機器學習概論作業

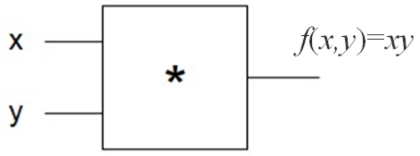
範圍： Hacker's guide to Neural Networks

銘傳大學電腦與通訊工程系

班　　級	電通三乙
姓　　名	李柏賢
學　　號	07050862
作業成果	應繳作業共 5 題，每題 20 分 我共完成 5 題，應得 100 分
授課教師	陳慶逸

■ 請確實填寫自己寫完成題數，填寫不實者(如上傳與作業明顯無關的答案，或是計算題數有誤者)，本次作業先扣 50 分。

一、　針對 f(x,y) = x.y。 初始值 x = -2, y = 3；當 h = 0.0001，且 learning rate = 0.01，迭代次數 = 300 時，利用 Numerical Gradient 找出讓 f(x,y) 變大的輸入值。



$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+h,y) - f(x,y)}{h}$$

(1) 求迭代完成後，此時的 x, y 和輸出 f。

x = 9.77 , y = 10.01 , f = 97.88

(2) 程式碼：

```
import numpy as np

def forwardMultGate(x, y):
    return x*y

def numericalGradient(x, y):
    h = 0.0001
    out = forwardMultGate(x, y)
    xh_out = forwardMultGate(x+h, y)
    x_derivative = (xh_out-out)/h
    yh_out = forwardMultGate(x, y+h)
    y_derivative = (yh_out-out)/h

    step_size = 0.01;
    better_x = x + step_size*x_derivative
    better_y = y + step_size*y_derivative
    better_out = forwardMultGate(better_x, better_y)
    print(better_x, better_y, better_out)
    return [better_x, better_y, better_out]

x,y = -2, 3
better_x, better_y, better_out=numericalGradient(x,y)
print(better_x, better_y, better_out)

for i in range(1,300):
    better_x, better_y, better_out=numericalGradient(better_x, better_y)
    print(better_x, better_y, better_out)
```

(3) 執行結果擷圖：

```
8.379841445281706 8.664943577054899 72.61085330803216
8.379841445281706 8.664943577054899 72.61085330803216
8.466490881051165 8.748741991507373 74.07114429176659
8.466490881051165 8.748741991507373 74.07114429176659
8.553978300965277 8.833406900317712 75.56077094891465
8.553978300965277 8.833406900317712 75.56077094891465
8.642312369967648 8.918946683327093 77.08032324839967
8.642312369967648 8.918946683327093 77.08032324839967
8.731501836800454 9.005369807025957 78.6304030111145
8.731501836800454 9.005369807025957 78.6304030111145
8.821555534871301 9.092684825393874 80.21162414829362
8.821555534871301 9.092684825393874 80.21162414829362
8.912482383124349 9.180900380742244 81.82461290458488
8.912482383124349 9.180900380742244 81.82461290458488
9.004291386931335 9.270025204572633 83.47000810616974
9.004291386931335 9.270025204572633 83.47000810616974
9.09699163897691 9.360068118442214 85.14846141372315
9.09699163897691 9.360068118442214 85.14846141372315
9.190592320161024 9.451038034832052 86.86063758047719
9.190592320161024 9.451038034832052 86.86063758047719
9.285102700509846 9.54294395803386 88.6072147155543
9.285102700509846 9.54294395803386 88.6072147155543
9.380532140089521 9.635794985038174 90.38888455246402
9.380532140089521 9.635794985038174 90.38888455246402
9.476890089939822 9.729600306438172 92.20635272315937
9.476890089939822 9.729600306438172 92.20635272315937
9.57418609300354 9.824369207337739 94.0603390374252
9.57418609300354 9.824369207337739 94.0603390374252
9.67242978507577 9.92011106826665 95.95157776796216
9.67242978507577 9.92011106826665 95.95157776796216
9.771630895758882 10.016835366117064 97.88081794127974
9.771630895758882 10.016835366117064 97.88081794127974
```

二、同第一題之 f(x,y) = x.y。 初始值 x = -2, y = 3；且 learning rate = 0.01, 迭代次數 = 300 時，利用 Analytic Gradient 找出讓 f(x,y)變大的輸入值。

(1) 求迭代完成後，此時的 x, y 和輸出 f。

x = 9.77 , y = 10.01 , f = 97.88

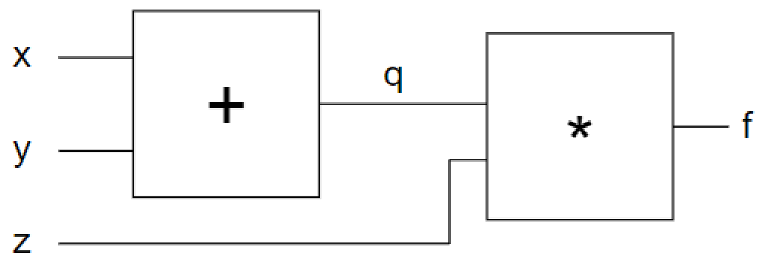
(2) 程式碼：

```
def forwardMultGate(x, y):
    return x*y

def analyticGradient(x, y):
    x_derivative = y
    y_derivative = x
    step_size = 0.01;
    better_x = x + step_size*x_derivative
    better_y = y + step_size*y_derivative
    better_out = forwardMultGate(better_x, better_y)
    return [better_x, better_y, better_out]

print(analyticGradient(-2,3))
x, y=-2, 3;
better_x, better_y, better_out = analyticGradient(x,y)
print(better_x, better_y, better_out)

for i in range (1,300):
    better_x, better_y, better_out = analyticGradient(better_x, better_y)
    print(better_x, better_y, better_out)
```

(3) 執行結果擷圖：

```
8.294021411668384 8.582003362953088 71.179319647343
8.379841445297915 8.664943577069772 72.61085330829725
8.466490881068612 8.748741991522751 74.07114429204942
8.55397830098384 8.833406900333436 75.56077094921314
8.642312369987174 8.918946683343275 77.08032324871367
8.731501836820607 9.005369807043147 78.63040301144608
8.821555534891038 9.092684825411354 80.21162414862728
8.912482383145152 9.180900380760264 81.82461290493647
9.004291386952755 9.270025204591715 83.47000810654012
9.096991638998672 9.360068118461243 85.14846141409996
9.190592320183285 9.45103803485123 86.86063758086384
9.285102700531796 9.542943958053062 88.60721471594208
9.380532140112328 9.63579498505838 90.38888455287332
9.476890089962911 9.729600306459504 92.20635272358618
9.574186093027507 9.824369207359133 94.06033903786549
9.672429785101098 9.920111068289408 95.95157776843355
9.771630895783993 10.01683536614042 97.88081794175949
```

三、如下圖；　初始值 x = -2, y = 5, z = -4 (f = -12)；learning rate = 0.01，迭代次數 = 300 時，利用 Analytic Gradient 找出讓 f(x,y, z)變大的輸入值。



(1) x = -2, y = 5, z = -4，可得到 f = -12，這三個變數的梯度(Analytic gradiet)為

gradiet_x = _____-4_____ , gradiet_y = _______-4_______ ，gradiet_z = _______3_____

若 learning rate = 0.01，則一次迭代後，新的 x, y, z 的值為何:

x = _____-2.04_____ , y = _______4.96_______ ， z = _______-3.97_______

(2) 程式碼：

```
import numpy as np

def forwardMultGate(x, y):
    return x*y

# f(x,y,z) =(x+y)*z
def forwardAddGate(x, y):
    return x+y

def forwardCircuit(x,y,z):
    q = forwardAddGate(x,y)
    f = forwardMultGate(q,z)
    return f

def chainRule(x,y,z):
    # f = mult(p,z)
    # q = add(x,y)
    f = forwardCircuit(x,y,z)
    q = forwardAddGate(x,y)


    # MULT gate
    derivative_f_wrt_q = z
    derivative_f_wrt_z = q
```

```
# ADD gate
    derivative_q_wrt_x = 1
    derivative_q_wrt_y = 1
    # chain rule
    derivative_f_wrt_x = derivative_f_wrt_q*derivative_q_wrt_x
    derivative_f_wrt_y = derivative_f_wrt_q*derivative_q_wrt_y
    print("Analytic gradiet:", [derivative_f_wrt_x, derivative_f_wrt_y,
derivative_f_wrt_z])


    step_size = 0.01
    x += step_size*derivative_f_wrt_x
    y += step_size*derivative_f_wrt_y
    z += step_size*derivative_f_wrt_z
    out = forwardCircuit(x, y, z)

    return [x, y, z, out]

x,y,z=-2,5,-4

better_x,better_y,better_z,better_out = chainRule(x,y,z)
print(better_x, better_y, better_z, better_out)

for i in range(1,300):
    better_x, better_y, better_z, better_out=chainRule (better_x, better_y,
better_z)
    print(better_x, better_y, better_z, better_out)
```

(3) 執行結果擷圖：

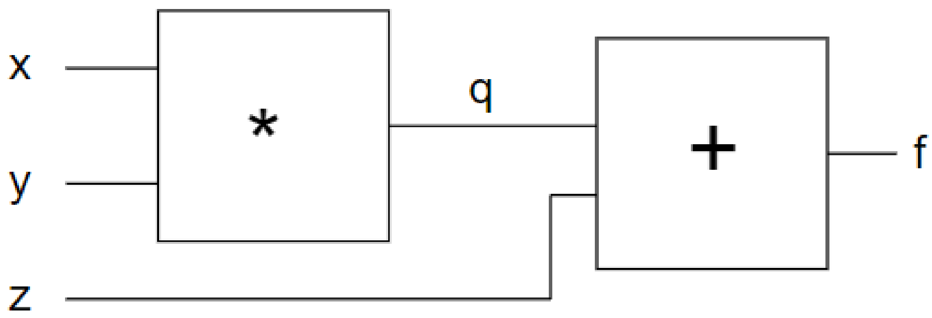
第一次迭代

```
for i in range(1,300):
        better_x, better_y, better_z, better_out=chainRu
        print(better_x, better_y, better_z, better_out)
```

```
Analytic gradiet: [-4, -4, 3]
-2.04 4.96 -3.97 -11.5924
```

```
-40.32461214197456 -33.32461214197454 -52.18203441924469 3843.166356535704
Analytic gradiet: [-52.18203441924469, -52.18203441924469, -73.64922428394911]
-40.84643248616701 -33.84643248616699 -52.918526662084176 3952.6363665059102
Analytic gradiet: [-52.918526662084176, -52.918526662084176, -74.69286497233401]
-41.37561775278785 -34.37561775278783 -53.66545531180751 4065.224543838679
Analytic gradiet: [-53.66545531180751, -53.66545531180751, -75.75123550557569]
-41.91227230590593 -34.91227230590591 -54.42296766686327 4181.01970743013
Analytic gradiet: [-54.42296766686327, -54.42296766686327, -76.82454461181183]
-42.456501982574565 -35.456501982574544 -55.19121311298139 4300.1132061131075
Analytic gradiet: [-55.19121311298139, -55.19121311298139, -77.9130039651491]
-43.008414113704376 -36.008414113704355 -55.97034315263288 4422.598990720714
Analytic gradiet: [-55.97034315263288, -55.97034315263288, -79.01682822740872]
-43.568117545230706 -36.568117545230685 -56.76051143490697 4548.573688202527
Analytic gradiet: [-56.76051143490697, -56.76051143490697, -80.13623509046138]
-44.13572265957978 -37.135722659579756 -57.56187378581158 4678.136677851949
Analytic gradiet: [-57.56187378581158, -57.56187378581158, -81.27144531915954]
-44.71134139743789 -37.71134139743787 -58.374588239003174 4811.390169704843
Analytic gradiet: [-58.374588239003174, -58.374588239003174, -82.42268279487575]
-45.295087279827925 -38.295087279827904 -59.19881506695193 4948.439285171295
Analytic gradiet: [-59.19881506695193, -59.19881506695193, -83.59017455965582]
-45.88707543049745 -38.887075430497426 -60.03471681254849 5089.39213996409
Analytic gradiet: [-60.03471681254849, -60.03471681254849, -84.77415086099487]
-46.487422598622935 -39.48742259862291 -60.88245832115844 5234.359929389369
Analytic gradiet: [-60.88245832115844, -60.88245832115844, -85.97484519724586]
-47.09624718183452 -40.0962471818345 -61.742206773130896 5383.4570160667035
Analytic gradiet: [-61.742206773130896, -61.742206773130896, -87.19249436366903]
-47.71366924956583 -40.71366924956581 -62.61413171676759 5536.801020147823
Analytic gradiet: [-62.61413171676759, -62.61413171676759, -88.42733849913165]
-48.33981056673351 -41.33981056673349 -63.498405101758905 5694.512912105147
```

四、計算下面電路每一個變數的梯度



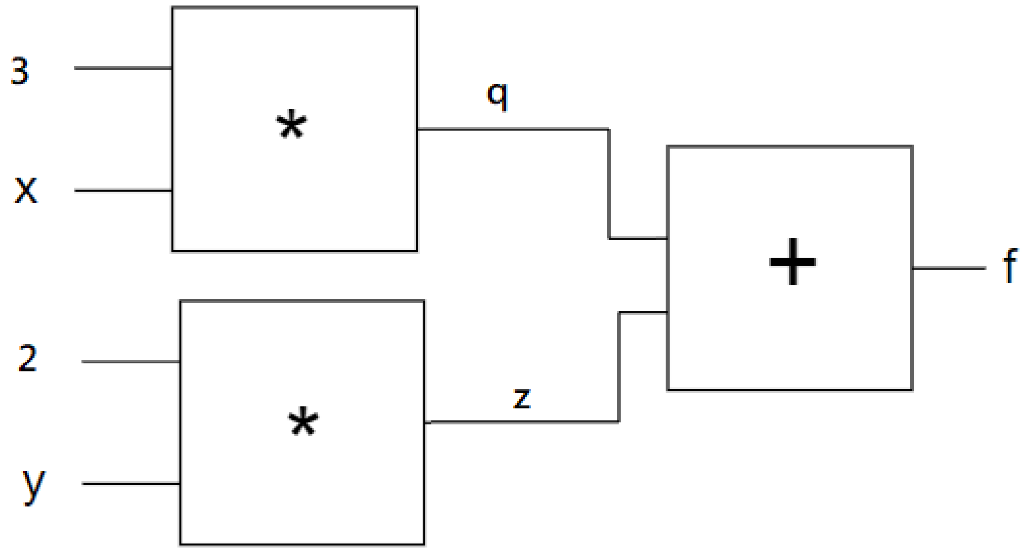
$$\frac{\partial f}{\partial q} = \frac{\partial}{\partial q}(q+z) = 1$$

$$\frac{\partial f}{\partial z} = \frac{\partial}{\partial z}(q+z) = 1$$

$$\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x} \cdot \frac{\partial f}{\partial q} = \frac{\partial}{\partial x}(x \cdot y) \times 1 = y$$

$$\frac{\partial f}{\partial y} = \frac{\partial q}{\partial y} \cdot \frac{\partial f}{\partial q} = \frac{\partial}{\partial y}(x \cdot y) \times 1 = x$$

五、計算下面電路每一個變數的梯度: f(x,y)= 3x+2y



$$\frac{\partial f}{\partial q} = \frac{\partial}{\partial q}(q+z) = 1$$

$$\frac{\partial f}{\partial z} = \frac{\partial}{\partial z}(q+z) = 1$$

$$\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x} \cdot \frac{\partial f}{\partial q} = \frac{\partial}{\partial x}(3 \cdot x) \times 1 = 3$$

$$\frac{\partial f}{\partial y} = \frac{\partial z}{\partial y} \cdot \frac{\partial f}{\partial z} = \frac{\partial}{\partial y}(2 \cdot y) \times 1 = 2$$