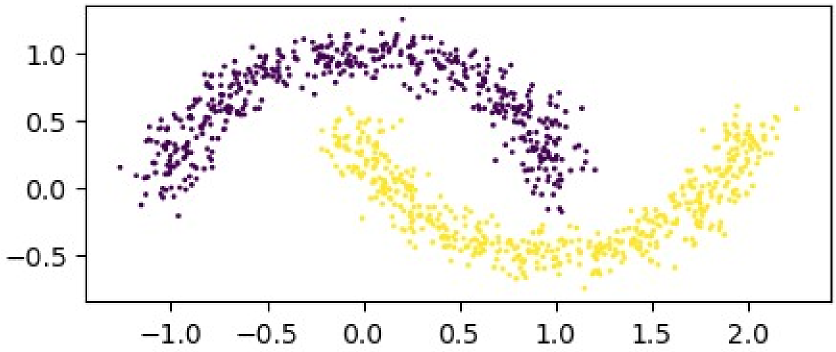
Assignment 02

For this task, you will be using the baseline code for ANN's provided in the class jupyter notebook. **Do not** use any ML libraries.

Task 0: Getting Ready with your Dataset

The dataset we will be using is two-moons.csv, which represents two inverted crescent moons as a toy dataset widely used to visualize clusters and classification algorithms. The columns of the dataset are simply x and y and the labels, which when plotted gives:

```
points = np.genfromtxt('data/cluster_moons.csv', delimiter=',')
points = points[1:len(points)]

x      = points[:,0]
y      = points[:,1]
labels = points[:,2]

plt.figure(figsize=(5,2))
plt.scatter(x, y, c=labels, s=1)
plt.show()
```



To curate the inputs and outputs, place them in separate variables of their own.

```
inputs  = np.column_stack((x, y))
outputs = labels.reshape(-1, 1)
```

Then answer the following questions:

	Question	Answer
1	How many features are in the inputs variable, and how much is its length?	
2	What many output features are in the outputs variable, and how much is its length?	
3	How many points belong to cluster 0, and how many points belong to cluster 1?	

Task 1: 0 Hidden Layer ANN Architecture

From the class jupyter notebook, take the very first ANN architecture of **two** inputs and **two** outputs. This is because in that example, both the input and output had two-two features. The code is reproduced here for brevity and quickness:

```
weights_input_output  = 2 * np.random.random((2,2)) - 1
bias_output = np.random.randn(1, 2)

cost_graph = []

for epoch in range(10000):

    output_layer = sigmoid(np.dot(inputs, weights_input_output) +
                                                bias_output)

    output_error = outputs - output_layer

    cost_graph.append(np.mean(np.square(output_error)))

    output_delta = output_error * sigmoid_derivative(output_layer)

    weights_input_output += inputs.T.dot(output_delta)

    bias_output          += np.sum(output_delta, axis=0, keepdims=True)
```

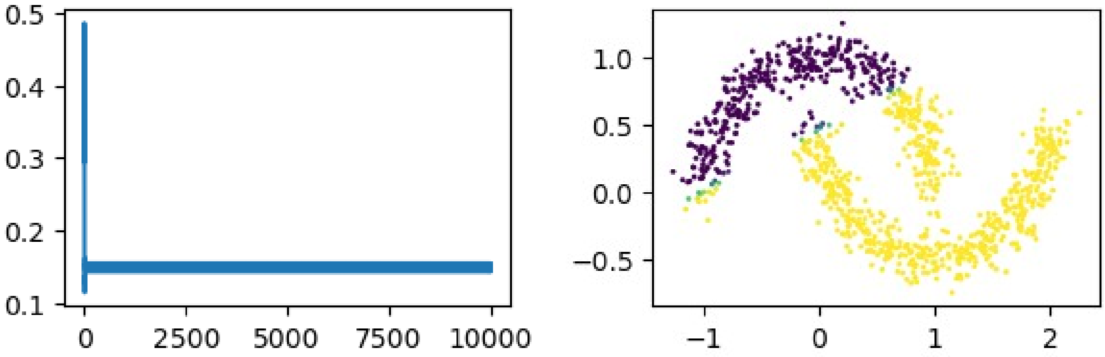
Since in our case, we have two input and one output feature, the code will be modified to:

```
weights_input_output  = 2 * np.random.random((2,1)) - 1
bias_output = np.random.randn(1, 1)
```

Insert at the end of the for loop the following:

```
plt.figure(figsize=(3, 2))
plt.plot(cost_graph)

plt.figure(figsize=(3, 2))
plt.scatter(x, y, c=output_layer, s=1)

plt.show()
```

Which should display for you something like below:



Answer the following questions:

	Question	Answer
1	Change from the sigmoid activation to tanh. What is the effect on the scatter plot? Note: You would need to change the sigmoid derivative as well.	
2	Change from the tanh activation to relu. What is the effect on the scatter plot?	
3	Change from the relu activation to softmax. What is the effect on the scatter plot?	
4	Switch back to sigmoid activation. We will now introduce a learning rate variable as: `lr = 1` which will be multiplied to all the portions involving gradient descent.	

```
weights_input_output += lr * inputs.T.dot(output_delta)

bias_output          += lr * np.sum(output_delta, axis=0,
                                               keepdims=True)
```

What is the effect of changing the learning rate to 0.1, 0.01, 0.001, 0.001?
a) Draw the scatter plot for each of the learning rate
b) Report the minimum error loss reported from cost_graph for each of the learning rates

Task 2: 1 Hidden Layer ANN Architecture

We will use the following code as an extension of the previous architecture (already provided in the class jupyter notebook.

```
weights_input_hidden  = 2 * np.random.random((2, 2)) - 1
weights_hidden_output = 2 * np.random.random((2, 1)) - 1

bias_hidden = np.random.randn(1, 2)
bias_output = np.random.randn(1, 1)

cost_graph = []
lr = 0.01

for epoch in range(10000):

    hidden_layer = sigmoid(np.dot(inputs, weights_input_hidden) +
                                                   bias_hidden)

    output_layer = sigmoid(np.dot(hidden_layer, weights_hidden_output) +
                                                   bias_output)

    output_error = outputs - output_layer
    cost_graph.append(np.mean(np.square(output_error)))

    output_delta = output_error * sigmoid_derivative(output_layer)
    hidden_delta = output_delta.dot(weights_hidden_output.T) *
                              sigmoid_derivative(hidden_layer)

    weights_hidden_output += lr * hidden_layer.T.dot(output_delta)
    weights_input_hidden += lr * inputs.T.dot(hidden_delta)

    bias_output += lr * np.sum(output_delta, axis=0, keepdims=True)
    bias_hidden += lr * np.sum(hidden_delta, axis=0, keepdims=True)

plt.figure(figsize=(3, 2))
plt.plot(cost_graph)

plt.figure(figsize=(3, 2))
plt.scatter(x, y, c=output_layer, s=1)

plt.show()
```

Then answer the following questions:

	Question	Answer
1	Explain why we have the combination of 2, 2 and 2, 1 for the weight matrices, and 1, 2, and 1, 1 for the bias?	
2	Draw the network diagram of the ANN architecture, inclusive of bias.	

3 Is the current result of Task 2 better than Task 1? Answer on the basis of minimum error (cost_graph).

4 Modify the network structure to the following:

```
weights_input_hidden  = 2 * np.random.random((2, 4)) - 1
weights_hidden_output = 2 * np.random.random((4, 1)) - 1

bias_hidden = np.random.randn(1, 4)
bias_output = np.random.randn(1, 1)
```

and report whether the Task 2 with larger ANN network is better than Task 1?

5 Draw the network diagram of ANN architecture for the above larger size.

6 Report the minimum error (cost_graph) for the following table:

Hidden Layer Size

		2	8	16	20	24
Learning Rate	1					
	0.1					
	0.01					
	0.001					

7 From the table in 6 above, what is the minimum error reported? Draw the scatter graph for this size.

Task 3: 2 Hidden Layers ANN Architecture

You should have found the best result so far from Task 2. We will now extend the code to more hidden layers, with the objective that the convergence of error (or learning rate should be faster than Task 2). The code is as follows:

```
weights_input_hidden1   = 2 * np.random.random((2, 2)) - 1
weights_hidden1_hidden2 = 2 * np.random.random((2, 2)) - 1
weights_hidden2_output  = 2 * np.random.random((2, 1)) - 1

bias_hidden1 = np.random.randn(1, 2)
bias_hidden2 = np.random.randn(1, 2)
bias_output = np.random.randn(1, 1)

cost_graph = []
lr = 0.001

for epoch in range(5000):

    hidden_layer1 = sigmoid(np.dot(inputs, weights_input_hidden1)
                              + bias_hidden1)
    hidden_layer2 = sigmoid(np.dot(hidden_layer1, weights_hidden1_hidden2)
                              + bias_hidden2)
    output_layer  = sigmoid(np.dot(hidden_layer2, weights_hidden2_output)
                              + bias_output)

    output_error = outputs - output_layer

    cost_graph.append(np.mean(np.square(output_error)))

    output_delta  = output_error * sigmoid_derivative(output_layer)
    hidden2_delta = output_delta.dot(weights_hidden2_output.T) *
```

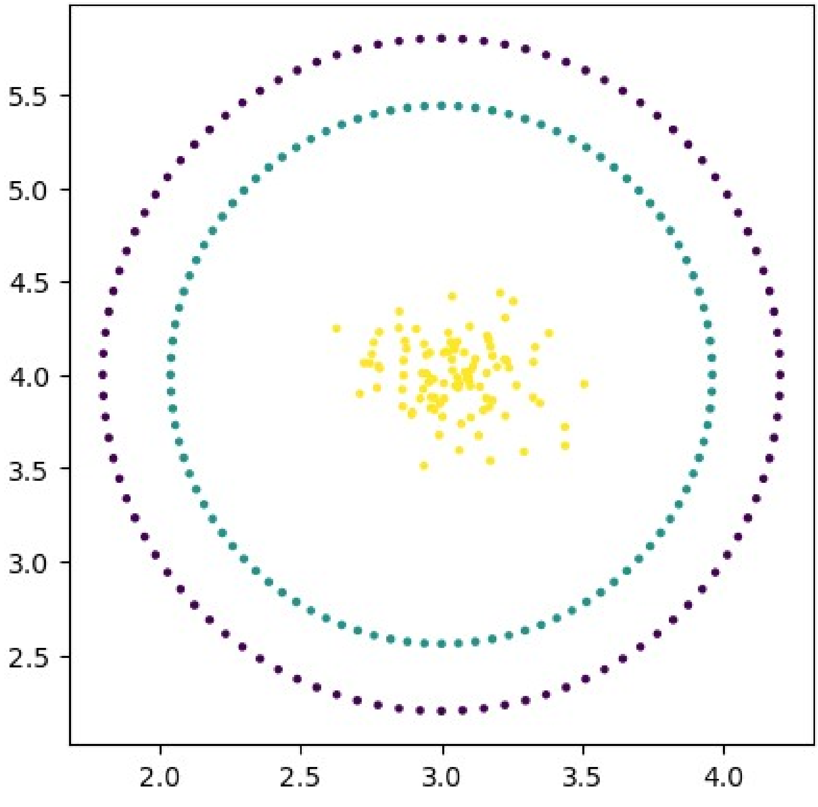
```
                              sigmoid_derivative(hidden_layer2)
    hidden1_delta = hidden2_delta.dot(weights_hidden1_hidden2.T) *
                              sigmoid_derivative(hidden_layer1)

    weights_hidden2_output  += lr * hidden_layer2.T.dot(output_delta)
    weights_hidden1_hidden2 += lr * hidden_layer1.T.dot(hidden2_delta)
    weights_input_hidden1   += lr * inputs.T.dot(hidden1_delta)

    bias_hidden1 += lr * np.sum(hidden1_delta, axis=0, keepdims=True)
    bias_hidden2 += lr * np.sum(hidden2_delta, axis=0, keepdims=True)
    bias_output  += lr * np.sum(output_delta, axis=0, keepdims=True)

plt.figure(figsize=(3, 2))
plt.plot(cost_graph)

plt.figure(figsize=(3, 2))
plt.scatter(x, y, c=output_layer, s=1)

plt.show()
```

Then answer the following questions:

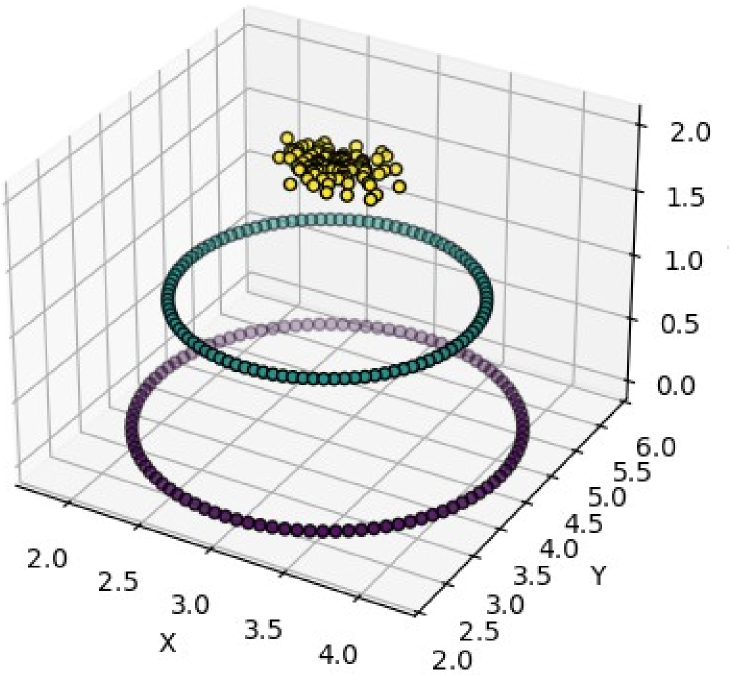
	Question	**Answer**
1	Chose a learning rate that is larger than the one you found in Task 2. Then for that particular learning rate, find a hidden layer size that gives you same or better results than Task 2.	

Task 4: Change the Dataset

We are now going to use the two circles dataset with some noise (two-circles.csv). The columns are x, y, and labels. When plotted it shows:



Answer the following questions:

	Question	Answer
1	Which ANN architecture is best suited for this dataset?	
2	What if we think of the labels as the z-axis for the scatter plot? Should it work now?  	

Deliverable

Submit a report containing answers asked, along with your full jupyter-notebook working.