40 :  0   1
## 39 / 40 :  0   1
## 40 / 40 :  0   1
```

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

The above code gives the final output of x=0 and f(x)=1. Which indicates that the function $\frac{1}{1+x^2}$ achieves its maximum that f(x)=1 when x=0.

(b) consider:

```
f <- function(x) (x[1]-1)^2 + 100*(x[1]^2-x[2])^2
```

i)proof f has a unique minimum

$$f : \mathbb{R}^2 \to \mathbb{R}$$

the function f has the form

$$f(x_1, x_2) = (x_1 - 1)^2 + 100(x_1^2 - x_2)^2$$

The function attains its minimum value when

$$f(x_1, x_2) = 0$$

we know the property of square, that is

$$(x_1 - 1)^2 \geq 0$$

and

$$(x_1^2 - x_2)^2 \geq 0$$

Theses two squares are minimised when both of them has value=0 Then we can get

$$(x_1 - 1) = 0$$

and

$$(x_1^2 - x_2) = 0$$

Solving these equations we get $x_1 = 1$ and because $x_1^2 = 1$ we have $x_2 = 1$. Because we only obtain one unique value for x_1 and x_2, so the function $f(x_1, x_2)$ has a unique minimum value $f(x_1, x_2) = 0$ when $x_1 = 1$ and $x_2 = 1$

ii) we use the partial derivative to get

$$\nabla f = \nabla f(x_1, x_2) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}) = (400x_1(x_1^2 - x_2) + 2(x_1 - 1), 200(x_2 - x_1^2))$$

We create the gradf function to output the partial derivative of $f(x_1, x_2)$ in the vector form with the input of $x = (x_1, x_2)$

```
gradf<- function(x) {
  part1<-400*x[1]^3-400*x[1]*x[2]+2*x[1]-2
  part2<-200*x[2]-200*x[1]^2
  c(part1, part2)
  }
```

We can use examples to check it actually works

```
gradf(c(1,1))
```

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

```
gradf(c(2,2))
```

```
## [1] 1602 -400
```

When $(x_1, x_2) = (0, 0)$, we get the unique minimum value of the function $f = 0$, this means the partial derivative in the vector form should also be 0 in each part. Similar situation apply for $(x_1, x_2) = (2, 2)$, put these two value into the part1 and part2 formula we would get the same answer.

iii) Firstly, we need to find the range of eta that could give us a value for the function f which is close to 0 in order to find the minimum value and get value of x_1 and x_2

```
gradient_descent2 <- function(f, gradf, x0, iterations) {
   # create a bunch of values of eta
  eta <- c(0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.5, 1)
  result <- list()
  for (k in 1:length(eta)) {
    x <- x0 #initial value of x
    for (i in 1:iterations) {
      x <- x - eta[k] * gradf(x) #updated value of x with iterations
    }
    # create a data frame to store the results
    value <- data.frame(
      eta = eta[k],
      iterations = iterations,
      x = x,
      fx=f(x)
    )

    result[[k]] <- value
  }

  return(result)
}

f <- function(x) {(x[1]-1)^2 + 100*(x[1]^2-x[2])^2}
gradf<-function(x) {c(400*x[1]^3-400*x[1]*x[2]+2*x[1]-2,200*x[2]-200*x[1]^2)}
gradient_descent2(f,gradf,c(3,4),iterations = 200)
```

```
## [[1]]
##     eta iterations        x       fx
## 1 1e-06        200 2.386365 251.7248
## 2 1e-06        200 4.114224 251.7248
##
## [[2]]
##     eta iterations        x       fx
## 1 1e-05        200 2.046603 1.095427
## 2 1e-05        200 4.189284 1.095427
##
## [[3]]
##     eta iterations        x       fx
## 1 1e-04        200 2.041234 1.084748
## 2 1e-04        200 4.169043 1.084748
```

```
##
## [[4]]
##     eta iterations   x  fx
## 1 0.001        200 NaN NaN
## 2 0.001        200 NaN NaN
##
## [[5]]
##    eta iterations   x  fx
## 1 0.01        200 NaN NaN
## 2 0.01        200 NaN NaN
##
## [[6]]
##   eta iterations   x  fx
## 1 0.5        200 NaN NaN
## 2 0.5        200 NaN NaN
##
## [[7]]
##   eta iterations   x  fx
## 1   1        200 NaN NaN
## 2   1        200 NaN NaN
```

The first row is the data for x_1 and the second row is the data for x_2. From the above we know that values of eta=C(0.0001,0.00001,0.000001), we could get the value for x and f_x, other values of eta bigger than 0.001 is not working because we have NaN.Then we could find values of x_1 and x_2 by finding the minimum value of function equals to 0 for each different values of eta. For x_1 and x_2 both equals to 0.0001, we have fx=1.08, for x_1 and x_2 both equals to 0.001, we have fx=NaN. So the minimum value must exist within this range.Hence, we need to choose the value of eta between 0.001-0.0001.

Then we can apply the function to find the exact value of eta :

```r
gradient_descent3 <- function(f, gradf, x0, iterations) {
   # create a bunch of values of eta using sequence function
  eta_1<- seq(from=0.0001, to=0.001, by=0.00001)
  result <- list()
  for (eta in eta_1) {
    x <- x0
    for (i in 1:iterations) {
      x <- x - eta * gradf(x)
    }
    # create a data frame to store the results
    value <- data.frame(
      eta = eta,
      iterations = iterations,
      x = x,
      fx=f(x)
    )
    #add the data frame
    result[[length(result) + 1]] <- value
  }

  return(result)
}

f <- function(x) {(x[1]-1)^2 + 100*(x[1]^2-x[2])^2}
```

```r
gradf <- function(x) {c(400*x[1]^3-400*x[1]*x[2]+2*x[1]-2, 200*x[2]-200*x[1]^2)}
gradient_descent3(f, gradf, c(3, 4), iterations = 100000)
```

```
## [[1]]
##     eta iterations        x         fx
## 1 1e-04      1e+05 1.127502 0.01627908
## 2 1e-04      1e+05 1.271733 0.01627908
##
## [[2]]
##       eta iterations        x          fx
## 1 0.00011      1e+05 1.090809 0.008258111
## 2 0.00011      1e+05 1.190208 0.008258111
##
## [[3]]
##       eta iterations        x          fx
## 1 0.00012      1e+05 1.063592 0.004049887
## 2 0.00012      1e+05 1.131472 0.004049887
##
## [[4]]
##       eta iterations        x          fx
## 1 0.00013      1e+05 1.043947 0.001934275
## 2 0.00013      1e+05 1.089997 0.001934275
##
## [[5]]
##       eta iterations        x           fx
## 1 0.00014      1e+05 1.030070 0.0009055757
## 2 0.00014      1e+05 1.061161 0.0009055757
##
## [[6]]
##       eta iterations        x           fx
## 1 0.00015      1e+05 1.020424 0.0004177763
## 2 0.00015      1e+05 1.041345 0.0004177763
##
## [[7]]
##       eta iterations        x           fx
## 1 0.00016      1e+05 1.013798 0.0001906834
## 2 0.00016      1e+05 1.027841 0.0001906834
##
## [[8]]
##       eta iterations        x           fx
## 1 0.00017      1e+05 1.009285 8.635682e-05
## 2 0.00017      1e+05 1.018694 8.635682e-05
##
## [[9]]
##       eta iterations        x           fx
## 1 0.00018      1e+05 1.006231 3.888513e-05
## 2 0.00018      1e+05 1.012525 3.888513e-05
##
## [[10]]
##       eta iterations        x           fx
## 1 0.00019      1e+05 1.004172 1.743333e-05
## 2 0.00019      1e+05 1.008378 1.743333e-05
##
```

```
## [[11]]
##     eta iterations        x           fx
## 1 2e-04      1e+05 1.002789 7.789133e-06
## 2 2e-04      1e+05 1.005596 7.789133e-06
##
## [[12]]
##       eta iterations        x           fx
## 1 0.00021      1e+05 1.001861 3.470307e-06
## 2 0.00021      1e+05 1.003734 3.470307e-06
##
## [[13]]
##       eta iterations        x           fx
## 1 0.00022      1e+05 1.001241 1.542316e-06
## 2 0.00022      1e+05 1.002488 1.542316e-06
##
## [[14]]
##       eta iterations        x          fx
## 1 0.00023      1e+05 1.000826 6.83898e-07
## 2 0.00023      1e+05 1.001657 6.83898e-07
##
## [[15]]
##       eta iterations        x           fx
## 1 0.00024      1e+05 1.000550 3.025926e-07
## 2 0.00024      1e+05 1.001102 3.025926e-07
##
## [[16]]
##       eta iterations        x           fx
## 1 0.00025      1e+05 1.000365 1.335902e-07
## 2 0.00025      1e+05 1.000732 1.335902e-07
##
## [[17]]
##       eta iterations        x           fx
## 1 0.00026      1e+05 1.000242 5.884562e-08
## 2 0.00026      1e+05 1.000486 5.884562e-08
##
## [[18]]
##       eta iterations        x           fx
## 1 0.00027      1e+05 1.000161 2.586006e-08
## 2 0.00027      1e+05 1.000322 2.586006e-08
##
## [[19]]
##       eta iterations        x           fx
## 1 0.00028      1e+05 1.000106 1.133588e-08
## 2 0.00028      1e+05 1.000213 1.133588e-08
##
## [[20]]
##       eta iterations        x           fx
## 1 0.00029      1e+05 1.000070 4.955753e-09
## 2 0.00029      1e+05 1.000141 4.955753e-09
##
## [[21]]
##     eta iterations        x           fx
## 1 3e-04      1e+05 1.000046 2.160198e-09
## 2 3e-04      1e+05 1.000093 2.160198e-09
```

```
##
## [[22]]
##       eta iterations        x           fx
## 1 0.00031      1e+05 1.000031 9.386201e-10
## 2 0.00031      1e+05 1.000061 9.386201e-10
##
## [[23]]
##       eta iterations       x           fx
## 1 0.00032      1e+05 1.00002 4.064061e-10
## 2 0.00032      1e+05 1.00004 4.064061e-10
##
## [[24]]
##       eta iterations        x           fx
## 1 0.00033      1e+05 1.000013 1.752835e-10
## 2 0.00033      1e+05 1.000027 1.752835e-10
##
## [[25]]
##       eta iterations        x           fx
## 1 0.00034      1e+05 1.000009 7.527272e-11
## 2 0.00034      1e+05 1.000017 7.527272e-11
##
## [[26]]
##       eta iterations        x           fx
## 1 0.00035      1e+05 1.000006 3.216746e-11
## 2 0.00035      1e+05 1.000011 3.216746e-11
##
## [[27]]
##       eta iterations        x           fx
## 1 0.00036      1e+05 1.000004 1.367088e-11
## 2 0.00036      1e+05 1.000007 1.367088e-11
##
## [[28]]
##       eta iterations        x          fx
## 1 0.00037      1e+05 1.000002 5.77342e-12
## 2 0.00037      1e+05 1.000005 5.77342e-12
##
## [[29]]
##       eta iterations        x           fx
## 1 0.00038      1e+05 1.000002 2.420483e-12
## 2 0.00038      1e+05 1.000003 2.420483e-12
##
## [[30]]
##       eta iterations        x           fx
## 1 0.00039      1e+05 1.000001 1.006159e-12
## 2 0.00039      1e+05 1.000002 1.006159e-12
##
## [[31]]
##     eta iterations        x          fx
## 1 4e-04      1e+05 1.000001 4.14028e-13
## 2 4e-04      1e+05 1.000001 4.14028e-13
##
## [[32]]
##       eta iterations        x           fx
## 1 0.00041      1e+05 1.000000 1.682884e-13
```

```
## 2 0.00041      1e+05 1.000001 1.682884e-13
##
## [[33]]
##       eta iterations        x           fx
## 1 0.00042      1e+05 1.000000 6.736729e-14
## 2 0.00042      1e+05 1.000001 6.736729e-14
##
## [[34]]
##       eta iterations x           fx
## 1 0.00043      1e+05 1 2.644896e-14
## 2 0.00043      1e+05 1 2.644896e-14
##
## [[35]]
##       eta iterations x          fx
## 1 0.00044      1e+05 1 1.01232e-14
## 2 0.00044      1e+05 1 1.01232e-14
##
## [[36]]
##       eta iterations x           fx
## 1 0.00045      1e+05 1 3.741358e-15
## 2 0.00045      1e+05 1 3.741358e-15
##
## [[37]]
##       eta iterations x           fx
## 1 0.00046      1e+05 1 1.312984e-15
## 2 0.00046      1e+05 1 1.312984e-15
##
## [[38]]
##       eta iterations x          fx
## 1 0.00047      1e+05 1 4.23848e-16
## 2 0.00047      1e+05 1 4.23848e-16
##
## [[39]]
##       eta iterations x           fx
## 1 0.00048      1e+05 1 1.167712e-16
## 2 0.00048      1e+05 1 1.167712e-16
##
## [[40]]
##       eta iterations x           fx
## 1 0.00049      1e+05 1 2.113072e-17
## 2 0.00049      1e+05 1 2.113072e-17
##
## [[41]]
##     eta iterations x           fx
## 1 5e-04      1e+05 1 5.394804e-19
## 2 5e-04      1e+05 1 5.394804e-19
##
## [[42]]
##       eta iterations x           fx
## 1 0.00051      1e+05 1 6.821035e-19
## 2 0.00051      1e+05 1 6.821035e-19
##
## [[43]]
##       eta iterations x           fx
```

```
## 1 0.00052      1e+05 1 3.782668e-19
## 2 0.00052      1e+05 1 3.782668e-19
##
## [[44]]
##       eta iterations x           fx
## 1 0.00053      1e+05 1 1.936525e-19
## 2 0.00053      1e+05 1 1.936525e-19
##
## [[45]]
##       eta iterations x          fx
## 1 0.00054      1e+05 1 9.58016e-20
## 2 0.00054      1e+05 1 9.58016e-20
##
## [[46]]
##       eta iterations x           fx
## 1 0.00055      1e+05 1 4.651031e-20
## 2 0.00055      1e+05 1 4.651031e-20
##
## [[47]]
##       eta iterations x           fx
## 1 0.00056      1e+05 1 2.230479e-20
## 2 0.00056      1e+05 1 2.230479e-20
##
## [[48]]
##       eta iterations x           fx
## 1 0.00057      1e+05 1 1.057575e-20
## 2 0.00057      1e+05 1 1.057575e-20
##
## [[49]]
##       eta iterations x           fx
## 1 0.00058      1e+05 1 4.989991e-21
## 2 0.00058      1e+05 1 4.989991e-21
##
## [[50]]
##       eta iterations x           fx
## 1 0.00059      1e+05 1 2.353403e-21
## 2 0.00059      1e+05 1 2.353403e-21
##
## [[51]]
##     eta iterations x           fx
## 1 6e-04      1e+05 1 1.102864e-21
## 2 6e-04      1e+05 1 1.102864e-21
##
## [[52]]
##       eta iterations x           fx
## 1 0.00061      1e+05 1 4.982723e-22
## 2 0.00061      1e+05 1 4.982723e-22
##
## [[53]]
##       eta iterations x           fx
## 1 0.00062      1e+05 1 2.178799e-22
## 2 0.00062      1e+05 1 2.178799e-22
##
## [[54]]
```

```
##       eta iterations x           fx
## 1 0.00063      1e+05 1 9.494006e-23
## 2 0.00063      1e+05 1 9.494006e-23
##
## [[55]]
##       eta iterations x           fx
## 1 0.00064      1e+05 1 4.137258e-23
## 2 0.00064      1e+05 1 4.137258e-23
##
## [[56]]
##       eta iterations x           fx
## 1 0.00065      1e+05 1 1.806249e-23
## 2 0.00065      1e+05 1 1.806249e-23
##
## [[57]]
##       eta iterations x           fx
## 1 0.00066      1e+05 1 7.862054e-24
## 2 0.00066      1e+05 1 7.862054e-24
##
## [[58]]
##       eta iterations x           fx
## 1 0.00067      1e+05 1 3.450532e-24
## 2 0.00067      1e+05 1 3.450532e-24
##
## [[59]]
##       eta iterations x           fx
## 1 0.00068      1e+05 1 1.510251e-24
## 2 0.00068      1e+05 1 1.510251e-24
##
## [[60]]
##       eta iterations x           fx
## 1 0.00069      1e+05 1 6.619244e-25
## 2 0.00069      1e+05 1 6.619244e-25
##
## [[61]]
##     eta iterations x           fx
## 1 7e-04      1e+05 1 2.876388e-25
## 2 7e-04      1e+05 1 2.876388e-25
##
## [[62]]
##       eta iterations x           fx
## 1 0.00071      1e+05 1 1.263952e-25
## 2 0.00071      1e+05 1 1.263952e-25
##
## [[63]]
##       eta iterations x           fx
## 1 0.00072      1e+05 1 5.832553e-26
## 2 0.00072      1e+05 1 5.832553e-26
##
## [[64]]
##       eta iterations x           fx
## 1 0.00073      1e+05 1 2.753555e-26
## 2 0.00073      1e+05 1 2.753555e-26
##
```

```
## [[65]]
##       eta iterations x           fx
## 1 0.00074      1e+05 1 1.487131e-26
## 2 0.00074      1e+05 1 1.487131e-26
##
## [[66]]
##       eta iterations x           fx
## 1 0.00075      1e+05 1 1.487131e-26
## 2 0.00075      1e+05 1 1.487131e-26
##
## [[67]]
##       eta iterations x          fx
## 1 0.00076      1e+05 1 1.46287e-26
## 2 0.00076      1e+05 1 1.46287e-26
##
## [[68]]
##       eta iterations x          fx
## 1 0.00077      1e+05 1 1.46287e-26
## 2 0.00077      1e+05 1 1.46287e-26
##
## [[69]]
##       eta iterations x           fx
## 1 0.00078      1e+05 1 1.438808e-26
## 2 0.00078      1e+05 1 1.438808e-26
##
## [[70]]
##       eta iterations x           fx
## 1 0.00079      1e+05 1 1.438808e-26
## 2 0.00079      1e+05 1 1.438808e-26
##
## [[71]]
##     eta iterations x           fx
## 1 8e-04      1e+05 1 1.438808e-26
## 2 8e-04      1e+05 1 1.438808e-26
##
## [[72]]
##       eta iterations x           fx
## 1 0.00081      1e+05 1 1.417588e-26
## 2 0.00081      1e+05 1 1.417588e-26
##
## [[73]]
##       eta iterations x           fx
## 1 0.00082      1e+05 1 1.417588e-26
## 2 0.00082      1e+05 1 1.417588e-26
##
## [[74]]
##       eta iterations x           fx
## 1 0.00083      1e+05 1 1.401776e-26
## 2 0.00083      1e+05 1 1.401776e-26
##
## [[75]]
##       eta iterations x           fx
## 1 0.00084      1e+05 1 1.378225e-26
## 2 0.00084      1e+05 1 1.378225e-26
```

```
##
## [[76]]
##       eta iterations x           fx
## 1 0.00085      1e+05 1 4.296279e-26
## 2 0.00085      1e+05 1 4.296279e-26
##
## [[77]]
##       eta iterations x           fx
## 1 0.00086      1e+05 1 1.378225e-26
## 2 0.00086      1e+05 1 1.378225e-26
##
## [[78]]
##       eta iterations x           fx
## 1 0.00087      1e+05 1 4.213878e-26
## 2 0.00087      1e+05 1 4.213878e-26
##
## [[79]]
##       eta iterations x           fx
## 1 0.00088      1e+05 1 1.354874e-26
## 2 0.00088      1e+05 1 1.354874e-26
##
## [[80]]
##       eta iterations x           fx
## 1 0.00089      1e+05 1 1.354874e-26
## 2 0.00089      1e+05 1 1.354874e-26
##
## [[81]]
##     eta iterations   x  fx
## 1 9e-04      1e+05 NaN NaN
## 2 9e-04      1e+05 NaN NaN
##
## [[82]]
##       eta iterations   x  fx
## 1 0.00091      1e+05 NaN NaN
## 2 0.00091      1e+05 NaN NaN
##
## [[83]]
##       eta iterations x           fx
## 1 0.00092      1e+05 1 1.318946e-26
## 2 0.00092      1e+05 1 1.318946e-26
##
## [[84]]
##       eta iterations x           fx
## 1 0.00093      1e+05 1 4.069359e-26
## 2 0.00093      1e+05 1 4.069359e-26
##
## [[85]]
##       eta iterations x           fx
## 1 0.00094      1e+05 1 4.006925e-26
## 2 0.00094      1e+05 1 4.006925e-26
##
## [[86]]
##       eta iterations x           fx
## 1 0.00095      1e+05 1 1.296105e-26
```

```
## 2 0.00095      1e+05 1 1.296105e-26
##
## [[87]]
##       eta iterations x           fx
## 1 0.00096      1e+05 1 3.927364e-26
## 2 0.00096      1e+05 1 3.927364e-26
##
## [[88]]
##       eta iterations x           fx
## 1 0.00097      1e+05 1 1.296105e-26
## 2 0.00097      1e+05 1 1.296105e-26
##
## [[89]]
##       eta iterations x         fx
## 1 0.00098      1e+05 1 7.1244e-27
## 2 0.00098      1e+05 1 7.1244e-27
##
## [[90]]
##       eta iterations   x  fx
## 1 0.00099      1e+05 NaN NaN
## 2 0.00099      1e+05 NaN NaN
##
## [[91]]
##     eta iterations   x  fx
## 1 0.001      1e+05 NaN NaN
## 2 0.001      1e+05 NaN NaN
```

From the above we can see that when the minimum of function approaches 0, we have both x_1 and x_2 with value 1 if we have 100000 iterations and we also obtain the maximum value of eta=0.00089.

(c) we know that the gradient decent momentum has an extra component in the function that is alpha, then we could construct

```r
gradient_descent.momentum <- function(f, gradf, x0, iterations, eta, alpha) {
  result <- list()
  for(p in 1:length(alpha)) {
    x1 <- x0
     # update x0 without momentum
    x0 <- x1 - eta * gradf(x1)
    # we have iterations-1 remaining
    for(i in 1:(iterations-1)) {
      #with momentum
      x2 <- x1 - eta * gradf(x1) + alpha[p] * (x1 - x0)
      x0 <- x1
      x1 <- x2
    }
    value <- data.frame(
      eta = eta,
      iterations = iterations,
      x = x1,
      alpha = alpha[p],
      fx = f(x1)
    )
```

```
    result[[p]] <- value
  }
  return(result) # return the whole list of values
}

f <- function(x) {(x[1]-1)^2 + 100*(x[1]^2-x[2])^2}
gradf <- function(x) {c(400*x[1]^3-400*x[1]*x[2]+2*x[1]-2, 200*x[2]-200*x[1]^2)}
gradient_descent.momentum(f, gradf, c(3, 4), iterations = 500,eta = 0.00089,
  alpha =c(0.000001,0.00001, 0.0001, 0.001, 0.01 , 0.1 ,0.5 ,1 ) )
```

```
## [[1]]
##       eta iterations         x alpha       fx
## 1 0.00089        500 -1.368061 1e-06 5.613544
## 2 0.00089        500  1.879227 1e-06 5.613544
##
## [[2]]
##       eta iterations         x alpha       fx
## 1 0.00089        500 -1.087298 1e-05 4.363088
## 2 0.00089        500  1.190139 1e-05 4.363088
##
## [[3]]
##       eta iterations          x alpha       fx
## 1 0.00089        500 -0.6854635 1e-04 2.847227
## 2 0.00089        500  0.4778852 1e-04 2.847227
##
## [[4]]
##       eta iterations           x alpha        fx
## 1 0.00089        500 0.073401666 0.001 0.8587278
## 2 0.00089        500 0.004190803 0.001 0.8587278
##
## [[5]]
##       eta iterations         x alpha       fx
## 1 0.00089        500 0.5017621  0.01 0.248865
## 2 0.00089        500 0.2492672  0.01 0.248865
##
## [[6]]
##       eta iterations         x alpha        fx
## 1 0.00089        500 0.6688224   0.1 0.1099316
## 2 0.00089        500 0.4457327   0.1 0.1099316
##
## [[7]]
##       eta iterations         x alpha         fx
## 1 0.00089        500 0.8085557   0.5 0.03672441
## 2 0.00089        500 0.6529051   0.5 0.03672441
##
## [[8]]
##       eta iterations        x alpha         fx
## 1 0.00089        500 1.191594     1 0.03675509
## 2 0.00089        500 1.420581     1 0.03675509
```

From the above data frame we get that when eta achieved the max value of 0.00089, we need to find the value of alpha that could lead to the minimum value of the function. That is when alpha has value between

0.001 and 1, we have fx less than 1. Then we look closer in the region to find the most accurate value of alpha by taking alpha from the sequence :

```
gradient_descent.momentum2<- function(f, gradf, x0, iterations, eta, alpha) {
  alpha_1<- seq(from=0.001, to=1, by=0.01)
  result <- list()
  for( alpha in alpha_1) {
    x1 <- x0
    # update x0 without momentum
    x0 <- x1 - eta * gradf(x1)
    # have iterations-1 remaining
    for(i in 1:(iterations-1)) {
      x2 <- x1 - eta * gradf(x1) + alpha* (x1 - x0)
      x0 <- x1
      x1 <- x2
      }
    value <- data.frame(
      eta = eta,
      iterations = iterations,
      x = x1,
      alpha = alpha,
      fx = f(x1)
    )
    #add the new term of data frame
    result[[length(result) + 1]] <- value
  }
  return(result) # return the whole list of values
}

f <- function(x) {(x[1]-1)^2 + 100*(x[1]^2-x[2])^2}
gradf <- function(x) {c(400*x[1]^3-400*x[1]*x[2]+2*x[1]-2, 200*x[2]-200*x[1]^2)}
gradient_descent.momentum2(f, gradf, c(3, 4), iterations = 100,eta = 0.00089,alpha )
```

```
## [[1]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.553493 0.001 6.531439
## 2 0.00089        100  2.423883 0.001 6.531439
##
## [[2]]
##       eta iterations        x alpha      fx
## 1 0.00089        100 -1.51143 0.011 6.31289
## 2 0.00089        100  2.29191 0.011 6.31289
##
## [[3]]
##       eta iterations         x alpha      fx
## 1 0.00089        100 -1.466493 0.021 6.08926
## 2 0.00089        100  2.158134 0.021 6.08926
##
## [[4]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.419361 0.031 5.859056
## 2 0.00089        100  2.022168 0.031 5.859056
##
## [[5]]
```

```
##       eta iterations         x alpha      fx
## 1 0.00089        100 -1.369773 0.041 5.62165
## 2 0.00089        100  1.883911 0.041 5.62165
##
## [[6]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.317409 0.051 5.376296
## 2 0.00089        100  1.743255 0.051 5.376296
##
## [[7]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.261879 0.061 5.122098
## 2 0.00089        100  1.600085 0.061 5.122098
##
## [[8]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.202696 0.071 4.857963
## 2 0.00089        100  1.454284 0.071 4.857963
##
## [[9]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.139238 0.081 4.582534
## 2 0.00089        100  1.305734 0.081 4.582534
##
## [[10]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 -1.070698 0.091 4.294089
## 2 0.00089        100  1.154331 0.091 4.294089
##
## [[11]]
##       eta iterations          x alpha       fx
## 1 0.00089        100 -0.9959961 0.101 3.990403
## 2 0.00089        100  1.0000100 0.101 3.990403
##
## [[12]]
##       eta iterations          x alpha       fx
## 1 0.00089        100 -0.9136414 0.111 3.668515
## 2 0.00089        100  0.8427979 0.111 3.668515
##
## [[13]]
##       eta iterations          x alpha       fx
## 1 0.00089        100 -0.8214940 0.121 3.324382
## 2 0.00089        100  0.6829402 0.121 3.324382
##
## [[14]]
##       eta iterations          x alpha       fx
## 1 0.00089        100 -0.7163468 0.131 2.952333
## 2 0.00089        100  0.5212068 0.131 2.952333
##
## [[15]]
##       eta iterations          x alpha       fx
## 1 0.00089        100 -0.5931776 0.141 2.544382
## 2 0.00089        100  0.3597129 0.141 2.544382
##
```

```
## [[16]]
##       eta iterations          x alpha       fx
## 1 0.00089        100 -0.4440176 0.151 2.090356
## 2 0.00089        100  0.2043410 0.151 2.090356
##
## [[17]]
##       eta iterations           x alpha       fx
## 1 0.00089        100 -0.25848333 0.161 1.586514
## 2 0.00089        100  0.07204213 0.161 1.586514
##
## [[18]]
##       eta iterations            x alpha       fx
## 1 0.00089        100 -0.041258047 0.171 1.084327
## 2 0.00089        100  0.002742535 0.171 1.084327
##
## [[19]]
##       eta iterations          x alpha        fx
## 1 0.00089        100 0.15369100 0.181 0.7167767
## 2 0.00089        100 0.02130199 0.181 0.7167767
##
## [[20]]
##       eta iterations          x alpha        fx
## 1 0.00089        100 0.29193346 0.191 0.5023106
## 2 0.00089        100 0.08213907 0.191 0.5023106
##
## [[21]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.3893259 0.201 0.3738051
## 2 0.00089        100 0.1486043 0.201 0.3738051
##
## [[22]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.4622524 0.211 0.2898957
## 2 0.00089        100 0.2109881 0.211 0.2898957
##
## [[23]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.5197643 0.221 0.2312052
## 2 0.00089        100 0.2677489 0.221 0.2312052
##
## [[24]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.5668540 0.231 0.1880794
## 2 0.00089        100 0.3191695 0.231 0.1880794
##
## [[25]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.6064742 0.241 0.1552372
## 2 0.00089        100 0.3658755 0.241 0.1552372
##
## [[26]]
##       eta iterations         x alpha      fx
## 1 0.00089        100 0.6404937 0.251 0.12955
## 2 0.00089        100 0.4084852 0.251 0.12955
```

```
##
## [[27]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.6701626 0.261 0.1090434
## 2 0.00089        100 0.4475346 0.261 0.1090434
##
## [[28]]
##       eta iterations         x alpha       fx
## 1 0.00089        100 0.6963564 0.271 0.092407
## 2 0.00089        100 0.4834716 0.271 0.092407
##
## [[29]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.7197104 0.281 0.0787352
## 2 0.00089        100 0.5166680 0.281 0.0787352
##
## [[30]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.7407005 0.291 0.06738118
## 2 0.00089        100 0.5474333 0.291 0.06738118
##
## [[31]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.7596918 0.301 0.0578701
## 2 0.00089        100 0.5760268 0.301 0.0578701
##
## [[32]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.7769707 0.311 0.0498453
## 2 0.00089        100 0.6026675 0.311 0.0498453
##
## [[33]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.7927659 0.321 0.04303356
## 2 0.00089        100 0.6275419 0.321 0.04303356
##
## [[34]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8072631 0.331 0.03722209
## 2 0.00089        100 0.6508101 0.331 0.03722209
##
## [[35]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8206148 0.341 0.03224268
## 2 0.00089        100 0.6726109 0.341 0.03224268
##
## [[36]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.8329483 0.351 0.0279607
## 2 0.00089        100 0.6930651 0.351 0.0279607
##
## [[37]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8443704 0.361 0.02426719
```

```
## 2 0.00089        100 0.7122786 0.361 0.02426719
##
## [[38]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8549722 0.371 0.02107305
## 2 0.00089        100 0.7303451 0.371 0.02107305
##
## [[39]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8648316 0.381 0.01830484
## 2 0.00089        100 0.7473476 0.381 0.01830484
##
## [[40]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.8740160 0.391 0.0159015
## 2 0.00089        100 0.7633606 0.391 0.0159015
##
## [[41]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8825839 0.401 0.01381193
## 2 0.00089        100 0.7784505 0.401 0.01381193
##
## [[42]]
##       eta iterations         x alpha        fx
## 1 0.00089        100 0.8905867 0.411 0.0119931
## 2 0.00089        100 0.7926774 0.411 0.0119931
##
## [[43]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.8980697 0.421 0.01040858
## 2 0.00089        100 0.8060957 0.421 0.01040858
##
## [[44]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9050729 0.431 0.009027318
## 2 0.00089        100 0.8187549 0.431 0.009027318
##
## [[45]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9116320 0.441 0.007822801
## 2 0.00089        100 0.8307001 0.441 0.007822801
##
## [[46]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9177790 0.451 0.006772242
## 2 0.00089        100 0.8419726 0.451 0.006772242
##
## [[47]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9235425 0.461 0.005856004
## 2 0.00089        100 0.8526105 0.461 0.005856004
##
## [[48]]
##       eta iterations         x alpha          fx
```

```
## 1 0.00089        100 0.9289486 0.471 0.005057107
## 2 0.00089        100 0.8626488 0.471 0.005057107
##
## [[49]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9340206 0.481 0.004360824
## 2 0.00089        100 0.8721198 0.481 0.004360824
##
## [[50]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9387800 0.491 0.003754345
## 2 0.00089        100 0.8810538 0.491 0.003754345
##
## [[51]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9432463 0.501 0.003226505
## 2 0.00089        100 0.8894786 0.501 0.003226505
##
## [[52]]
##       eta iterations         x alpha         fx
## 1 0.00089        100 0.9474373 0.511 0.00276755
## 2 0.00089        100 0.8974204 0.511 0.00276755
##
## [[53]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9513695 0.521 0.002368945
## 2 0.00089        100 0.9049035 0.521 0.002368945
##
## [[54]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9550579 0.531 0.002023206
## 2 0.00089        100 0.9119508 0.531 0.002023206
##
## [[55]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9585166 0.541 0.001723767
## 2 0.00089        100 0.9185839 0.541 0.001723767
##
## [[56]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9617586 0.551 0.001464857
## 2 0.00089        100 0.9248230 0.551 0.001464857
##
## [[57]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9647959 0.561 0.001241401
## 2 0.00089        100 0.9306872 0.561 0.001241401
##
## [[58]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9676397 0.571 0.001048934
## 2 0.00089        100 0.9361945 0.571 0.001048934
##
## [[59]]
```

```
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9703005 0.581 0.0008835228
## 2 0.00089        100 0.9413621 0.581 0.0008835228
##
## [[60]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9727882 0.591 0.0007417058
## 2 0.00089        100 0.9462062 0.591 0.0007417058
##
## [[61]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9751120 0.601 0.0006204333
## 2 0.00089        100 0.9507424 0.601 0.0006204333
##
## [[62]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9772806 0.611 0.0005170205
## 2 0.00089        100 0.9549852 0.611 0.0005170205
##
## [[63]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9793021 0.621 0.0004291046
## 2 0.00089        100 0.9589488 0.621 0.0004291046
##
## [[64]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9811844 0.631 0.0003546072
## 2 0.00089        100 0.9626467 0.631 0.0003546072
##
## [[65]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9829347 0.641 0.0002917018
## 2 0.00089        100 0.9660916 0.641 0.0002917018
##
## [[66]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9845599 0.651 0.0002387852
## 2 0.00089        100 0.9692959 0.651 0.0002387852
##
## [[67]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9860668 0.661 0.0001944515
## 2 0.00089        100 0.9722714 0.661 0.0001944515
##
## [[68]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9874615 0.671 0.0001574703
## 2 0.00089        100 0.9750296 0.671 0.0001574703
##
## [[69]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9887501 0.681 0.0001267665
## 2 0.00089        100 0.9775814 0.681 0.0001267665
##
```

```
## [[70]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9899383 0.691 0.0001014026
## 2 0.00089        100 0.9799373 0.691 0.0001014026
##
## [[71]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9910316 0.701 8.056305e-05
## 2 0.00089        100 0.9821075 0.701 8.056305e-05
##
## [[72]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9920352 0.711 6.354044e-05
## 2 0.00089        100 0.9841018 0.711 6.354044e-05
##
## [[73]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9929543 0.721 4.972276e-05
## 2 0.00089        100 0.9859298 0.721 4.972276e-05
##
## [[74]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9937935 0.731 3.858242e-05
## 2 0.00089        100 0.9876007 0.731 3.858242e-05
##
## [[75]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9945577 0.741 2.966627e-05
## 2 0.00089        100 0.9891232 0.741 2.966627e-05
##
## [[76]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9952513 0.751 2.258669e-05
## 2 0.00089        100 0.9905060 0.751 2.258669e-05
##
## [[77]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9958786 0.761 1.701363e-05
## 2 0.00089        100 0.9917576 0.761 1.701363e-05
##
## [[78]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9964437 0.771 1.266745e-05
## 2 0.00089        100 0.9928858 0.771 1.266745e-05
##
## [[79]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9969508 0.781 9.312574e-06
## 2 0.00089        100 0.9938987 0.781 9.312574e-06
##
## [[80]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9974037 0.791 6.751729e-06
## 2 0.00089        100 0.9948037 0.791 6.751729e-06
```

```
##
## [[81]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9978061 0.801 4.820879e-06
## 2 0.00089        100 0.9956082 0.801 4.820879e-06
##
## [[82]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9981617 0.811 3.384648e-06
## 2 0.00089        100 0.9963195 0.811 3.384648e-06
##
## [[83]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9984741 0.821 2.332251e-06
## 2 0.00089        100 0.9969443 0.821 2.332251e-06
##
## [[84]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9987465 0.831 1.573886e-06
## 2 0.00089        100 0.9974895 0.831 1.573886e-06
##
## [[85]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9989822 0.841 1.037525e-06
## 2 0.00089        100 0.9979614 0.841 1.037525e-06
##
## [[86]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9991845 0.851 6.660808e-07
## 2 0.00089        100 0.9983664 0.851 6.660808e-07
##
## [[87]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9993564 0.861 4.14914e-07
## 2 0.00089        100 0.9987106 0.861 4.14914e-07
##
## [[88]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9995007 0.871 2.496507e-07
## 2 0.00089        100 0.9989997 0.871 2.496507e-07
##
## [[89]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9996205 0.881 1.442828e-07
## 2 0.00089        100 0.9992395 0.881 1.442828e-07
##
## [[90]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9997182 0.891 7.952782e-08
## 2 0.00089        100 0.9994354 0.891 7.952782e-08
##
## [[91]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9997966 0.901 4.142453e-08
```

```
## 2 0.00089        100 0.9995925 0.901 4.142453e-08
##
## [[92]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9998582 0.911 2.014382e-08
## 2 0.00089        100 0.9997158 0.911 2.014382e-08
##
## [[93]]
##       eta iterations         x alpha          fx
## 1 0.00089        100 0.9999052 0.921 8.99378e-09
## 2 0.00089        100 0.9998101 0.921 8.99378e-09
##
## [[94]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9999400 0.931 3.600754e-09
## 2 0.00089        100 0.9998798 0.931 3.600754e-09
##
## [[95]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9999647 0.941 1.247964e-09
## 2 0.00089        100 0.9999293 0.941 1.247964e-09
##
## [[96]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9999812 0.951 3.540664e-10
## 2 0.00089        100 0.9999623 0.951 3.540664e-10
##
## [[97]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9999914 0.961 7.462135e-11
## 2 0.00089        100 0.9999827 0.961 7.462135e-11
##
## [[98]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9999969 0.971 9.618614e-12
## 2 0.00089        100 0.9999938 0.971 9.618614e-12
##
## [[99]]
##       eta iterations         x alpha           fx
## 1 0.00089        100 0.9999993 0.981 4.528562e-13
## 2 0.00089        100 0.9999987 0.981 4.528562e-13
##
## [[100]]
##       eta iterations x alpha           fx
## 1 0.00089        100 1 0.991 5.336769e-18
## 2 0.00089        100 1 0.991 5.336769e-18
```

From the above we can see when we have the iterations=100, we could get the minimum value of function close to 0 when we have x_1 and x_2 both equals to 1 and the alpha is 0.991 nearly equals to 1.

To summarise, we can see through the comparison of gradient decent with momentum and gradient decent which indicates gradient decent with momentum runs more quickly than that without momentum because we have less iterations but still get the same value.

Q2 Support vector machines

Run the following code to load the tiny MNIST dataset:

```
library(knitr)

library(grid)
grid.raster(array(aperm(array(train.X[1:50,],c(5,10,28,28)),c(4,1,3,2)),c(140,280)),
              interpolate=FALSE)
```

(a) Use three-fold cross validation on the training set to compare SVMs with linear kernels, polynomial kernels and RBF kernels, i.e.

```
library(e1071)
load("~/Downloads/mnist.tiny.rdata")
train.X=train.X/255
test.X=test.X/255
svm(train.X,train.labels,type="C-classification",kernel="linear",cross=3)$tot.accuracy
```

```
## [1] 85.7
```

```
svm(train.X,train.labels,type="C-classification",kernel="poly",
degree=2,coef=1,cross=3)$tot.accuracy
```

```
## [1] 81.7
```

Linear kernels: We create one function *lin.acc* It can return the accuracy of linear model , max accuracy, and its relevant cost value with different input a list of *cost* and *train.X*. Here we set $cost = c(0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10)$ and print out the list of results.

```
lin.acc<-function(cost,train.X){
set.seed(123)
linear.acc<-rep(0,length(cost))
  for(i in 1:length(cost)){
  linear.acc[i]<-svm(train.X,train.labels,type="C-classification",kernel="linear",
                     cross=3,cost=cost[i])$tot.accuracy
  }
a<-linear.acc
## Get the max accuracy
max<-max(a)
## Locate the position of the max accuracy
p<-which(a==max,arr.ind = TRUE)
## Print out the list of results
list(a,paste("max accuracy is",max,", cost value is",cost[p]))
}

## Give the range of cost
cost = c(0.001,0.01,0.1,0.5,1,2,5,10)
## Give a random example here
lin.acc(cost,train.X)
```

```
## [[1]]
## [1] 64.1 85.6 86.6 85.4 86.1 86.1 86.3 86.3
##
## [[2]]
## [1] "max accuracy is 86.6 , cost value is 0.1"
```

From the result of the linear kernel SVM, it shows that we can get better accuracy with the values of $cost \geq = 0.1$. Here the maximum accuracy is 86.6, which can be obtained at the $cost = 0.1$.

Polynomial kernels: Similarly, for the polynomial kernel, with input of some values of *cost*, *gamma* and *train.X*, we can get the results of all accuracies, the max accuracy, and the its *cost* and *gamma*

```
set.seed(123)

## Three input in this function

poly.acc<-function(cost,gamma,train.X){

## Initialize the grid matrix
acc.grid<-matrix(nrow=length(gamma),ncol=length(cost),
                 dimnames=list(gamma,cost))

  for(i in 1:length(cost)){
    for (j in 1:length(gamma)) {

       s<-svm(train.X,train.labels,type="C-classification",kernel="poly",degree=2,coef=1,
              cross=3,cost = cost[i],gamma = gamma[j])
       ##Define the grid with the accuracy entries
        acc.grid[j,i]<-s$tot.accuracy
    }
  }
   a<-acc.grid
   ## Same as the linear kernel step
   max<-max(a)
   p<-which(a==max,arr.ind = TRUE)
   g<-gamma[p[,1]]
   c<-cost[p[,2]]
list(a,paste("max accuracy is",max,", gamma is",g,", and cost is",c))
}

## Give the range of cost and gamma
c=c(0.001,0.01,0.1,0.5,1,2,5,10)
g=c(0.001,0.01,0.1,0.5,1,2,5,10)
## Give a random example here
poly.acc(c,g,train.X)
```

```
## [[1]]
##       0.001 0.01  0.1  0.5    1    2    5   10
## 0.001  10.1 14.0 14.4 60.9 78.3 83.1 86.6 87.3
## 0.01   10.6 14.6 81.4 87.1 88.2 88.0 87.9 88.5
## 0.1    62.8 87.5 90.1 88.7 88.4 88.6 88.8 89.3
## 0.5    88.2 88.9 88.7 88.3 88.0 89.3 88.5 89.6
## 1      88.6 87.1 88.8 88.2 89.0 88.8 88.3 88.8
## 2      87.7 88.0 87.9 88.5 89.3 88.6 87.9 87.5
```

```
## 5      89.0 88.0 87.9 88.5 87.7 88.5 88.8 87.4
## 10     88.9 87.4 87.5 88.6 88.8 88.9 88.2 88.0
##
## [[2]]
## [1] "max accuracy is 90.1 , gamma is 0.1 , and cost is 0.1"
```

From the result of the polynomial kernels,it shows that we can get better accuracy with the values of $\gamma \geq 0.1$. Here the maximum accuracy is 90.1, which can be obtained at the $cost = 0.1$ and $\gamma = 0.1$.

RBF kernel:

```
set.seed(123)
rbf.acc<-function(cost,gamma,train.X){
## Same as the polynomial kernel step
n<-length(cost)
m<-length(gamma)
acc.grid<-matrix(nrow=length(gamma),ncol=length(cost),
                 dimnames=list(gamma,cost))

  for(i in 1:n){
    for (j in 1:m) {
     ## we only need to change the part of " kernel ='radial' "
       s<-svm(train.X,train.labels,type="C-classification",kernel="radial",degree=2,coef=1,
              cross=3,cost = cost[i],gamma = gamma[j])
        acc.grid[j,i]<-s$tot.accuracy
    }
  }
   a<-acc.grid
   max<-max(a)
   p<-which(a==max,arr.ind = TRUE)
   g<-gamma[p[,1]]
   c<-cost[p[,2]]
list(a,paste("max accuracy is",max,", gamma is",g,", and cost is",c))
}

c=c(0.001,0.01,0.1,0.5,1,2,5,10)
g=c(0.001,0.01,0.1,0.5,1,2,5,10)
## Give a random example here
rbf.acc(c,g,train.X)
```

```
## [[1]]
##       0.001 0.01  0.1  0.5    1    2    5   10
## 0.001  10.1 14.2 14.5 57.3 76.8 82.9 86.5 87.3
## 0.01   10.6  9.5 59.1 87.0 88.7 89.0 90.4 90.4
## 0.1    12.2 11.3 12.2 20.5 50.5 56.7 53.6 50.4
## 0.5    10.1 10.9 12.4 10.6 12.1 12.5 14.3 13.4
## 1      12.2 12.4 10.8 12.2 10.6 10.2 12.2 10.2
## 2      10.6  8.8 12.2 10.4 10.4 10.1 10.6 10.1
## 5      10.7 12.2 10.8 10.3 12.2 11.1 10.7 10.6
## 10      9.9  9.3 12.2 10.8 10.1 12.2 13.2 12.2
##
## [[2]]
## [1] "max accuracy is 90.4 , gamma is 0.01 , and cost is 5"
## [2] "max accuracy is 90.4 , gamma is 0.01 , and cost is 10"
```

From the result of the RBF kernels,it shows that we can get better accuracy with the values of $\gamma \geq 0.01$. Here the maximum accuracy is 90.4, which can be obtained at the $cost = 10$ and γ=0.01

conclusion: The best accuracy is RBF kernels (90.4), then Polynomial kernels (90.1), and linear kernels (86.6) respectively. RBF kernels need relatively small range of values of γ and *cost*. Linear kernels can get better accuracy when the $cost \geq 0.1$.

Warning explanation: Warning shows that $Variable(s) : 'X1' and 'X2' and ... and 'Xi' constant. Cannot Scale data.$ This happens when svm functions have a lot of zero variables and the warning just tells us we should realize the marix is very sparse.

(b) We can use same steps as three svm functions in the first part, just changing the input into pairs of (lc,lg) entries ,x and y. The output will return four parameters: The accuracy grid (matrix), the best cross-validation error (max accuracy), the corresponding gamma and cost.

```
set.seed(123)
rbf.search<-function(lc,lg,x,y){

## Initialize the grid matrix
grid<-matrix(nrow=length(lg),ncol=length(lc), dimnames=list(lg,lc))

for(i in 1:length(lg)){
  for(j in 1:length(lc)){

    ## The pair of lists attempts cross-validation with parameters cost =exp(lc) and
    ## gamma=exp(lg)
    s<-svm(x,y,type="C-classification",kernel="radial",degree=2,coef=1,cross=3,
           gamma =exp(lg[i]),cost =exp(lc[j]))

   ## Same as before, define the grid matrix
     grid[i,j]<-s$tot.accuracy

  }
}
   ## looking for the best (max) accuracy of all elements
   a<-grid
   max<-max(a)
   p1<-which(a==max,arr.ind = TRUE)[1]
   p2<-which(a==max,arr.ind = TRUE)[2]

## Return the best list of all parameters
list(a,max,lg[p1],lc[p2])
}
```

We choose the *log.gamma.range* (lg) $\in [-4, 4]$ and *log.C.range* (lc) $\in [-4, 4]$ with inputs *train.X* and *train.lebales*.

```
k<-rbf.search(c(-4:4),c(-4:4),train.X,train.labels)
## Print out all accuracy
k[[1]]
```

```
##      -4   -3   -2   -1    0    1    2    3    4
## -4 10.1 21.2 67.6 87.1 90.1 90.5 91.6 92.4 90.6
## -3  9.5 13.2 19.7 59.6 87.9 87.8 87.4 89.1 87.8
```

```
## -2 12.2 16.8 10.8 18.4 28.6 33.5 33.2 32.3 33.1
## -1 10.6 12.3 10.4 10.3 14.3 14.7 16.1 16.5 16.2
## 0  10.6 10.4 12.2 10.1 10.6 12.2 10.2 10.7 10.2
## 1  10.1 15.3 12.2 15.2 12.2 10.7 12.2 12.2 10.6
## 2  10.7 13.2 12.2 12.2 12.5 11.0 10.2 10.1 10.6
## 3  12.2  9.8 10.9  8.8 10.9 12.2 13.4 10.7 10.9
## 4  10.1  9.9 10.2 10.8 10.6 11.1 12.2 12.2 10.8
```

Here we can see a better performance region from matrix k when $gamma \in [-4, -3]$, and $cost \in [0, 4]$, and we can run the second round for the $gamma \in [-4, -3]$ and $cost \in [0, 4]$ with gaps of 0.25 and 0.5 respectively.

```
## Choosing the better accuracy from the above results and run the second time
## New input with the better results
new.c<- seq(0, 4, 0.5)
new.g <-seq(-4,-3,0.25)

## Second running and get the final best parameters
k2 <- rbf.search(new.c, new.g, train.X, train.labels)
k2[[1]]
```

```
##          0  0.5    1  1.5    2  2.5    3  3.5    4
## -4    89.9 91.9 90.3 91.3 90.4 90.5 91.5 91.1 90.8
## -3.75 89.0 91.2 91.1 90.3 91.1 90.9 91.1 90.7 90.2
## -3.5  89.7 90.4 91.7 89.2 90.9 90.6 89.6 90.3 91.0
## -3.25 89.1 90.1 89.5 90.0 89.8 89.5 90.6 90.6 90.4
## -3    88.0 87.6 88.5 87.4 89.1 87.7 87.3 89.1 88.3
```

```
## Print out the best cross-validation error, the corresponding gamma and cost values.

best.cve<-k2[[2]]
best.g<-k2[[3]]
best.c<-k2[[4]]

## A list of three best parameters
list(best.cve, best.g, best.c)
```

```
## [[1]]
## [1] 91.9
##
## [[2]]
## [1] -4
##
## [[3]]
## [1] 0.5
```

From the results $k2$ of the second round, we can get the best values for *gamma* and *cost*. By using these best parameters (*gamma*=-3.75 and *cost* =0.5) and svm function to get the final best model *best*. And we have the test data *test.X* and *test.labels* to test accuracy and it returns the mean accuracy predict values through the final model *best*.

```
## Confirm the best svm and predict accuracy by using the test.X and test.labels
best<-svm(train.X,train.labels,type="C-classification",kernel="radial",degree=2,coef=1,
          cross=3,cost = exp(best.c),gamma = exp(best.g))
best.acc<-mean(predict(best,test.X)==test.labels)
best.acc
```

```
## [1] 0.912
```

We can get the final best accuracy which is around 91% when c and γ are not the extreme values from the ranges of *cost* and *gamma* ranges and it shows a perfect and ideal result. After two second runs and choosing the better pair of c and γ from a range of values, we can get a suitable answer to test a better accuracy.