Q1. Activation function: $\phi(x) = \frac{1}{1+e^{-x}}$; $\phi'(x) = \phi(x)(1-\phi(x))$

$I/p: X = [1, 1]^T$; t = Desired $o/p = [1, 0]^T$

Solution

We know, $J = \frac{1}{2}\sum_{i=1}^{N} e_i^2 = \frac{1}{2}\sum_{i=1}^{N} (t_i - y_i)^2$

$$\frac{dJ}{dw_{jL}} = -E_L \phi'(v_L) y_j$$

$$w_{jL}^{t+1} = w_{jL}^{t} + \eta E_L \phi'(v_L) y_j$$

Step 1: Forward Pass: To calculate the output $Y = [Y_1, Y_2]$

@ Node 1:

$$W = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} ; X = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$(W^T X + b) = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (b = 0)$$

$$V_{N1} = 0.2 \quad ; \quad y_{N1} = \phi(0.2)$$

$$= \frac{1}{1+e^{-0.2}} = 0.56$$

@ Node 2:

$$W = \begin{bmatrix} 0.2 \\ 0.7 \end{bmatrix} ; X = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$y_{N_2} = (W^T X + b) = [0.2 \; 0.7] \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$V_{N_2} = 0.95$$

$$y_{N_2} = \phi(V_{N_2}) = \frac{1}{1+e^{-0.95}} = \underline{0.72}$$

@ Node 3:

$$W = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} \quad X = \begin{bmatrix} 0.56 \\ 0.72 \end{bmatrix}$$

$$(W^T X + b) \Rightarrow [0.4 \; 0.6] \begin{bmatrix} 0.56 \\ 0.72 \end{bmatrix}$$

$$V_3 = 0.66$$

$$Y_{N_3} = \frac{1}{1+e^{.66}} = \underline{0.66}$$

@ Node 4:

$$X = \begin{bmatrix} 0.56 \\ 0.72 \end{bmatrix} \quad W = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$$

$$[W^T X + b] \Rightarrow [0.5 \; 0.3] \begin{bmatrix} 0.56 \\ 0.72 \end{bmatrix} = V_{N_4} = 0.48$$

$$Y_{N_4} = \frac{1}{1+e^{-0.48}} = 0.62$$

Final output $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.66 \\ 0.62 \end{bmatrix}$

Step 2 : Backward step : Backpropagation.

we can calculate Net Error at o/p $E_{Total} = \frac{1}{2}\sum (t-y)^2$

$$E_{Total} = \frac{1}{2}(1-0.66)^2 + \frac{1}{2}(0-0.62)^2$$

$$= 0.057 + 0.192$$

$$= 0.249.$$

Learning Rate : $\eta = 0.1$.

Layer 3 L

$$w_{jL}^{t+1} = w_{jL}^{t} + \eta E_L \phi'(v_L) y_j$$

$$w_{13}^{t+1} = w_{13}^{t} + 0.1(1-0.66)(0.66).(1-0.66) \times 0.56$$

w_{13}^{t} ⇓ 0.4

$$= 0.404$$

$$w_{24}^{t+1} = w_{24}^{t} + 0.1(0-0.62)(0.62)(1-0.62)0.72$$

$$= 0.289$$

- $w_{14}^{t+1} = w_{14}^{t} + \eta E_L \quad 0.1(0-0.62)(0.62)(1-0.62)0.56$

$= \underline{-0.49}$

- $w_{23}^{t+1} = w_{23}^{t} + (0.1)(-0.66)(0.66)(1-0.66)(0.72)$

$= \underline{0.605}$

Layer II

First lets calculate δ_L at Node 3 & Node 4

Node 3

$\delta_3 = E_L \phi'(v_L)$

$= (1-0.66)(0.66)(1-0.66) = \underline{0.076}$

Node 4

$\delta_4 = (0-0.62)(0.62)(1-0.62) = \underline{-0.146}$

Calculating δ_1 & δ_2

$\delta_1 = 0.56(1-0.56)(\delta_3 w_{13} + \delta_4 w_{14})$

$= 0.2464(0.0304 - 0.073)$

$\delta_1 = -0.01049$

$$\delta_2 = (0.72)(1-0.72)(\delta_3 w_{23} + \delta_4 w_{24})$$

$$= 0.2016(0.045 - 0.0438)$$

$$= 0.00024$$

Updating values:

$$w_{ij}^{t+1} = w_{ij} + (0.1)\,\delta_j\, y_i$$

(i → I, j → J)

- $w_{11} = 0.1 + (0.1)(-0.0105)(1)$

$$= 0.0989$$

- $w_{12}^{t+1} = w_{12}^{t} + (0.1)(\delta_2 y_1)$

$$= 0.25 + 0.000024$$

$$= 0.250024$$

- $w_{21}^{t+1} = w_{21}^{t} + (0.1)(\delta_1 y_2)$

$$= 0.1 + 0.1(-0.0105)(1)$$

$$= 0.0989$$

- $w_{22}^{t+1} = w_{22}^{t} + 0.1 \cdot \delta_2 y_2$

$$= 0.7 + (0.1)(0.00024)(1)$$

$$= 0.700024$$

HOME WORK 5

Rishab Lokray (UfId- 9357 3447)

Question 2. a ¶

For a 2 hidden layer MLP i have chosen a combination of 17 neurons in 1st later and 6 neurons in the 2nd hidden layer

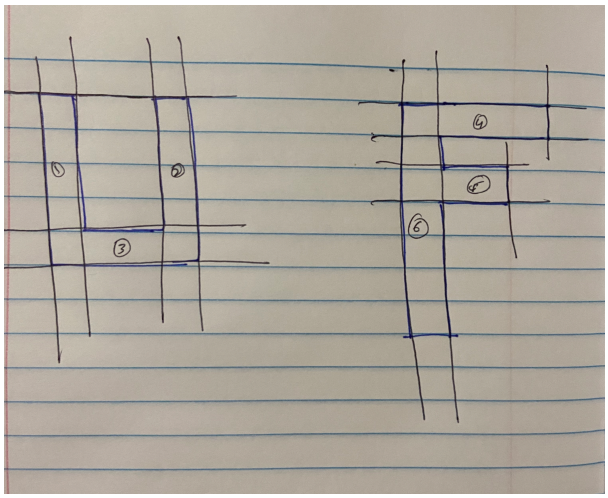
I have chosen these values as discussed in class, we need one neuron for each boundary of a shape. In our case there are 17 lines that define the U and F letters.

In second layer i have chosen 6 neurons as i have broken down the shapes into multiple rectangles. A total of 6 rectangles is enough to form U and F.

The output layer will have one neuron.

Roles:

1. 1st hidden layer is used to draw decision boundaries.
2. 2nd hidden layer is used to combine all these boundaries
3. 3rd output layer is used to organise these boundaries into classes.



Question 2. b

Yes, As discussed in class it is possible to get any classification output using just one single hidden layer.

This is given by the UNIVERSAL APPROXIMATION THEORAM.

"The *Universal Approximation Theorem* states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^N$, under mild assumptions on the activation function. The theorem thus states that simple neural networks can represent a wide variety of interesting functions when given appropriate parameters; however, it does not touch upon the algorithmic learnability of those parameters."

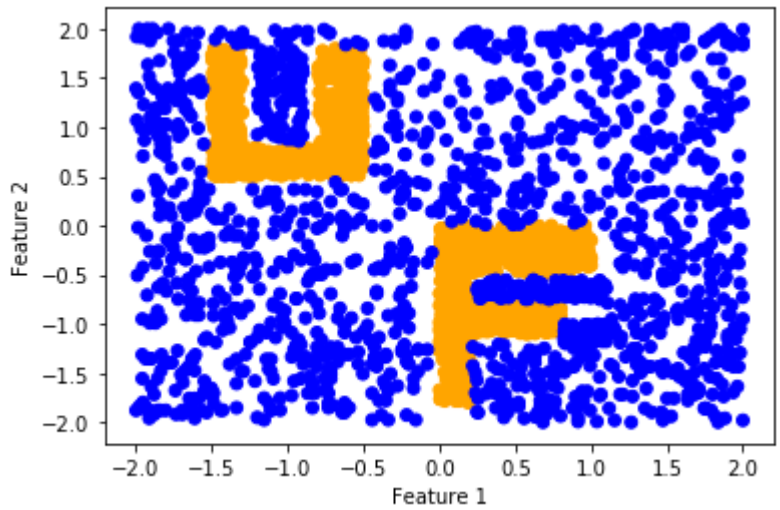
Essentially, the Universal Approximation Theorem states that a single hidden layer is sufficient for a multilayer perceptron to compute a uniform ϵ approximation to a given training set - provided you have the *right* number of neurons and the *right* activation function.

- However, this does not say that a single hidden layer is optimal with regards to learning time, generalization, etc.)
- In other words, a **feed-forward MLP with one hidden layer can approximate arbitrarily closely any continuous function**. (Wow!)

Question 2. c

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.colors import ListedColormap

UF_network = np.load('UF_network.npy')
X = UF_network[:,:2]
y = UF_network[:,2]
cm = ListedColormap(['blue', 'orange'])
plt.scatter(X[:,0],X[:,1],c=y, cmap=cm);
plt.xlabel('Feature 1')
plt.ylabel('Feature 2');
```



In [5]:

```
def plot_LC_and_DB(X, model):
    '''This function will plot the learning curve and decision boundary
    for a given trained model and training data X.
    model: is a sklearn model structure
    X: training data'''
    x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
    y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                         np.arange(y_min, y_max, 0.02))

    fig = plt.figure(figsize=(15,5))
    fig.add_subplot(1,2,1)
    plt.plot(model.loss_curve_)
    plt.title('Learning Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Loss Function')

    cm = ListedColormap(['blue', 'orange'])
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    fig.add_subplot(1,2,2)
    plt.contourf(xx, yy, Z, cmap=cm, alpha=.8);
    plt.title('Decision Boundary')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show();
```

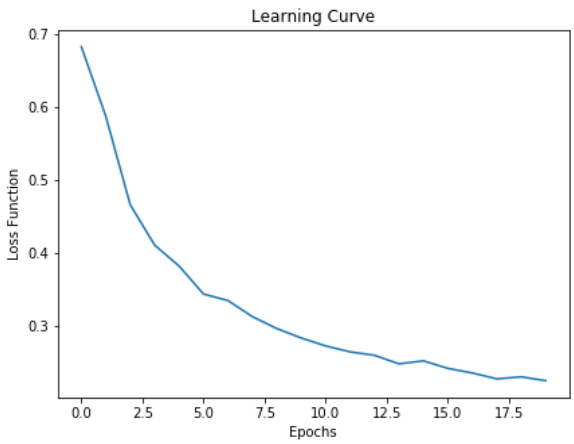
In [7]:

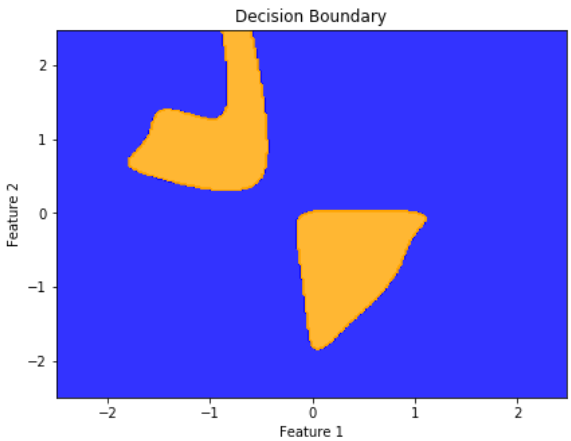
```
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier

# Do not change these parameters
net = MLPClassifier(activation='tanh',
                    n_iter_no_change = 1000)
```

In [16]:

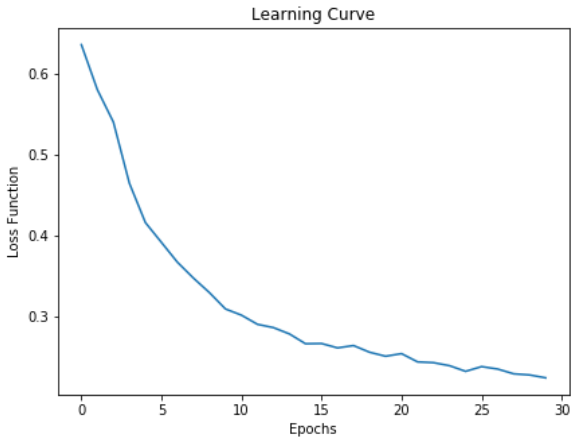
```
# Create a list with your own set of values for all of these parameters
net_hidden_layers = [(17,6)]
net_learning_rate = [0.04,0.1,0.2]
epochs = [20,30,110]

for i in net_hidden_layers:
    for j in net_learning_rate:
        for k in epochs:
            net.set_params(hidden_layer_sizes = i, learning_rate_init = j, max_iter
            net.fit(X, y)
            y_pred = net.predict(X)

            acc_score = accuracy_score(y, y_pred)
            print('-----------------------------------')
            print('Hidden Layer Architecture: ', i)
            print('Learning Rate: ', j)
            print('Number of Epochs: ', k)
            print('Accuracy = ', np.round(acc_score*100,2),'%')
            print('-----------------------------------')
            plot_LC_and_DB(X, net)
```

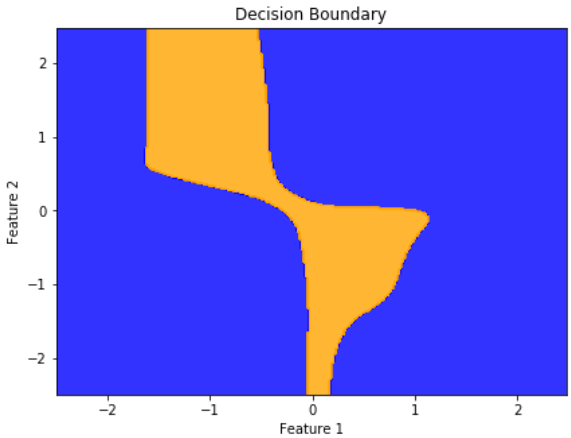
```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (20) reached and the optimiza
tion hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.04
Number of Epochs:  20
Accuracy =  91.67 %
-----------------------------------
```



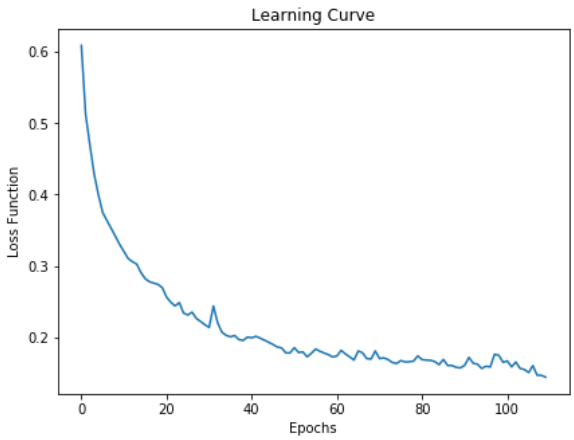


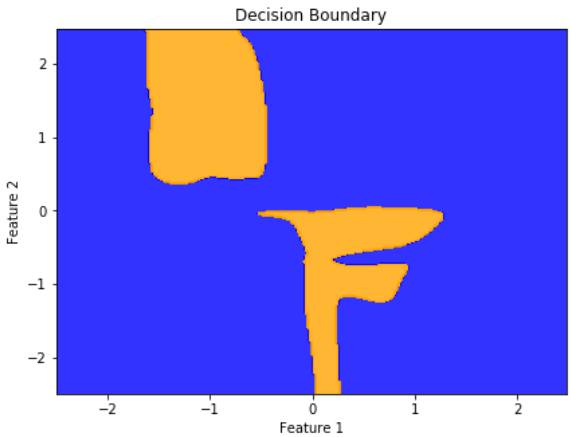

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (30) reached and the optimiza
tion hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.04
Number of Epochs:  30
Accuracy =  91.7 %
-----------------------------------
```



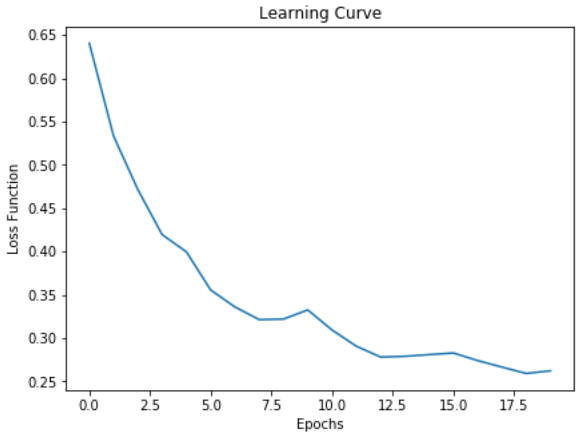


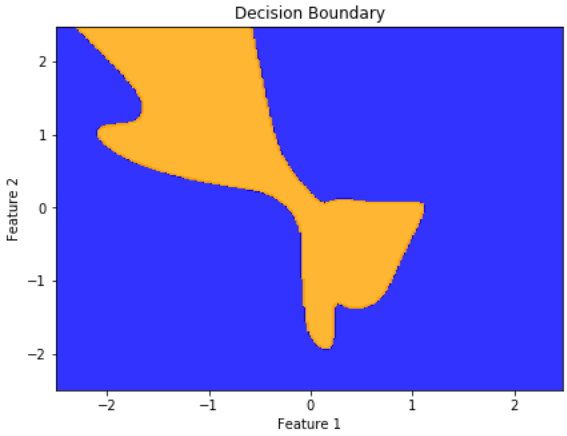

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (110) reached and the optimiz
ation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.04
Number of Epochs:  110
Accuracy =  94.83 %
-----------------------------------
```



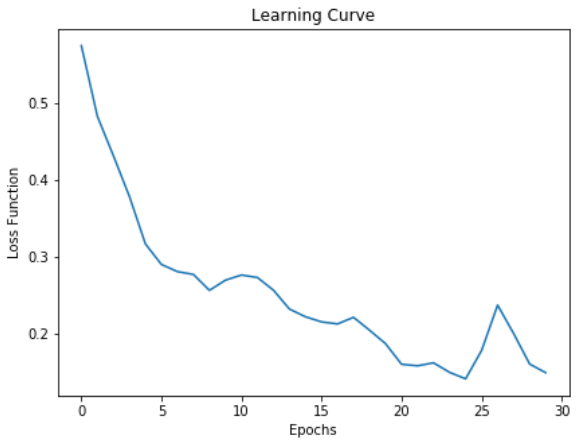


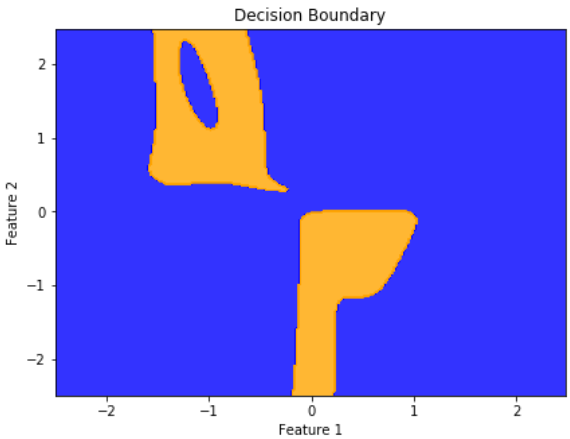

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (20) reached and the optimiza
tion hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.1
Number of Epochs:  20
Accuracy =  89.4 %
-----------------------------------
```



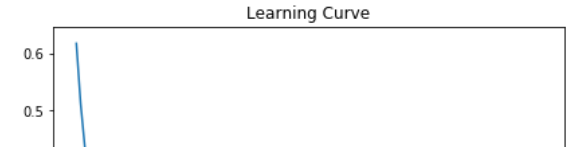


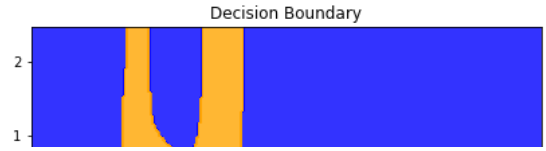

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (30) reached and the optimiza
tion hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.1
Number of Epochs:  30
Accuracy =  94.13 %
-----------------------------------
```



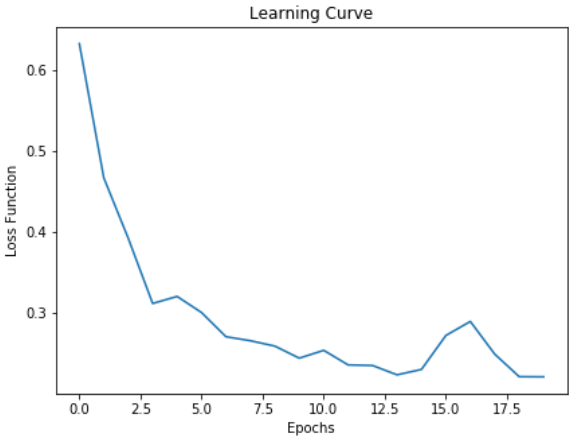


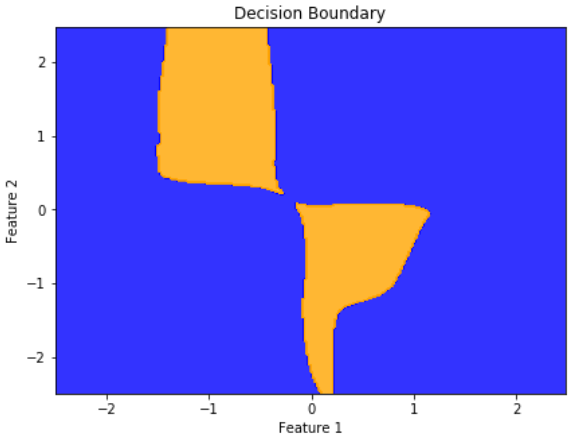

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (110) reached and the optimiz
ation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.1
Number of Epochs:  110
Accuracy =  97.87 %
-----------------------------------
```



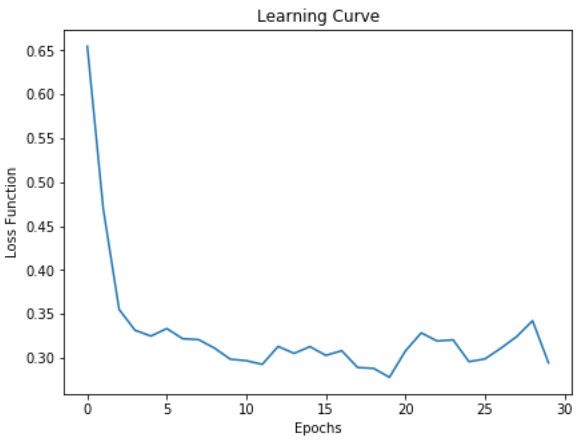


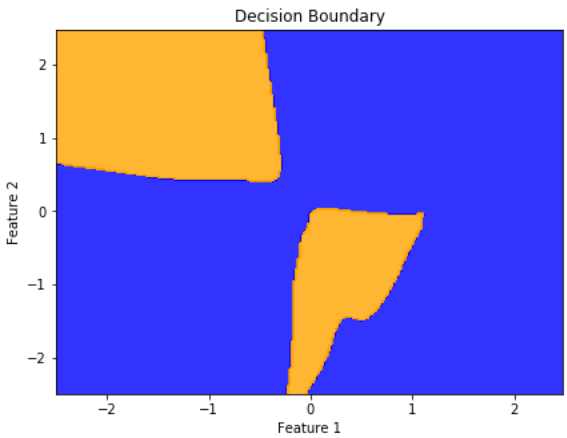

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (20) reached and the optimiza
tion hasn't converged yet.
  % self.max_iter, ConvergenceWarning)


-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.2
Number of Epochs:  20
Accuracy =  91.4 %
-----------------------------------
```





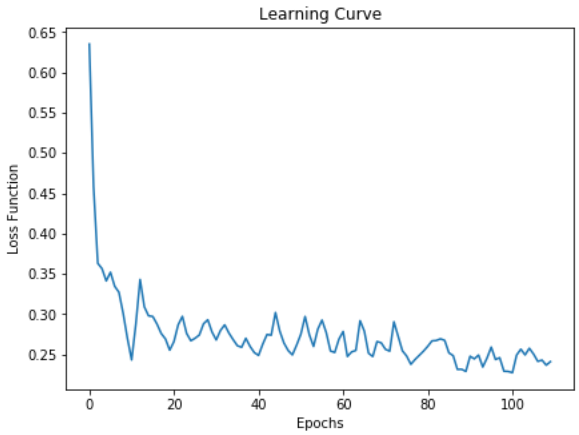

```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (30) reached and the optimiza
tion hasn't converged yet.
  % self.max_iter, ConvergenceWarning)


-----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.2
Number of Epochs:  30
Accuracy =  88.73 %
-----------------------------------
```

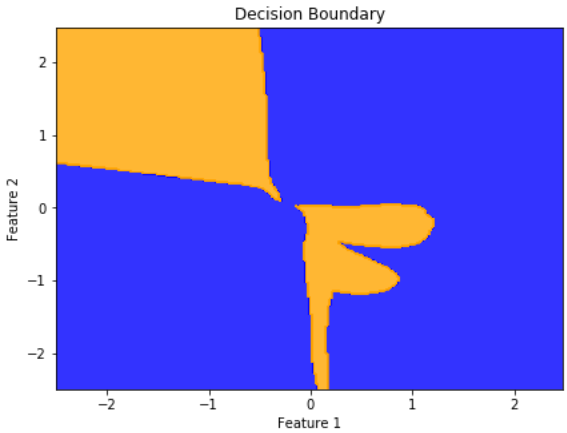





```
/Users/rishablokray/anaconda/anaconda3/lib/python3.7/site-packages/skl
earn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (110) reached and the optimiz
```

```
ation hasn't converged yet.
  % self.max iter, ConvergenceWarning)
----------------------------------
Hidden Layer Architecture:  (17, 6)
Learning Rate:  0.2
Number of Epochs:  110
Accuracy =  90.43 %
----------------------------------
```





The Best 6 combinations out of the above 9 are:

Hidden Layer Architecture: (17, 6) Learning Rate: 0.1 Number of Epochs: 110 Accuracy = 97.87 %

Hidden Layer Architecture: (17, 6) Learning Rate: 0.04 Number of Epochs: 110 Accuracy = 94.83 %

Hidden Layer Architecture: (17, 6) Learning Rate: 0.1 Number of Epochs: 30 Accuracy = 94.13 %

Hidden Layer Architecture: (17, 6) Learning Rate: 0.04 Number of Epochs: 30 Accuracy = 91.7 %

Hidden Layer Architecture: (17, 6) Learning Rate: 0.2 Number of Epochs: 20 Accuracy = 91.4 %

Hidden Layer Architecture: (17, 6) Learning Rate: 0.2 Number of Epochs: 110 Accuracy = 90.43 %

(Find graphs for these above cell)

Question 2.d

From the range of values i have chosen i got the best performance for learning rate = 0.1 ephoc = 110 Acuracy = 97.87%

--As discussed in class learning rate values can be inbetween 0 and 1. Small learning rate values will learn the data points slowly and takes much more computation power. While large learning rates have large gradient descent which updates the decision boundaries by a large amount, the issue with large rates is that it can result in larger convergence error and faster convergence.

-- Epoc chosen is 110 as with a small learning rate we would need more epocs so that the NN can learn all the data points. This would need more computation power but would have less convergence error.

In [ ]: