修改後的程式碼

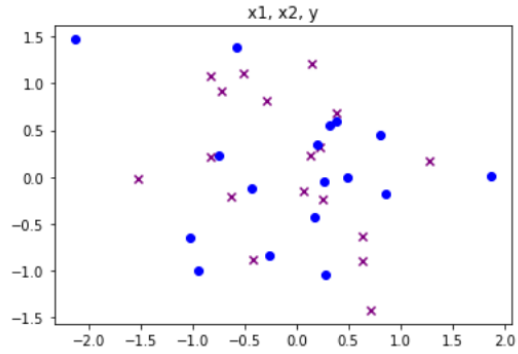
```
import matplotlib
import time
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import sklearn.datasets


n0 = 2; n1 = 3; n2 = 4; n3 = 1; m = 35;
alpha = 0.05
# load image dataset: blue/red dots in circles
train_X, train_Y = sklearn.datasets.make_circles(n_samples=35, noise=0.5)
A0=train_X.T; Y=train_Y.reshape(1,35)
print("np.shape(Y):",np.shape(Y),"np.shape(A0):",np.shape(A0))
for i in range(35):
    if(Y[0][i]<0.5):
        plt.scatter(A0[0][i], A0[1][i],c="blue",marker="o")
    else:
        plt.scatter(A0[0][i], A0[1][i],c="purple",marker="x")
plt.title('x1, x2, y')
plt.show()
```

```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.1;b1 = np.random.rand(n1,1)
WT2 = np.random.randn(n2,n1)*0.1;b2 = np.random.rand(n2,1)
WT3 = np.random.randn(n3,n2)*0.1;b3 = np.random.rand(n3,1)
Z1 = np.dot(WT1,A0) + b1
A1 = ((np.exp(Z1) - np.exp(-Z1))/(np.exp(Z1) + np.exp(-Z1)))
Z2 = np.dot(WT2,A1) + b2
A2 = ((np.exp(Z2) - np.exp(-Z2))/(np.exp(Z2) + np.exp(-Z2)))
Z3 = np.dot(WT3,A2) + b3
A3 = 1/(1 + np.exp(-Z3))
dA3 = -Y / A3 + (1 - Y) / (1 - A3)
dZ3 = dA3 * (A3 * (1 - A3))
dWT3 = 1 / m * np.dot(dZ3,A2.T)
db3 = 1 / m * np.sum(dZ3, axis = 1, keepdims = True)
dA2 = np.dot(WT3.T,dZ3)
dZ2 = dA2 * (1-A2**2)
dWT2 = 1 / m * np.dot(dZ2,A1.T)
db2 = 1 / m * np.sum(dZ2, axis = 1, keepdims = True)
dA1 = np.dot(WT2.T,dZ2)
dZ1 = dA1 * (1-A1**2)
dWT1 = 1 / m * np.dot(dZ1,A0.T)
db1 = 1 / m * np.sum(dZ1, axis = 1, keepdims = True)
```

```
def forwardfunc(A0,A1,A2,A3,WT1,WT2,WT3,b1,b2,b3):
    Z1 = np.dot(WT1,A0) + b1
    A1 = ((np.exp(Z1) - np.exp(-Z1))/(np.exp(Z1) + np.exp(-Z1)))
    Z2 = np.dot(WT2,A1) + b2
    A2 = ((np.exp(Z2) - np.exp(-Z2))/(np.exp(Z2) + np.exp(-Z2)))
    Z3 = np.dot(WT3,A2) + b3
    A3 = 1/(1 + np.exp(-Z3))
    return A1,A2,A3
```

```
np.shape(Y): (1, 35) np.shape(A0): (2, 35)
```



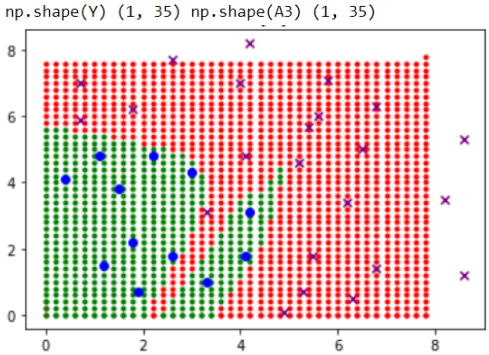

```
def backwardprop(Y,A3,A2,A1,A0,dA2,dA1,dZ3,dZ2,dZ1,dWT3,dWT2,dWT1,db3,db2,db1):
    dA3 = -Y / A3 + (1 - Y) / (1 - A3)
    dZ3 = dA3 * (A3 * (1 - A3))
    dWT3 = 1 / m * np.dot(dZ3,A2.T)
    db3 = 1 / m * np.sum(dZ3, axis = 1, keepdims = True)
    dA2 = np.dot(WT3.T,dZ3)
    dZ2 = dA2 * (1-A2**2)
    dWT2 = 1 / m * np.dot(dZ2,A1.T)
    db2 = 1 / m * np.sum(dZ2, axis = 1, keepdims = True)
    dA1 = np.dot(WT2.T,dZ2)
    dZ1 = dA1 * (1-A1**2)
    dWT1 = 1 / m * np.dot(dZ1,A0.T)
    db1 = 1 / m * np.sum(dZ1, axis = 1, keepdims = True)
    return dA2,dA1,dZ3,dZ2,dZ1,dWT3,dWT2,dWT1,db3,db2,db1
```

```
itera = 0
cost = 100
while(cost > 0.05):
    A1,A2,A3 = forwardfunc(A0,A1,A2,A3,WT1,WT2,WT3,b1,b2,b3)
    cost = -1 / m * np.sum((Y * np.log(A3) + (1 - Y) * np.log(1 - A3)), axis = 1, keepdims = True )
    if(itera % 10000 == 0):
        print("itera ", itera, "cost ", cost)
    dA2,dA1,dZ3,dZ2,dZ1,dWT3,dWT2,dWT1,db3,db2,db1 = backwardprop(Y,A3,A2,A1,A0,dA2,dA1,dZ3,dZ2,dZ1,dWT3,dWT2,dWT1,db3,db2,db1)
    WT3 = WT3 - alpha * dWT3; b3 = b3 - alpha * db3;
    WT2 = WT2 - alpha * dWT2; b2 = b2 - alpha * db2;
    WT1 = WT1 - alpha * dWT1; b1 = b1 - alpha * db1;
    itera = itera + 1
print("np.shape(Y)",np.shape(Y), "np.shape(A3)",np.shape(A3))
```

```
itera  0 cost  [[0.66039906]]
itera  10000 cost  [[0.65969081]]
itera  20000 cost  [[0.396865]]
itera  30000 cost  [[0.25698869]]
itera  40000 cost  [[0.24991327]]
itera  50000 cost  [[0.24713943]]
itera  60000 cost  [[0.2438447]]
itera  70000 cost  [[0.2373221]]
itera  80000 cost  [[0.22550253]]
itera  90000 cost  [[0.21080392]]
itera  100000 cost  [[0.19473781]]
itera  110000 cost  [[0.16078206]]
itera  120000 cost  [[0.10503701]]
itera  130000 cost  [[0.07512948]]
itera  140000 cost  [[0.0595101]]
itera  150000 cost  [[0.05105958]]
```

```
A0 = np.zeros((n0,1600))
for i in range(40):
    for j in range(40):
        A0[0][i * 39 + j] = i * 0.2
        A0[1][i * 39 + j] = j *0.2
A1,A2,A3 = forwardfunc(A0,A1,A2,A3,WT1,WT2,WT3,b1,b2,b3)
print("A3 ",A3)
plt.scatter(A0[0][0:2], A0[1][0:2], c = "red", marker = ".")
plt.scatter(A0[0][2], A0[1][2], c = "green", marker = ".")
for i in range(40):
    for j in range(40):
        if(A3[0][i * 39 + j] > 0.5):
            plt.scatter(A0[0][i * 39 + j],A0[1][i * 39 + j], c = "red", marker = ".");
        else:
            plt.scatter(A0[0][i * 39 + j],A0[1][i * 39 + j], c = "green", marker = ".");
A0 = np.array([x1,x2])
A1,A2,A3 = forwardfunc(A0,A1,A2,A3,WT1,WT2,WT3,b1,b2,b3)
for i in range(m):
    if(A3[0][i] < 0.5):
        plt.scatter(A0[0][i],A0[1][i], c = "blue", marker = "o");
    else:
        plt.scatter(A0[0][i],A0[1][i], c = "purple", marker = "x");
plt.title('xx, xy, y')
plt.show()
```

```
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```



修改
(1)Change alpha
(2)Change initial condition to others make it faster
(3)Change to 1 Layer 2 Layer
(4)Change A0 by sklearn datasets
先複製資料給function後運行function的內容再將有改動的資料回傳給需要改動的資料。這樣就不需要用到global

(1)Change alpha:

Alpha = 0.01 有一點慢會停下來

```
itera  130000 cost  [[0.07512948]]
itera  140000 cost  [[0.0595101]]
itera  150000 cost  [[0.05105958]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

Alpha = 0.05 很快會停下來

```
itera  0 cost  [[0.66039906]]
itera  10000 cost  [[0.24713962]]
itera  20000 cost  [[0.19474574]]
itera  30000 cost  [[0.05341245]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

Alpha = 0.06 很快會停下來

```
itera  0 cost  [[0.66039906]]
itera  10000 cost  [[0.24368556]]
itera  20000 cost  [[0.10353737]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

Alpha = 0.07 很快會停下來

```
itera  0 cost  [[0.66039906]]
itera  10000 cost  [[0.23285686]]
itera  20000 cost  [[0.06273253]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

Alpha = 0.075 很快會停下來

```
itera  0 cost  [[0.66039906]]
itera  10000 cost  [[0.22442392]]
itera  20000 cost  [[0.06281668]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

Alpha = 0.08 很慢會停下來

```
itera  2970000 cost  [[0.14313194]]
itera  2980000 cost  [[0.14313149]]
itera  2990000 cost  [[0.14313103]]
itera  3000000 cost  [[0.14313058]]
itera  3010000 cost  [[0.14313014]]
itera  3020000 cost  [[0.14312969]]
itera  3030000 cost  [[0.14317537]]
itera  3040000 cost  [[0.14313894]]
itera  3050000 cost  [[0.14313799]]
```

(2)Change initial condition to others

WT array 用WT Random number * 0.001 倍 => 較慢會停下來

```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.001;b1 = np.random.rand(n1,1)
WT2 = np.random.randn(n2,n1)*0.001;b2 = np.random.rand(n2,1)
WT3 = np.random.randn(n3,n2)*0.001;b3 = np.random.rand(n3,1)
```

```
itera  170000 cost  [[0.08655227]]
itera  180000 cost  [[0.07618494]]
itera  190000 cost  [[0.06335548]]
itera  200000 cost  [[0.05352847]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

WT array 用WT Random number * 0.01 倍 => 比較快會停下來

```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.01;b1 = np.random.rand(n1,1)
WT2 = np.random.randn(n2,n1)*0.01;b2 = np.random.rand(n2,1)
WT3 = np.random.randn(n3,n2)*0.01;b3 = np.random.rand(n3,1)
```

```
itera  120000 cost  [[0.10503701]]
itera  130000 cost  [[0.07512948]]
itera  140000 cost  [[0.0595101]]
itera  150000 cost  [[0.05105958]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

WT array 用Random number * 0.1 倍 => 更快會停下來

```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.1;b1 = np.random.rand(n1,1)
WT2 = np.random.randn(n2,n1)*0.1;b2 = np.random.rand(n2,1)
WT3 = np.random.randn(n3,n2)*0.1;b3 = np.random.rand(n3,1)
```

```
itera  120000 cost  [[0.1623773]]
itera  130000 cost  [[0.15268263]]
itera  140000 cost  [[0.07785665]]
np.shape(Y) (1, 35) np.shape(A3) (1, 35)
```

b array 用random number * 0.1倍數 => 非常慢停下來

```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.01;b1 = np.random.rand(n1,1)*0.1
WT2 = np.random.randn(n2,n1)*0.01;b2 = np.random.rand(n2,1)*0.1
WT3 = np.random.randn(n3,n2)*0.01;b3 = np.random.rand(n3,1)*0.1
```

```
itera  190000 cost  [[0.0704819]]
itera  200000 cost  [[0.06984221]]
itera  210000 cost  [[0.06931243]]
itera  220000 cost  [[0.06886734]]
itera  230000 cost  [[0.06848881]]
itera  240000 cost  [[0.06816348]]
```

b array 用random number * 0倍數 => 非常慢停下來

```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.01;b1 = np.zeros((n1,1))
WT2 = np.random.randn(n2,n1)*0.01;b2 = np.zeros((n2,1))
WT3 = np.random.randn(n3,n2)*0.01;b3 = np.zeros((n3,1))
```

```
itera  400000 cost  [[0.06584153]]
itera  410000 cost  [[0.06578579]]
itera  420000 cost  [[0.06573366]]
itera  430000 cost  [[0.0656848]]
itera  440000 cost  [[0.06563892]]
```

b array 用random number * 0.01倍數 => 非常慢停下來

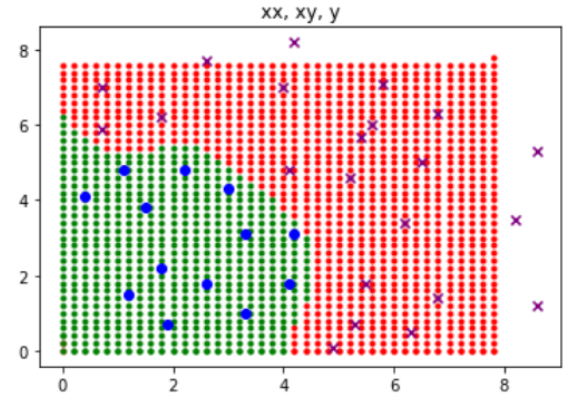
```
np.random.seed(1)
WT1 = np.random.randn(n1,n0)*0.01;b1 = np.random.rand(n1,1)*10
WT2 = np.random.randn(n2,n1)*0.01;b2 = np.random.rand(n2,1)*10
WT3 = np.random.randn(n3,n2)*0.01;b3 = np.random.rand(n3,1)*10
```

```
itera  100000 cost  [[0.37093456]]
itera  110000 cost  [[0.37071357]]
itera  120000 cost  [[0.37052463]]
itera  130000 cost  [[0.37036053]]
```

(3) Reduce layer numbers to two, one layer and redraw the distribution plot

2 layer => 更改成cost > 0.41

```
np.random.seed(1)
WT2 = np.random.randn(n2,n0)*0.1;b2 = np.random.rand(n2,1)
WT3 = np.random.randn(n3,n2)*0.1;b3 = np.random.rand(n3,1)
Z2 = np.dot(WT2,A0) + b2
A2 = ((np.exp(Z2) - np.exp(-Z2))/(np.exp(Z2) + np.exp(-Z2)))
Z3 = np.dot(WT3,A2) + b3
A3 = 1/(1 + np.exp(-Z3))
dA3 = -Y / A3 + (1 - Y) / (1 - A3)
dZ3 = dA3 * (A3 * (1 - A3))
dWT3 = 1 / m * np.dot(dZ3,A2.T)
db3 = 1 / m * np.sum(dZ3, axis = 1, keepdims = True)
dA2 = np.dot(WT3.T,dZ3)
dZ2 = dA2 * (1-A2**2)
dWT2 = 1 / m * np.dot(dZ2,A0.T)
db2 = 1 / m * np.sum(dZ2, axis = 1, keepdims = True)
```




```
itera = 0
cost = 100
while(cost > 0.2):
    A2,A3 = forwardfunc(A0,A2,A3,WT2,WT3,b2,b3)
    cost = -1 / m * np.sum((Y * np.log(A3) + (1 - Y) * np.log(1 - A3)), axis = 1, keepdims = True )
    if(itera % 10000 == 0):
        print("itera ", itera, "cost ", cost)
    dA2,dZ3,dZ2,dWT3,dWT2,db3,db2 = backwardprop(Y,A3,A2,A0,dA2,dZ3,dZ2,dWT3,dWT2,db3,db2)
    WT3 = WT3 - alpha * dWT3; b3 = b3 - alpha * db3;
    WT2 = WT2 - alpha * dWT2; b2 = b2 - alpha * db2;
    itera = itera + 1
print("np.shape(Y)",np.shape(Y), "np.shape(A3)",np.shape(A3))
```

```
def forwardfunc(A0,A2,A3,WT2,WT3,b2,b3):
    Z2 = np.dot(WT2,A0) + b2
    A2 = ((np.exp(Z2) - np.exp(-Z2))/(np.exp(Z2) + np.exp(-Z2)))
    Z3 = np.dot(WT3,A2) + b3
    A3 = 1/(1 + np.exp(-Z3))
    return A2,A3
```

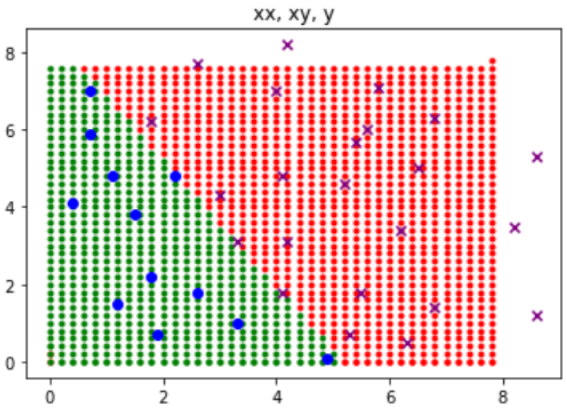
```
def backwardprop(Y,A3,A2,A0,dA2,dZ3,dZ2,dWT3,dWT2,db3,db2):
    dA3 = -Y / A3 + (1 - Y) / (1 - A3)
    dZ3 = dA3 * (A3 * (1 - A3))
    dWT3 = 1 / m * np.dot(dZ3,A2.T)
    db3 = 1 / m * np.sum(dZ3, axis = 1, keepdims = True)
    dA2 = np.dot(WT3.T,dZ3)
    dZ2 = dA2 * (1-A2**2)
    dWT2 = 1 / m * np.dot(dZ2,A1.T)
    db2 = 1 / m * np.sum(dZ2, axis = 1, keepdims = True)
    return dA2,dZ3,dZ2,dWT3,dWT2,db3,db2
```

1layer => 更改成cost > 0.2

```
np.random.seed(1)
WT3 = np.random.randn(n3,n0)*0.1;b3 = np.random.rand(n3,1)
Z3 = np.dot(WT3,A0) + b3
A3 = 1/(1 + np.exp(-Z3))
dA3 = -Y / A3 + (1 - Y) / (1 - A3)
dZ3 = dA3 * (A3 * (1 - A3))
dWT3 = 1 / m * np.dot(dZ3,A0.T)
db3 = 1 / m * np.sum(dZ3, axis = 1, keepdims = True)
```

```
def forwardfunc(A0,A2,A3,WT2,WT3,b2,b3):
    Z2 = np.dot(WT2,A0) + b2
    A2 = ((np.exp(Z2) - np.exp(-Z2))/(np.exp(Z2) + np.exp(-Z2)))
    Z3 = np.dot(WT3,A2) + b3
    A3 = 1/(1 + np.exp(-Z3))
    return A2,A3
```

```
def backwardprop(Y,A3,A0,dZ3,dWT3,db3):
    dA3 = -Y / A3 + (1 - Y) / (1 - A3)
    dZ3 = dA3 * (A3 * (1 - A3))
    dWT3 = 1 / m * np.dot(dZ3,A0.T)
    db3 = 1 / m * np.sum(dZ3, axis = 1, keepdims = True)
```




```
itera = 0
cost = 100
while(cost > 0.41):
    A3 = forwardfunc(A0,A3,WT3,b3)
    cost = -1 / m * np.sum((Y * np.log(A3) + (1 - Y) * np.log(1 - A3)), axis = 1, keepdims = True )
    if(itera % 10000 == 0):
        print("itera ", itera, "cost ", cost)
    dZ3,dWT3,db3 = backwardprop(Y,A3,A0,dZ3,dWT3,db3)
    WT3 = WT3 - alpha * dWT3; b3 = b3 - alpha * db3;
    WT2 = WT2 - alpha * dWT2; b2 = b2 - alpha * db2;
    itera = itera + 1
print("np.shape(Y)",np.shape(Y), "np.shape(A3)",np.shape(A3))
```

(4) Change A0 by sklearn datasets:

```
import matplotlib
import time
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import sklearn.datasets


n0 = 2; n1 = 3; n2 = 4; n3 = 1; m = 35;
alpha = 0.05
# load image dataset: blue/red dots in circles
train_X, train_Y = sklearn.datasets.make_circles(n_samples=35, noise=0.5)
A0=train_X.T; Y=train_Y.reshape(1,35)
print("np.shape(Y):",np.shape(Y),"np.shape(A0):",np.shape(A0))
for i in range(35):
    if(Y[0][i]<0.5):
        plt.scatter(A0[0][i], A0[1][i],c="blue",marker="o")
    else:
        plt.scatter(A0[0][i], A0[1][i],c="purple",marker="x")
plt.title('x1, x2, y')
plt.show()
```

```
np.shape(Y): (1, 35) np.shape(A0): (2, 35)
```