In [2]:

```
from math import exp
from random import seed
from random import random

def initialize_network(n_inputs,n_hidden,n_outputs):
    network=list()
    hidden_layer=[{'weights':[random() for i in range(n_inputs+1)]} for i in range(n_hidden)]
    network.append(hidden_layer)
    output_layer=[{'weights':[random() for i in range(n_hidden+1)]} for i in range(n_outputs)]
    network.append(output_layer)
    return network

def activate(weights,inputs):
    activation=weights[-1]
    for i in range(len(weights)-1):
        activation+=weights[i]*inputs[i]
    return activation

def transfer(activation):
    return 1.0/(1.0+exp(-activation))

def forward_propagate(network,row):
    inputs=row
    for layer in network:
        new_inputs=[]
        for neuron in layer:
            activation=activate(neuron['weights'],inputs)
            neuron['output']=transfer(activation)
            new_inputs.append(neuron['output'])
        inputs=new_inputs
    return inputs

def transfer_derivative(output):
    return output*(1.0-output)

def backward_propagate_error(network,expected):
    for i in reversed(range(len(network))):
        layer=network[i]
        errors=list()
        if i!=len(network)-1:
            for j in range(len(layer)):
                error=0.0
                for neuron in network[i+1]:
                    error+=(neuron['weights'][j]*neuron['delta'])
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron=layer[j]
                errors.append(expected[j]-neuron['output'])
        for j in range(len(layer)):
            neuron=layer[j]
            neuron['delta']=errors[j]*transfer_derivative(neuron['output'])

def update_weights(network,row,l_rate):
    for i in range(len(network)):
        inputs=row[:-1]
        if i!=0:
            inputs=[neuron['output'] for neuron in network[i-1]]
        for neuron in network[i]:
            for j in range(len(inputs)):
                neuron['weights'][j]+=l_rate*neuron['delta']*inputs[j]
                neuron['weights'][-1]+=l_rate*neuron['delta']

def train_network(network,train,l_rate,n_epoch,n_outputs):
    for epoch in range(n_epoch):
        sum_error=0
        for row in train:
            outputs=forward_propagate(network,row)
            expected=[0 for i in range(n_outputs)]
            expected[row[-1]]=1
            sum_error+=sum([(expected[i]-outputs[i])**2 for i in range(len(expected))])
            backward_propagate_error(network,expected)
            update_weights(network,row,l_rate)
        print('>epoch=%d,lrate=%.3f,error=%.3f'%(epoch,l_rate,sum_error))

seed(1)
dataset=[[2.7810836,2.550537003,0],
         [1.465489372,2.362125076,0],
         [3.396561688,4.400293529,0],
         [1.38807019,1.850220317,0],
         [3.06407232,3.005305973,0],
         [7.627531214,2.759262235,1],
         [5.332441248,2.088626775,1],
         [6.922596716,1.77106367,1],
         [8.675418651,-0.242068655,1],
         [7.673756466,3.508563011,1]]
n_inputs=len(dataset[0])-1
n_outputs=len(set([row[-1] for row in dataset]))
network=initialize_network(n_inputs,2,n_outputs)
print(network)
train_network(network,dataset,0.5,5,n_outputs)
print('The layers are as follows')
for layer in network:
    print(layer)
```

```
[[{'weights': [0.13436424411240122, 0.8474337369372327, 0.763774618976614]}, {'weights': [0.2550690257394217, 0.4954350
8709194095, 0.4494910647887381]}], [{'weights': [0.651592972722763, 0.7887233511355132, 0.0938595867742349]}, {'weight
s': [0.02834747652200631, 0.8357651039198697, 0.43276706790505337]}]]
>epoch=0,lrate=0.500,error=6.226
>epoch=1,lrate=0.500,error=5.397
>epoch=2,lrate=0.500,error=5.269
>epoch=3,lrate=0.500,error=5.068
>epoch=4,lrate=0.500,error=4.638
The layers are as follows
[{'weights': [-0.7444356666741193, 0.7284228301055004, 0.6113616076781042], 'output': 0.09725284017225566, 'delta': -0.
008672831331566391}, {'weights': [0.1450573285090563, 0.35035454986710357, 0.34466898073031205], 'output': 0.9328121730
435498, 'delta': 0.0015392207627615757}]
[{'weights': [0.5638197606549719, 0.3028360642948306, -0.9085628564473949], 'output': 0.39258703768368464, 'delta': -0.
09361726901390004}, {'weights': [-0.5259158798627953, 0.7030440434263575, 0.05486804156053663], 'output': 0.63082095350
3204, 'delta': 0.08597658642857658}]
```