Problem 3

L2 and Sigmoid

```
[ 1.75925765  0.42222715  0.97302781  2.23545643  1.76205194 -1.04551143]
[ 0.92301026 -0.13231169 -0.22835087]
[0. 0. 0. 0.]

[ 1.75639473  0.44768815  0.96365942  2.2279127   1.68543601 -1.08787   ]
[ 0.90364831 -0.11697428 -0.32024887]
[0. 0. 0. 0.]

[ 1.75517014  0.47429405  0.95128588  2.21913511  1.63252109 -1.11038935]
[ 0.88990237 -0.10558866 -0.38504499]
[0. 0. 0. 0.]
```

L2 and ReLU

```
[ 1.76405235  0.40015721  0.97873798  2.2408932   0.27984019 -0.97727788]
[-2.17083698 -0.15135721 -1.77434522]
[0. 0. 0. 0.]

[ 1.76405235  0.40015721  0.97873798  2.2408932   0.27984019 -0.97727788]
[-2.17083698 -0.15135721 -1.77434522]
[0. 0. 0. 0.]

[ 1.76405235  0.40015721  0.97873798  2.2408932   0.27984019 -0.97727788]
[-2.17083698 -0.15135721 -1.77434522]
[0. 0. 0. 0.]
```

L2 and tanh

```
[ 1.76394099  0.79543567  0.97875095  2.21120739  2.32649648 -0.24566039]
[ 1.1897533  -0.00522277  0.43245803]
[0.89068748 0.22016163 0.98952119 0.98855594]

[ 1.76379641  0.79538701  0.97876066  2.21124784  2.27040644 -0.27332827]
[1.16515396 0.00313831 0.37059281]
[0.98253276 0.98259367 0.98661795 0.98527789]

[ 1.76359679  0.79537839  0.97875443  2.21130349  2.19714684 -0.31096816]
[1.13322901 0.01501468 0.28961954]
[0.97748175 0.97584019 0.98174523 0.97975951]
```

Cross Entropy and Sigmoid

```
[ 1.72762891  0.52267639  0.9677286   2.1793897   1.29910391 -1.29521033]
[ 0.84057746 -0.07503757 -0.69900555]
[0. 0. 0. 0.]

[ 1.7086309   0.61782198  0.94135492  2.09095611  1.17997518 -1.27637024]
[ 0.86057705 -0.08176463 -0.75814116]
[0. 0. 0. 0.]

[ 1.69474177  0.69449611  0.91493861  1.99587439  1.1188301  -1.22849317]
[ 0.88027882 -0.10766722 -0.76325045]
[0. 0. 0. 0.]
```

Cross Entropy and ReLU

```
[ 2.08984213  0.81315914  0.90891115  2.13319208  4.14635659 -0.66406269]
[ 4.31287148 -0.29685722  1.55164802]
[0. 0. 0. 0.]

[ 2.35708971  1.12331108  0.87601159  2.09038689  5.02567952 -0.49947899]
[ 4.98889586 -0.3576289   1.70471414]
[0. 0. 0. 0.]

[ 2.59147289  1.38683699  0.85709534  2.0670795   5.7643893  -0.3865542 ]
[ 5.57267489 -0.39132415  1.81580965]
[0. 0. 0. 0.]
```

Cross Entropy and tanh

```
[ 1.75636844  0.39228937  0.97928936  2.24091301  0.42706334 -0.76215214]
[-0.66230524  1.65508155 -2.06298887]
[0. 0. 0. 0.]

[ 1.75115703  0.38691316  0.97971534  2.24092925  0.40031521 -0.8164279 ]
[-0.67326696  1.65601464 -2.11788433]
[0. 0. 0. 0.]

[ 1.74725016  0.38285109  0.98006668  2.24094321  0.37987118 -0.85941186]
[-0.68151514  1.65681619 -2.16136673]
[0. 0. 0. 0.]
```

Code

```
#%% Load Packages
import numpy as np
#%% Activation and Derivatives
def sig(a):
    return 1/(1+np.exp(-a))

def dsig(a):
    return sig(a)*(1-sig(a))

def relu(a):
    return max(0,a)

def drelu(a):
    if a > 0:
        return 1
    else:
        return 0

#def tanh(a):
#    return (np.exp(2*a)-1)/(np.exp(2*a)+1)

def dtanh(a):
    return 1-(np.tanh(a))**2
#%% Loss Function
def CE(in_, out_):
    if in_ == 1:
        return -in_*np.log(out_)
#    elif (in_ == 0 & out_ == 0):
```

```
#         return 0
    else:
        return -(1-in_)*np.log(1-out_)

def L2(in_, out_):
    return 0.5*(in_-out_)**2

def dCE(in_, out_):
    return (-out_/(in_+1e-5))+(1-out_)/(1-in_+1e-5)
#%% Forward Propagation
def SGD(X,Y,W,B,LR):
    yout = np.zeros(x.shape[0])
    for i in np.arange(x.shape[0]):
        H1 = X[i,0]*W[0] + X[i,1]*W[1] + B[0]
        Z1 = np.tanh(H1)
        H2 = X[i,0]*W[2] + X[i,1]*W[3] + B[1]
        Z2 = np.tanh(H2)
        H3 = Z1*W[4] + Z2*W[5] + B[2]
        Y_hat = np.tanh(H3)

        dLdW11 = X[i,0]*dtanh(H1)*W[4]*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdW12 = X[i,0]*dtanh(H2)*W[5]*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdW21 = X[i,1]*dtanh(H1)*W[4]*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdW22 = X[i,1]*dtanh(H2)*W[5]*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdW31 = Z1*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdW32 = Z2*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdB1 = dtanh(H1)*W[4]*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdB2 = dtanh(H2)*W[5]*dtanh(H3)*dCE(Y_hat,Y[i])
        dLdB3 = dtanh(H3)*dCE(Y_hat,Y[i])
        W[0] = W[0] - dLdW11
```

```
        W[2] = W[2] - dLdW12
        W[1] = W[1] - dLdW21
        W[3] = W[3] - dLdW22
        W[4] = W[4] - dLdW31
        W[5] = W[5] - dLdW32
        B[0] = B[0] - dLdB1
        B[1] = B[1] - dLdB2
        B[2] = B[2] - dLdB3
    return W,B,yout
def rinit():
    np.random.seed(seed=0)
    return np.random.randn(6), np.random.randn(3)
def RUN(X,Y,W,B,LR,E):
    for i in np.arange(E):
        W,B,Y_hat = SGD(X,Y,W,B,LR)
        print(W)
        print(B)
        print(Y_hat)
    return W,B,Y_hat
#%%
x = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0],[1],[1],[0]])

w, b = rinit()

learning_rate = 0.1
epochs = 3
t1, t2, yhat = RUN(x,y, w, b, learning_rate, epochs)
#%
```