R 語言報告

Gradient Descent

組員：

統計四劉明翰

統計四許邦德

統計四謝佳叡

統計二黎明翰

統計二林育安

目錄

1.Basic Idea：

為了找出任意函數的最小值，必須要找出這個函數微分等於 0 的點，但是大多數程式語言沒有辦法解方程式，利用梯度（Gradient）找到這個點的演算法就稱「Gradient Descent」。

2.Formula：

在這個演算法裡面，利用梯度（partial derivative with respect to Xi）來找出函數的最小值，以二維平面為例：

如下圖所示，當 X 在 X1 時，函數最小值 Xn 在 X1 的右邊，也就是說 X2 必須要往右移（X1<X2），但此時梯度（也就是斜率）小於 0，因此推導出 Xj+1 = Xj － $\frac{\partial f}{\partial x}$，直到收斂於函數最小值。但是有時候$\frac{\partial f}{\partial x}$相對於 x 可能會太大或大小，可能會導致收斂到最小值需要耗費非常久的時間和資源，因此為了控制每一次在函數圖形上 X「移動」的大小，因此在$\frac{\partial f}{\partial x}$前加上一個常數$\eta$，並將方程式改成 Xj+1 = Xj － η＊$\frac{\partial f}{\partial x}$，這個常數$\eta$通常為[0,1]之間的實數。若把它擴展為 n 維度，可以推導出：

X1,j+1 = X1,j － $\eta*\frac{\partial f}{\partial x_1}$

X2,j+1 = X2,j － $\eta*\frac{\partial f}{\partial x_2}$

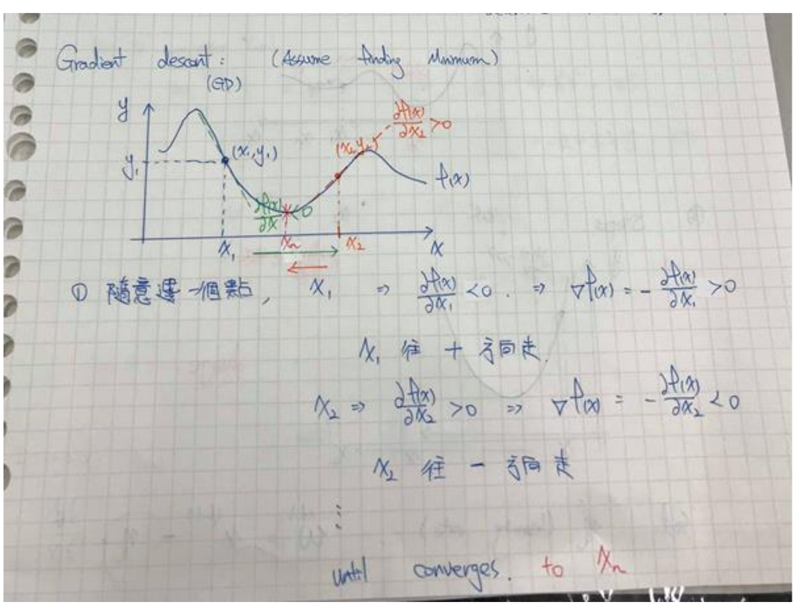
.　　.　　.

.　　.　　.

.　　.　　.

Xn,j+1 = Xn,j － $\eta*\frac{\partial f}{\partial x_n}$

,where j = 0,1,2,3,……,m, η ∈ [0,1],and ∈ R ,we call it “learning rate”.



Take Linear Regression as example:

$$Y = w_1 X \qquad (Y = 10,\ X = 5,\ w_{1\,correct} = 2)$$

$$\mathcal{L} = MSE = (Y - \hat{Y})^2$$

iter 1:

$w_1^{(0)} = 0 \Rightarrow \hat{Y} = 0 \Rightarrow \mathcal{L} = (10-0)^2 = 100$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial w_1} = 2(Y - \hat{Y}) \times (-1) \times X$$

$$= 2 \times 10 \times (-1) \times 5$$

$$= -100$$

Let $\eta = 0.01$,

$$w_1^{(1)} = w_1^{(0)} - \eta \times \frac{\partial \mathcal{L}}{\partial w_1}$$

$$= 0 - 0.01 \times (-100)$$

$$= 1.$$

iter 2:

$w_1^{(1)} = 1 \Rightarrow \hat{Y} = 5 \Rightarrow \mathcal{L} = (10-5)^2 = 25$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial w_1} = -2 \times 5 \times 5 = -50$$

$$w_1^{(2)} = w_1^{(1)} - \eta \times \frac{\partial \mathcal{L}}{\partial w_1}$$

$$= 1 - 0.01 \times (-50) = 1.5$$

iter 3:

⋮

until $\frac{\partial \mathcal{L}}{\partial w_1} = 0$.

3.Problems of the Gradient Descent：

1.如果函數是不可微的，那麼這個演算法將無法找到最小值。

2.如果剛好起始值(X1)取到剛好是極值的位置（微分=0），那麼將會卡在 X1 不動。

—> 可以在起始值加一個極小正數ξ來解決這個問題。

3.如果函數有 local minimum,那麼很有可能因為起始值（也就是上述的 X1）選擇不當，而造成找不到 global minimum 的問題。

—> 可以取一個以上的起始值，並比較這幾個起始值收斂到的函數最小值，以此找到 global minimum。

4.如果$\frac{\partial f}{\partial x}$太大，X 會在函數圖形上亂跳，很有可能會讓 Xn 發散，或者找到不是最小值的點。

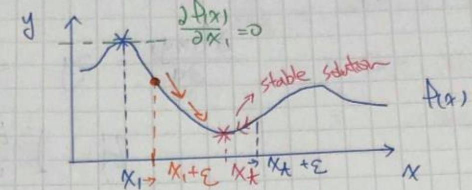
—>要將 learning rate 設定非常小。

Facing Problem:

① Stochastic point = Max or Min (i.e. $\frac{\partial f(x)}{\partial x} = 0$)

Sol. ⇒ test if its a stable dynamic system solution.

$$x^{(t)} = x^{(t-1)} + \varepsilon \quad , \quad \varepsilon << \ (很小)$$

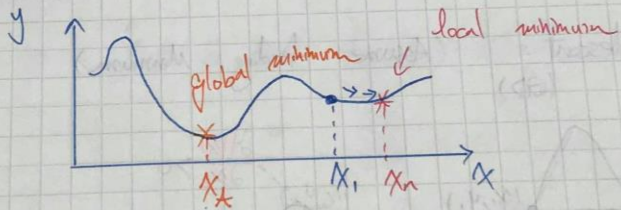


if choose Max ⇒ system solution will change

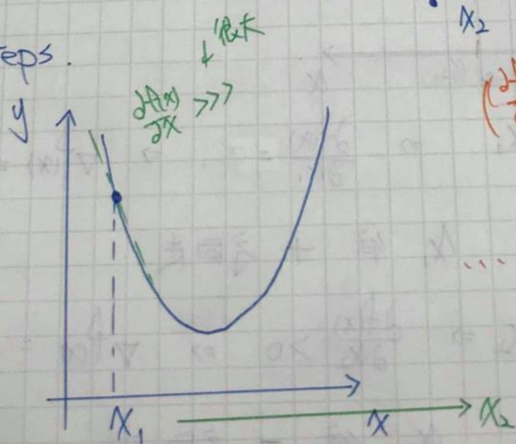
Min ⇒ system solution unchanged.

② local minimum problem.

(Hard to find global minimum)



③ Steps.



• x_2

$\left(\frac{\partial f(x)}{\partial x_2} \times 偏大 \right)$

...... diverge.

small , EX: 0.000

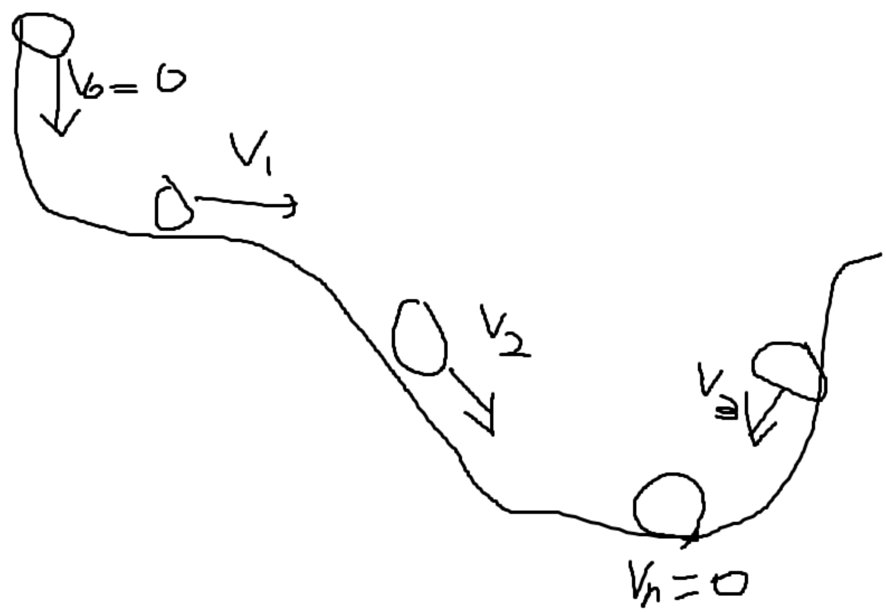
Sol. η (learning rate) , $x^{(t)} = x^{(t-1)} - \eta \cdot \frac{\partial f}{\partial x}$

4.Method to Enhance Gradient Descent:

1.Momentum

—>為了解決只能找到local minimum的問題

如下圖所示，想像現在有一個滑梯，將一顆球以初速為0從頂端往下放，當遇到一個平坦的地面（想像為local minimum），會因為慣性而繼續有一個V1的速度往前前進，並且最後收斂到global minimum。



為了在電腦裡面模擬慣性，創造一個新的變數V，在斜坡的時候V會變大，在爬坡的時候V會變小，而坡度也就是斜率，並且為了讓V不要膨脹的太快，Vj前面會有一個常數，因此推導出：

Vj+1 = β*Vj - $\eta*\frac{\partial f}{\partial x_i}$,where β, η are in [0,1], β usually is 0.9.

X1, j+1 = X1, j + Vj+1

X2, j+1 = X2, j + Vj+1

.　　.　　.

.　　.　　.

.　　.　　.

Xn, j+1 = Xn, j + Vj+1

2.Adagrad Gradient Descent：

因為X有可能越跑越快以至於難以收斂到最小值，為了讓X越跑越慢，讓收斂到最小值更加容易，所以創造新的一個變數n：

$$n = \sum_{r=0}^{j} \left(\frac{\partial \mathrm{f}}{\partial \mathrm{x}_{\mathrm{i},\mathrm{r}}}\right)^2$$

並將原本的公式Xi, j+1 = Xi, j － $\eta * \frac{\partial f}{\partial x_i}$改為：

Xi, j+1 = Xi, j － $\eta * \frac{1}{\sqrt{n+\xi}} * \frac{\partial f}{\partial x_i}$, where n =$\sum_{r=0}^{j} \left(\frac{\partial \mathrm{f}}{\partial \mathrm{x}_{\mathrm{i},\mathrm{r}}}\right)^2$, $\xi \approx 0$ is a positive number.

（為了避免n=0，所以加上一個極小正數ξ來避免這個情況）

3. Adam :

Adam這個演算法主要是結合了上述的Momentum 和 Adagrad Gradient Descent,以下為常見到的算法：

Vj+1 = α*Vj + (1-α)*$\frac{\partial f}{\partial x_i}$ <— the idea of Momentum

Nj+1 = β*Nj + (1-β)*$\left(\frac{\partial f}{\partial x_i}\right)^2$ <— the idea of AGD

,where j = 0,1,2,3……,m ,and let V0 and N0 be 0, α be 0.9 and β be 0.999.

->Xi, j+1 = Xi, j － $\eta * \frac{Vj+1}{\sqrt{Nj+1+\xi}}$

但Vj+1和Nj+1會偏向於V0和N0（biased to V0,N0），為了解決這個問題，必須要讓誤差（bias）等於0，也就是讓E($\widehat{\mathrm{Vj}+1}$)=E($\frac{\partial f}{\partial x_i}$)和E($\widehat{\mathrm{Nj}+1}$)=E($\left(\frac{\partial f}{\partial x_i}\right)^2$)

令 t = j+1（也就是j+1跑到t） ,Gj+1 = $\frac{\partial f}{\partial x_i}$ |xi = xi, j

則Vt = α*Vt-1 + (1-α)*Gt —*Equation 1*

將遞迴展開後可以發現：

Vt = (1- α)$\sum_{j=1}^{t} \alpha^{t-j}\,\mathrm{Gj}$

E(Vt) = E[(1- α)$\sum_{j=1}^{t} \alpha^{t-j}\,\mathrm{Gj}$]

= (1- α)E(Gt)$\sum_{j=1}^{t} \alpha^{t-j}+\zeta$ -> approximate Gj with Gt, ζ is bias.

= (1- αˆt) E(Gt) +ζ

biased to V0, when α -> 1, then ζ -> 0. (from *Equation 1*)

-> E(Vt) = (1- αˆt) E(Gt)

To correct bias, make E($\widehat{\mathrm{V}}$t)=E(Gt).

-> E($\widehat{\mathrm{V}}$t) = $\frac{E(Vt)}{(1-\alpha\text{^}t)}$ = E(Gt)

-> $\widehat{\mathrm{V}}\mathrm{t} = \frac{Vt}{(1-\alpha\text{^}t)}$

從以上的推導過程可以得到以下式子

$\widehat{\mathrm{Vj}+1} = \frac{Vj+1}{1-\alpha\text{^}(j+1)}$

$\widehat{\mathrm{Nj}+1} = \frac{Nj+1}{1-\beta\text{^}(j+1)}$

->Xi, j+1 = Xi, j － $\eta * \frac{\widehat{\mathrm{Vj+1}}}{\sqrt{\widehat{\mathrm{Nj+1}}+\xi}}$

5. Implement the Basic Method of Gradient Descent:

```
#derivative

library(stats) #import package "stats"
f1.exp <- expression(x^2 + 3*x +1) # define the function we want to differentiate
dx1 <- D(f1.exp,"x") # first derivative
x <- 1:5 #define x
eval(dx1) #get value
dx2 <- D(dx1, "x") # second derivative
eval(dx2) #get value

#partial derivative

f2.exp <- expression(x^2*y+y^2) #define the function we want to differentiate
dx1 <- D(f2.exp,"x") #first partial derivative with respect to x
dy1 <- D(f2.exp,"y") #first partial derivative with respect to y
x <- 1:5 #define x
y <- 2:6 #define y
eval(dx1) #get value
eval(dy1)
#if length(x)/length(y) == integer
x <- 1:5
y <- 1:10
eval(dx1) #calculable
#if length(x)/length(y) != integer
eval(dy1)
x <- 1:5
y <- 2:5
eval(dx1)#calculable, but a warning message appears
```

```
eval(dy1)

#the shoter vector will repeat


#gradient descent


#2-dimentional case

#find the minimum value of f(x) = 5*x^6-3*x^3-2*x^2+x-10


f <- expression(5*x^6-3*x^3-2*x^2+x-10) #define function

# define gradientdescent function, where gamma is our learning rate(0 to 1 usually), and iter is how many times we try to find our minimum.

gradientDesc_D2 <- function(f,initial_value,gamma,iter)

{

  l <- gamma

  dx <- D(f,"x") #find derivative

  result <- numeric(length = length(initial_value)) #create a vector to store results

  x_ans <- numeric(length = length(initial_value)) #create a vector to store x

  for (i in 1:length(initial_value)) {

    x <- initial_value[i] #take initial x

    for (j in 1:iter) {

      x <- x-l*eval(dx) #find min x

    }

    result[i] <- eval(f)

    x_ans[i] <- x

  }

  ans <- min(result) #find min value

  x <- unique(round(x_ans[grep(ans,result)],10)) #find x value when the function is minimized

  if (is.nan(ans) == TRUE){

    print("The minimum of the function does not exist")

  }else{

      paste("The minimum of the function is",ans,"at postion x =", x , sep = " ")
```

```
  }

}

#test
set.seed(100)
x <- runif(3,0,1)
gradientDesc_D2(f,x,0.12,500)
#check
fx <- function(x){
  return(5*x^6-3*x^3-2*x^2+x-10)
}
curve(fx,-1,1)
#we find that Gradint Descent only found local minimum
#try to adjust our learning rate
gradientDesc_D2(f,x,0.0012,500)
#we find the global minimum
#we can alter any parameter to find global minimum

#3-dimentional case

#find the minimium of f(x,y) = (x+y)^2-(x+y)-2

f <- expression((x+y)^2-(x+y)-2)

gradientDesc_D3 <- function(f,initial_value_x,initial_value_y,gamma,iter)
{
  l <- gamma
  dx <- D(f,"x") #find partial derivative with respect to x
  dy <- D(f,"y") #find partial derivative with respect to y
```

```
  result <- numeric(length = length(initial_value_x)) #create a vector to store results
  x_ans <- numeric(length = length(initial_value_x)) #create a vector to store x
  y_ans <- numeric(length = length(initial_value_y))
  for (i in 1:length(initial_value_x)) {
    x <- initial_value_x[i] #take initial x
    y <- initial_value_y[i] #take initial y
    for (j in 1:iter) {
      x_new <- x - l*eval(dx) #store new-found x temporarily
      y <- y - l*eval(dy) #find min x and min y
      x <- x_new
    }
    result[i] <- eval(f)
    x_ans[i] <- x
    y_ans[i] <- y
  }
  ans <- min(result) #find min value
  x <- x_ans[grep(ans,result)]
  y <- y_ans[grep(ans,result)]#find x value when the function is minimized
  if (is.nan(ans) == TRUE){
    print("The minimum of the function does not exist")
  }else{
    paste("The minimum of the function is",ans,"at postion x =", x ,", y =",y, sep = " ")
  }

}
#test
set.seed(10)
x <- runif(10,-1,1)
y <- runif(10,-1,1)
gradientDesc_D3(f,x,y,0.2,50000) #minimum at several (x,y)
```

6.Question:

試著以上述Momentum的方式改變 gradientDesc_D3 function 裡的程式，然後找出例題中函數（(x+y)^2-(x+y)-2）的最小值吧！

7.Reference:

[1] Sebastian Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747v2, 2017.
[2] Diederik P. Kingma, Jimmy Lei Ba. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. arXiv preprint arXiv:1412.6980v9, 2017.