# exam2

October 24, 2023

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     import sympy as sy
     from IPython.display import display, Math, Latex
```

## 1 Problem 1:

```
[ ]: def sigmoid(x: float, deriv: bool = False):
         if deriv:
             vals = sigmoid(x, deriv=False)*(1-sigmoid(x, deriv=False))
             return np.array([np.diag(val) for val in vals])
         else:
             return 1/(1 + np.exp(-x))
```

```
[ ]: W1 = np.ones((4,3))
     X = np.array([1, 2, 3, 4])
     X@W1
```

```
[ ]: array([10., 10., 10.])
```

```
[ ]: H = np.array([2, 2, 2])
     W = np.ones([3, 1])
     H@W
     sigmoid(H@W)
```

```
[ ]: array([0.99752738])
```

```
[ ]: -np.log(0.9)
```

```
[ ]: 0.10536051565782628
```

## 2 Problem 2:

### 2.1 Part 1-2:

```
[ ]: X = np.array([[1, 2, 3, 0],
                   [2, 4, 1, 0],
                   [4, 3, 2, 0]])
     Y = np.array([[0],
                   [1],
                   [1]])
     W1 = np.ones((4, 3))
     W2 = np.ones((3, 1))
     B = np.zeros((1, 3))
     C = np.zeros((1, 1))
     Math("W1 = " + sy.latex(sy.Matrix(W1)) +\
          "\quad W2 = " + sy.latex(sy.Matrix(W2)))
```

[ ]:

$$W1 = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix} \quad W2 = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

### 2.2 Part 3-6:

```
[ ]: Z1 = X@W1 + B
     H1 = sigmoid(Z1)
     Z2 = H1@W2 + C
     yhat = sigmoid(Z2)
     display(Math("Z1 = " + sy.latex(sy.Matrix(np.round(Z1,2)))))
     display(Math("H1 = " + sy.latex(sy.Matrix(np.round(H1,2)))))
     display(Math("Z2 = " + sy.latex(sy.Matrix(np.round(Z2,2)))))
     display(Math("yhat = " + sy.latex(sy.Matrix(np.round(yhat,2)))))
```

$$Z1 = \begin{bmatrix} 6.0 & 6.0 & 6.0 \\ 7.0 & 7.0 & 7.0 \\ 9.0 & 9.0 & 9.0 \end{bmatrix}$$

$$H1 = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

$$Z2 = \begin{bmatrix} 2.99 \\ 3.0 \\ 3.0 \end{bmatrix}$$

$$yhat = \begin{bmatrix} 0.95 \\ 0.95 \\ 0.95 \end{bmatrix}$$

### 2.3 Part 7-8:

```
[ ]: def loss_bce(yhat, Y):
         eps = 1e-10
         return -Y*np.log(yhat + eps) - (1-Y)*np.log(1-yhat+eps)
     loss = loss_bce(yhat, Y)
     Math("Loss = " + sy.latex(sy.Matrix(np.round(loss,2))))
```

[ ]:

$$Loss = \begin{bmatrix} 3.04 \\ 0.05 \\ 0.05 \end{bmatrix}$$

```
[ ]: dL_dz1 = (Y-yhat)*W2.T@H*(1-H)
     dL_dW1 = [np.round(np.outer(xi,dL_dz1),3) for xi in X]
     print("for all X_i:")
     [display(Math("dL/DW1 = " + sy.latex(sy.Matrix(np.round(mat,2))))) for mat in␣
      ↪dL_dW1]
     print("averaged over X_i:")
     Math("dL/DW1 = " + sy.latex(sy.Matrix(np.round(np.mean(dL_dW1,axis=0),3))))
```

```
for all X_i:
```

$$dL/DW1 = \begin{bmatrix} 5.71 & -0.28 & -0.28 \\ 11.43 & -0.57 & -0.57 \\ 17.14 & -0.86 & -0.85 \\ 0 & 0 & 0 \end{bmatrix}$$

$$dL/DW1 = \begin{bmatrix} 11.43 & -0.57 & -0.57 \\ 22.85 & -1.14 & -1.14 \\ 5.71 & -0.28 & -0.28 \\ 0 & 0 & 0 \end{bmatrix}$$

$$dL/DW1 = \begin{bmatrix} 22.85 & -1.14 & -1.14 \\ 17.14 & -0.86 & -0.85 \\ 11.43 & -0.57 & -0.57 \\ 0 & 0 & 0 \end{bmatrix}$$

```
averaged over X_i:
```

[ ]:

$$dL/DW1 = \begin{bmatrix} 13.331 & -0.666 & -0.664 \\ 17.14 & -0.856 & -0.854 \\ 11.427 & -0.571 & -0.569 \\ 0 & 0 & 0 \end{bmatrix}$$

## 3 Problem 3:

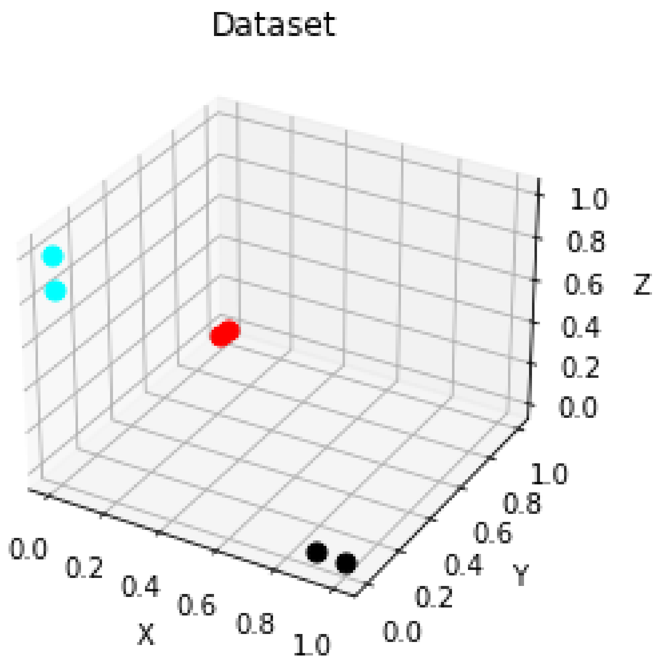
Define the Data to be Used. Make it a simple clustering problem in 3 dimensions with 2 in each class:

```
[ ]: X = np.array([[1, 0, 0],
                   [0.9, 0, 0],
                   [0, 1, 0],
                   [0, 0.95, 0],
                   [0, 0, 1],
                   [0, 0, 0.85]])
     Y = np.array([[1, 0, 0],
                   [1, 0, 0],
                   [0, 1, 0],
                   [0, 1, 0],
                   [0, 0, 1],
                   [0, 0, 1]])
```

Plot to verify that the data points are well separated.

[ ]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')

# plotting
colors = ["k", "r", "cyan"]
for i in range(Y.shape[0]):
    ax.scatter(X[i,0], X[i,1], X[i,2],
               s=50, c=colors[np.argmax(Y[i])])
ax.set_title('Dataset')
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.show()
```



To make the Neural Network, I will use the general code that I developed on the homework assignment. I'll attach the source file.

```
[ ]: import nn
     network = nn.NeuralNetwork(input_size=3, output_size=3, hidden_layer_sizes=[2],
                                activ_funcs=[nn.sigmoid, nn.softmax], loss_func=nn.
      ↪loss_CCE,
                                random_initialize=True)
```

Initial Values of stuff:

```
[ ]: display(Math("X = " + sy.latex(sy.Matrix(np.round(X,2)))))
     display(Math("Y = " + sy.latex(sy.Matrix(np.round(Y,2)))))
     display(Math("W1 = " + sy.latex(sy.Matrix(np.round(network.weights[0],4)))))
     display(Math("W2 = " + sy.latex(sy.Matrix(np.round(network.weights[1],2)))))
     display(Math("B = " + sy.latex(sy.Matrix(np.round(network.biases[0],2)))))
     display(Math("C = " + sy.latex(sy.Matrix(np.round(network.biases[1],2)))))
```

$$X = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.9 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0.95 & 0 \\ 0 & 0 & 1.0 \\ 0 & 0 & 0.85 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$W1 = \begin{bmatrix} 0.5938 & 0.0634 \\ 0.0018 & 0.1043 \\ 0.3504 & 0.0656 \end{bmatrix}$$

$$W2 = \begin{bmatrix} 1.0 & 0.41 & 0.26 \\ 0.24 & 0.53 & 0.71 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.76 & 0.76 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.95 & 0.21 & 0.98 \end{bmatrix}$$

Now train the network for one epoch:

```
[ ]: loss = network.train(X, Y, X_test=X, Y_test=Y, epochs=1, batch_size=6, lr=1)
     pred = network.feed_forward(X)
```

```
Epoch 0 (out of 1) -- Loss: 1.5997
```

Print out desired information:

```
[ ]: display(Math("Z1 = " + sy.latex(sy.Matrix(np.round(network.layers[0],2)))))
     display(Math("H1 = " + sy.latex(sy.Matrix(np.round(network.activ_fns[0](network.
      ↪layers[0]),2)))))
     display(Math("Z2 = " + sy.latex(sy.Matrix(np.round(network.layers[1],2)))))
     display(Math("\hat{Y} = " + sy.latex(sy.Matrix(np.round(pred,2)))))
     display(Math("\hat{Y}-Y = " + sy.latex(sy.Matrix(np.round(pred - Y,2)))))


     display(Math(r"\frac{dL}{dW1} (avg) = " + sy.latex(sy.Matrix(np.round(network.
      ↪dLdw[0],4)))))
     display(Math(r"\frac{dL}{dW2} (avg) = " + sy.latex(sy.Matrix(np.round(network.
      ↪dLdw[1],2)))))
     display(Math(r"\frac{dL}{dB} (avg) = " + sy.latex(sy.Matrix(np.round(network.
      ↪dLdb[0],4)))))
     display(Math(r"\frac{dL}{dC} (avg) = " + sy.latex(sy.Matrix(np.round(network.
      ↪dLdb[1],2)))))

     display(Math("L_{avg} = " + str(np.sum(network.loss_fn(Y, pred)))))

     display(Math("W1_{new} = " + sy.latex(sy.Matrix(np.round(network.
      ↪weights[0],4)))))
```

$$Z1 = \begin{bmatrix} 1.34 & 0.83 \\ 1.28 & 0.82 \\ 0.75 & 0.87 \\ 0.75 & 0.87 \\ 1.1 & 0.83 \\ 1.04 & 0.82 \end{bmatrix}$$

$$H1 = \begin{bmatrix} 0.79 & 0.7 \\ 0.78 & 0.69 \\ 0.68 & 0.7 \\ 0.68 & 0.7 \\ 0.75 & 0.7 \\ 0.74 & 0.69 \end{bmatrix}$$

$$Z2 = \begin{bmatrix} 1.65 & 1.23 & 1.6 \\ 1.65 & 1.23 & 1.6 \\ 1.55 & 1.18 & 1.58 \\ 1.55 & 1.18 & 1.58 \\ 1.62 & 1.21 & 1.59 \\ 1.61 & 1.21 & 1.59 \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} 0.38 & 0.25 & 0.36 \\ 0.38 & 0.25 & 0.37 \\ 0.37 & 0.25 & 0.38 \\ 0.37 & 0.25 & 0.38 \\ 0.38 & 0.25 & 0.37 \\ 0.38 & 0.25 & 0.37 \end{bmatrix}$$

$$\hat{Y} - Y = \begin{bmatrix} -0.62 & 0.25 & 0.36 \\ -0.62 & 0.25 & 0.37 \\ 0.37 & -0.75 & 0.38 \\ 0.37 & -0.75 & 0.38 \\ 0.38 & 0.25 & -0.63 \\ 0.38 & 0.25 & -0.63 \end{bmatrix}$$

$$\frac{dL}{dW1}(avg) = \begin{bmatrix} 0.0041 & -0.0019 \\ 0.0042 & -0.002 \\ 0.0039 & -0.0019 \end{bmatrix}$$

$$\frac{dL}{dW2}(avg) = \begin{bmatrix} 0.09 & -0.12 & 0.03 \\ 0.09 & -0.11 & 0.02 \end{bmatrix}$$

$$\frac{dL}{dB}(avg) = \begin{bmatrix} 0.0128 \\ -0.0062 \end{bmatrix}$$

$$\frac{dL}{dC}(avg) = \begin{bmatrix} 0.12 \\ -0.16 \\ 0.04 \end{bmatrix}$$

$$L_{avg} = 9.59841928248252$$

$$W1_{new} = \begin{bmatrix} 0.5898 & 0.0653 \\ -0.0024 & 0.1063 \\ 0.3465 & 0.0675 \end{bmatrix}$$