In [180]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [181]:

```
import random
import numpy as np
import pandas as pd
import scipy.special
import matplotlib.pyplot as plt

np.random.seed(1)
```

In [182]:

```
class neuralNetwork:
    def __init__(self, inputnodes, hiddennodes, outputnodes, learningrate):
        self.inodes = inputnodes
        self.hnodes = hiddennodes
        self.onodes = outputnodes

        self.wih = np.random.normal(0.0, pow(self.hnodes, -0.5), (self.hnodes, self.inodes))
        self.who = np.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes))

        self.lr = learningrate

        self.sigmoid = lambda x: 1 / (1 + np.exp(-x))

        self.mse_doda = lambda t, a: (t - a)
        self.mse_dadz = lambda z: self.sigmoid(z) * (1.0 - self.sigmoid(z))
        self.mse_dzdw = lambda a: a

        pass

    def train(self, inputs_list, targets_list):
        inputs = np.array(inputs_list, ndmin = 2).T
        targets = np.array(targets_list, ndmin = 2)

        hidden_inputs = np.dot(self.wih, inputs)
        hidden_outputs = self.sigmoid(hidden_inputs)

        final_inputs = np.dot(self.who, hidden_outputs)
        final_outputs = self.sigmoid(final_inputs).T

        output_errors = - np.dot((self.mse_doda(targets, final_outputs).T *
                                  self.mse_dadz(final_inputs))
                                 , self.mse_dzdw(hidden_outputs).T)
        hidden_errors = np.dot(np.dot(output_errors.T
                                      , self.mse_dadz(hidden_inputs))
                               , self.mse_dzdw(hidden_outputs).T)

        self.who = self.who - output_errors * self.lr
        self.wih = self.wih - hidden_errors * self.lr

        pass

    def query(self, inputs_list):
        inputs = np.array(inputs_list, ndmin = 2).T

        hidden_inputs = np.dot(self.wih, inputs)
        hidden_outputs = self.sigmoid(hidden_inputs)

        final_inputs = np.dot(self.who.T, hidden_outputs)
        final_outputs = self.sigmoid(final_inputs).T

        return final_outputs
```

In [183]:

```
#output_errors = np.dot((self.mse_doda(targets, final_outputs).T * self.mse_dadz(final_inputs)), sel
#hidden_errors = np.dot(np.dot(output_errors.T, self.mse_dadz(hidden_inputs)), self.mse_dzdw(hidden_
```

In [184]:

```
input_nodes = 2
hidden_nodes = 2
output_nodes = 2

learning_rate = 0.5

n = neuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_rate)
```

In [185]:

```
training_data_list = [[3.5064385449265267, 2.345470928926325255, 0],
                     [4.384621956392097, 3.4530853889904205, 0],
                     [4.841442919897487, 4.0250785231752015 4, 0],
                     [3.5985868973088437, 4.1621314217538705, 0],
                     [2.887219775424049, 3.3152308252919 0005, 0],
                     [9.79822645535526, 1.1052409596099566, 1],
                     [7.8261241795117422, 0.6711054766067182, 1],
                     [2.5026163932400305, 5.800780055043912, 1],
                     [5.032436157202415, 8.650625621472184, 1],
                     [4.095084253434162, 7.69104329159447, 1]]
```

In [186]:

```
test_data_list = [[3.5064385449265267, 2.345470928926325255, 0],
                 [4.384621956392097, 3.4530853889904205, 0],
                 [4.841442919897487, 4.0250785231752015 4, 0],
                 [3.5985868973088437, 4.1621314217538705, 0],
                 [2.887219775424049, 3.3152308252919 0005, 0],
                 [9.79822645535526, 1.1052409596099566, 1],
                 [7.8261241795117422, 0.6711054766067182, 1],
                 [2.5026163932400305, 5.800780055043912, 1],
                 [5.032436157202415, 8.650625621472184, 1],
                 [4.095084253434162, 7.69104329159447, 1]]
```

In [187]:

```
epochs = 10000

for i in range(epochs):
    for record in training_data_list:
        all_values = record

        inputs = (np.asfarray(all_values[0:2]))

        if all_values[2] == 0:
            a = [1, 0]
        else:
            a = [0, 1]

        targets = np.asarray(a)

        n.train(inputs, targets)

        pass
    if (i % 1 == 10):
        print("--------------------------------------")
        print("epochs:", i)
        all_values
        inputs

        scorecard = []

        for record_ in test_data_list:
            all_values_ = record_
            correct_label_ = int(all_values_[2])
            inputs_ = (np.asfarray(all_values_[0:2]))
            outputs_ = n.query(inputs_)
            label_ = np.argmax(outputs_)

            outputs_

            print(correct_label_, "      correct label")
            print(label_, "       prediction₩n")


            if label_ == correct_label_:
                scorecard.append(1)
            else:
                scorecard.append(0)
                pass
            pass

        scorecard_array = np.asarray(scorecard)
        print("performance =", scorecard_array.sum() / scorecard_array.size, "₩n₩n₩n")
pass
```

In [188]:

```
scorecard = []

for record in test_data_list:
    all_values = record
    correct_label = int(all_values[2])
    inputs = (np.asfarray(all_values[0:2]))
    outputs = n.query(inputs)
    label = np.argmax(outputs)

    outputs

    print(correct_label, "      correct label")
    print(label, "       prediction₩n")

    if label == correct_label:
        scorecard.append(1)
    else:
        scorecard.append(0)
        pass
    pass

scorecard_array = np.asarray(scorecard)
print("performance =", scorecard_array.sum() / scorecard_array.size)
```

Out[188]:

```
array([[0.69493024, 0.29684188]])
```

```
0       correct label
0        prediction
```

Out[188]:

```
array([[0.67290588, 0.31978778]])
```

```
0       correct label
0        prediction
```

Out[188]:

```
array([[0.65901283, 0.33428757]])
```

```
0       correct label
0        prediction
```

Out[188]:

```
array([[0.64916886, 0.34452188]])
```

```
0       correct label
0        prediction
```

Out[188]:

```
array([[0.65563205, 0.33771836]])

0       correct label
0        prediction
```

Out[188]:

```
array([[0.66607667, 0.32697815]])

1       correct label
0        prediction
```

Out[188]:

```
array([[0.71986162, 0.27115518]])

1       correct label
0        prediction
```

Out[188]:

```
array([[0.59639558, 0.39978554]])

1       correct label
0        prediction
```

Out[188]:

```
array([[0.58576993, 0.41105035]])

1       correct label
0        prediction
```

Out[188]:

```
array([[0.59318036, 0.40324341]])

1       correct label
0        prediction

performance = 0.5
```

In [ ]:

```

```