Name:

Student Number:

Graduate students must answer the bonus question.

1. (10 points) Given an input vector $\mathbf{x} = [x_0, x_1, x_2]^T$, and a function $f : \mathbb{R}^3 \to \mathbb{R}^2$ of the following form:

$$f(\mathbf{x}) = \begin{bmatrix} \sum_i x_i \sin(x_{(i+2)\%3}) \\ \sum_i \sin(x_i) \cos(x_{(i+1)\%3}) \end{bmatrix} \qquad \forall i \in \{0, 1, 2\} \tag{1}$$

write the Jacobian of $f(\mathbf{x})$. $\sum$ is the sum operator and % is the modulus operator.

2. (20 points) Suppose you have an input feature vector in $\mathbb{R}^3$ as $\mathbf{x} = [2, -5, 1]^T$ and a trained 2-layer linear classifier with the following weights and biases:

$$\mathbf{W_1} = \begin{bmatrix} 1 & 2 & 3 \\ -4 & -2 & 1 \\ 1 & -2 & -3 \end{bmatrix}, \mathbf{b_1} = \begin{bmatrix} -2 \\ -1 \\ 3 \end{bmatrix} \quad \text{and} \quad \mathbf{W_2} = \begin{bmatrix} -2 & -3 & -1 \\ 3 & 1 & 2 \end{bmatrix}, \mathbf{b_2} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \tag{2}$$

   (a) (2 points) Write the full equation of this classifier.

   (b) (6 points) Calculate the output of this classifier.

   (c) (2 points) Which class does this classifier label the input as? Why?

   (d) (4 points) Assuming the first class was the correct class for the feature vector $\mathbf{x}$, calculate the hinge loss for this classifier.

   (e) (6 points) Assuming the first class was the correct class for the feature vector $\mathbf{x}$, calculate the softmax loss of this classifier.

3. (30 points) In the classifier from question 2, an activation layer with the function $f(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$ is used in each of the layers of the classifier after the linear combination and bias additions. Answer the following questions:
   (a) (5 points) Draw the computational graph of the network. (Hint: Use the activation function as its own block, and use vector format.)
   (b) (5 points) Perform the forward pass of the graph with the input vector of $\mathbf{x} = [2, 3, 4]^T$. (Hint: Write down the local values on each node.)
   (c) (20 points) Calculate the $\nabla_{\mathbf{w}} L$ through the backward pass of the backprop procedure. (For full credit, show all your work.)

4. (40 points) **The XOR Problem**: As we discussed in class, the linear classification with one layer fails to solve the problem of classifying objects that represent a population whose probabilities resemble the XOR function. That is, in a two dimensional feature space, one population occupies the first and third quadrants, while the other population occupies the second and fourth quadrants - see Figure 1.

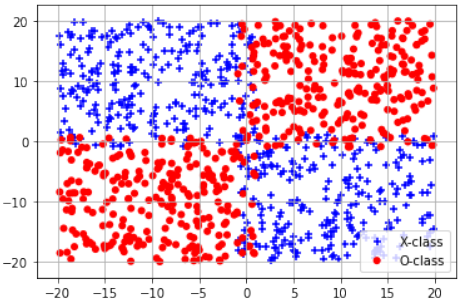

Figure 1: Two classes in the XOR problem

In this problem, you will implement a 2-layer neural network to address the problem of classifying classes who represent the XOR property. In order to accomplish this task you will write a program (either on google collab or a jupyter notebook) to perform the following:

1. **Data Generation:** You will need to create a data set of samples in 2-D belonging to the x-class and the o-class.
2. **Data Visualization:** You will plot the data in a 2D scatter plot shown in Figure 1.
3. **Network Setup:** You will set up your network with two layers. You will use Keras for the purpose of establishing, training, and evaluating your models.
4. **Model Training:** You will train the network using a predefined loss function, an optimization algorithm, etc.
5. **Model Accuracy:** You will generate another dataset of samples and labels for testing purposed. You will then evaluate the trained model on this test dataset and present and discuss your results.

(a) (5 points) **Data Generation:** Use the base code below to create the two input o-class and x-class with their associated labels.

```
# set up number of training samples
N = 250# how many samples

#set up upper and lower bound on your input probabilities
Uh= 20 # upper limit
Ul= -1 # lower limit

# set up the O-class (first and third quadrant)
O = '...' # the o-class contains N uniformly randomly
            # distributed samples whose x and y are both (-Ul, Uh) or (Ul,-Uh) first and third quadrants

# set up the X-class (second and fourth quadrant)
X = '..'  # the x-class contains N uniformly randomly
            # distributed samples whose x and y are in the second and fourth quadrdant



#plot the X-class and the O-class
plt.scatter(X[:,0],X[:,1],marker='+',c='blue',  label='X-class')
plt.scatter(O[:,0],O[:,1],marker='o',c='red', edgecolors='none', label='O-class')
plt.legend()
plt.grid(True)
```

(b) (5 points) **Training Dataset Establishment:** Create a training dataset comprising of x_train and y_train elements.

```
#set up the training samples from the X-class and the O-class
x_train = # compile the x- and o- classes into the x_train array

# set up the labels [1 0] for each x_class and [0 1] for each o-class
y_x= '..'
y_o= '..'

# set up the trraining labels
y_train = # compile the labels into the y_train array for class labels
```

(c) (5 points) **Model Setup:**

```
from keras.models import Sequential
# Import `Dense` from `keras.layers`
from keras.layers import Dense


# set up your model as sequential
model = '...'

# add a dense layer with 8 nodes and relu activations. Note: the input is two-dimensional
model.add('...')

# add another dense layer as the output layer with sigmoid activations. Note: output will be two-dimensional
model.add('...')

# print the shape, summary, configuration, and initial weights of your network
```

(d) (5 points) **Model Training:** Determine a loss function, optimizer, etc. and train your network. Include the training process of your network in the report.

```
# setup the loss as binary cross entropy, optimizer as adam, and metrics as accuracy for your model and compile
model.compile(loss='..',
              optimizer='..',
              metrics=['..'])

# train your model using x_train, y_train, and for a certain number of epochs, with a batch_size of n, and set verbose to 1
model.fit('...')
```

(e) (5 points) **Model Evaluation:** Create a test set with its associated labels and evaluate the network on this set. Show the loss and accuracy values of the network on this test set.

```
# set up 300 samples as test (150 for the x-class and 150 for the o-class)
# create the labels for these items as well [1 0] for each x-class and [0 1] for each o-class

x_test = '...'
y_test = '...'

# evaluate your model
score = model.evaluate('...')

print(score)
```

(f) (8 points) **Code**: Submit your code as a python 3 file.

(g) (7 points) **Report:** Write a report on the results of your network. Conduct some experimentation on various parameters and show how the results change.

(h) (10 points (bonus)) **Grad Only [EC for Undergrad]** Plot the decision boundaries of the model on the test data. Also label the false negative and false positive test samples with a complementary color.