In this video we will see how we can use same model on different other datasets. And we will see how good it performs without changing anything in model architecture.

Testing on other non-linear datasets

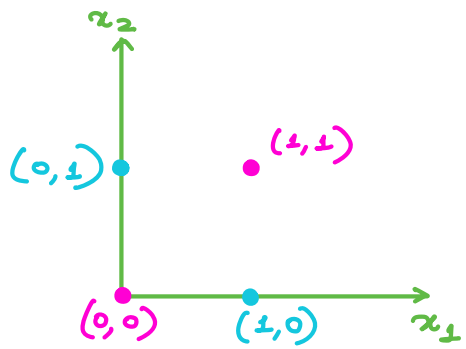
```
model = NeuralNetwork(input_size=2, layers=[10,5], output_size=2)
```

XOR Dataset

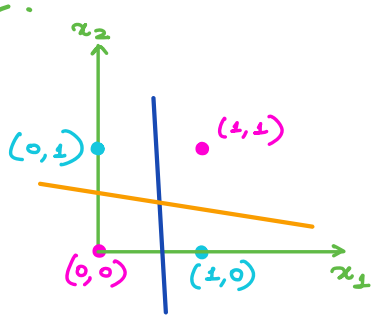
```
X = np.array([
                [0,0],
                [0,1],
                [1,0],
                [1,1]
               ])

Y = np.array([0,1,1,0])
```

XOR:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



XOR Data points are non linearly Separable. If you use a linear classifier you will only get 50% accuracy.



Now lets see if our model is able to classify XOR Dataset or not.

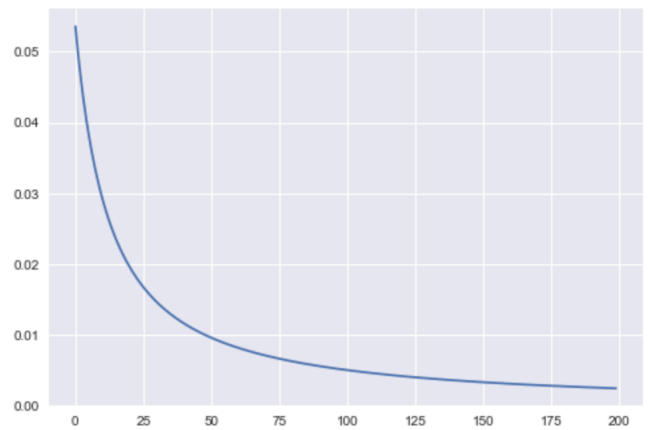
no. of epochs and learning rate is a hyperparameter, try different different values. For this example even a high value of learning rate will work.

```
losses = train(X, Y, model, 200, 0.1)
```

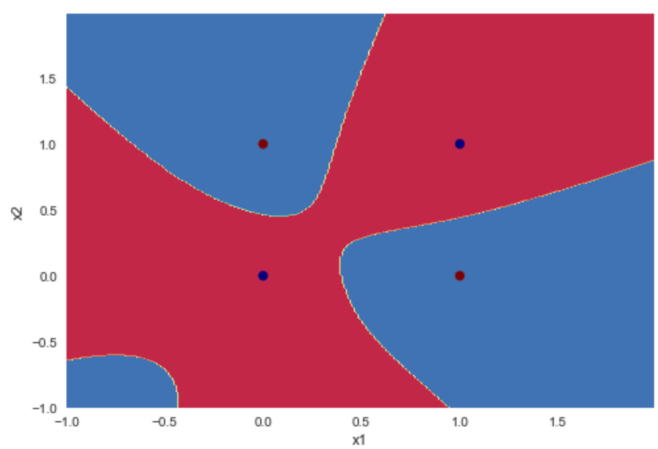
```
Epoch 181 Loss 0.0027
Epoch 182 Loss 0.0027
Epoch 183 Loss 0.0027
Epoch 184 Loss 0.0027
Epoch 185 Loss 0.0026
Epoch 186 Loss 0.0026
Epoch 187 Loss 0.0026
Epoch 188 Loss 0.0026
Epoch 189 Loss 0.0026
Epoch 190 Loss 0.0026
Epoch 191 Loss 0.0025
Epoch 192 Loss 0.0025
Epoch 193 Loss 0.0025
Epoch 194 Loss 0.0025
Epoch 195 Loss 0.0025
Epoch 196 Loss 0.0025
Epoch 197 Loss 0.0025
Epoch 198 Loss 0.0025
Epoch 199 Loss 0.0024
```

```
plt.plot(losses)
```

```
[<matplotlib.lines.Line2D at 0x12c350b50>]
```




```
plot_decision_boundary(lambda x:model.predict(x), X, Y)
```



Accuracy is 100%

```
outputs = model.predict(X)
training_accuracy = np.sum(outputs==Y)/Y.shape[0]
print("Training Accuracy %.4f"%(training_accuracy*100))
```

```
Training Accuracy 100.0000
```