# Homework 5

Ze Liu (zl265)

Question 1

(1)

```python
# Homework 5
# Software Engineering and Web Application
# -*- coding: utf-8 -*-
import  math
import  random
random.seed( 0 )

def   randomNum (a, b):
    return  (b - a) * random.random() + a

def   constructMatrix (I, J, fill= 0.0 ):
    m = []
    for  i  in  range(I):
        m.append([fill] * J)

    return  m

def   sigmoid (x):
    return  1.0  / ( 1.0 + math.exp(-x))


def   dsigmoid (y):
    return  y * ( 1  - y)

# Create a matrix with random value between a and b
def   randomizeMatrix (matrix, a, b):
    for  i  in  range(len(matrix)):
        for  j  in  range(len(matrix[ 0 ])):
            matrix[i][j] = random.uniform(a, b)

class   NN :
    def   __init__ (self, ni, nh, no):
        # number of input, hidden, and output nodes
        """
        create the neural network
        :param ni:number of input unit
        :param nh:number of hidden layer unit
        :param no:number of output unit
        """
```

```
40          self.ni = ni +  1
41          self.nh = nh
42          self.no = no
43
44          self.ai = [ 1.0 ] * self.ni
45          self.ah = [ 1.0 ] * self.nh
46          self.ao = [ 1.0 ] * self.no
47
48          # weight matrix
49          self.wi = constructMatrix(self.ni, self.nh)
50          self.wo = constructMatrix(self.nh, self.no)
51
52          randomizeMatrix(self.wi,  -1 ,  1 )
53          randomizeMatrix(self.wo,  -1 ,  1 )
54          print   ("\n"  +  'Initial weights:' )
55          print   ('Theta1: ')
56          for  i  in  range(self.ni):
57              print  (self.wi[i])
58          print   ('Theta2: ')
59
60          for  j  in  range(self.nh):
61              print  (self.wo[j])
62          self.ci = constructMatrix(self.ni, self.nh)
63          self.co = constructMatrix(self.nh, self.no)
64
65      def  runNN (self, inputs):
66          if  len(inputs) != self.ni -  1 :
67              print   ('incorrect number of inputs')
68          for  i  in  range(self.ni -  1 ):
69              self.ai[i] = inputs[i]
70          for  j  in  range(self.nh):
71              sum =  0.0
72              for  i  in  range(self.ni):
73                  sum += (self.ai[i] * self.wi[i][j])
74              self.ah[j] = sigmoid(sum)
75
76          for  k  in  range(self.no):
77              sum =  0.0
```

```
78              for  j  in  range(self.nh):
79                  sum += (self.ah[j] * self.wo[j][k])
80              self.ao[k] = sigmoid(sum)
81          return  self.ao
82
83      def  backPropagate (self, targets, N, M):
84          """
85          backpropagation function
86          :param targets:
87          :param N: learning rate
88          :param M: old learning rate
89          :return:
90          """
91
92          # the delata of output layer
93          output_deltas = [ 0.0 ] * self.no
94          for  k  in  range(self.no):
95              error = targets[k] - self.ao[k]
96              output_deltas[k] = error * dsigmoid(self.ao[k])
97
98          # update the Theta2
99          for  j  in  range(self.nh):
100             for  k  in  range(self.no):
101                 change = output_deltas[k] * self.ah[j]
102                 self.wo[j][k] += N * change + M * self.co[j][k]
103                 self.co[j][k] = change
104
105         # the delta for hidden layer
106         hidden_deltas = [ 0.0 ] * self.nh
107         for j in  range(self.nh):
108             error =  0.0
109             for  k  in  range(self.no):
110                 error += output_deltas[k] * self.wo[j][k]
111             hidden_deltas[j] = error * dsigmoid(self.ah[j])
112
113         # update the Theta1
114         for  i  in  range(self.ni):
115             for  j  in  range(self.nh):
116                 change = hidden_deltas[j] * self.ai[i]
117                 self.wi[i][j] += N * change + M * self.ci[i][j]
```

```
                self.ci[i][j] = change

        error = 0.0
        for k in range(len(targets)):
            error = 0.5 * (targets[k] - self.ao[k]) ** 2

        return error

    def weights (self):
        print ("\n" + 'Final weights:')
        print ('Theta 1: ')
        for i in range(self.ni):
            print (self.wi[i])
        print ('Theta 2: ')
        for j in range(self.nh):
            print (self.wo[j])
        print ('')

    def test (self, patterns):
        print ("\n")
        for p in patterns:
            inputs = p[ 0 ]
            print ('Inputs:' , p[ 0 ], '-->' , self.runNN(inputs), 'Target' .rjust( 10 ), p[ 1 ])

    def train (self, patterns, max_iterations= 1000 , N= 0.5 , M= 0.1 ):
        N = learningRate
        for i in range(max_iterations):
            for p in patterns:
                inputs = p[ 0 ]
                targets = p[ 1 ]
                self.runNN(inputs)
                error = self.backPropagate(targets, N, M)

            if i == 0 :
                print 'first-batch error ' , error
            if error < expectedError:
                print 'final error ' , error
                print 'the total number of batches run through in the training: ' , i + 1
                break
        self.test(patterns)

def main ():
    pat = [[[ 0 , 0 ], [ 1 ]],
           [[ 1 , 0 ], [ 1 ]],
           [[ 0 , 1 ], [ 1 ]],
           [[ 1 , 1 ], [ 0 ]]]
    global expectedError
    global learningRate
    learningRate = input( 'Please input the learning rate: ' )
    expectedError = input( 'Please input the target error: ' )
    myNN = NN( 2 , 2 , 1 )
    myNN.train(pat)
    myNN.weights()


if __name__ == "__main__":
    main()
```

(2)

learning rate η = 0.5

target error ε = 0.1

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 0.5
Please input the target error: 0.1

Initial weights:
Theta1:
[0.6888437030500962, 0.5159088058806O5]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.164695634571
final error  0.0989567822614
the total number of batches run through in the training:  516


Inputs: [0, 0] --> [0.9864097463601232]     Target [1]
Inputs: [1, 0] --> [0.8785586137485344]     Target [1]
Inputs: [0, 1] --> [0.890590316496019]     Target [1]
Inputs: [1, 1] --> [0.4398787848666478]     Target [0]

Final weights:
Theta 1:
[-3.4206156287269534, 0.1274302567356514]
[-3.5245935009333786, -0.8085153587030462]
[3.4887002291316866, 0.11068263889842679]
Theta 2:
[5.01006572041616]
[-1.0932994237781106]
```

learning rate η = 1

target error ε = 0.1

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 1
Please input the target error: 0.1

Initial weights:
Theta1:
[0.6888437030500962, 0.5159088058806O5]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.179213976801
final error  0.0978167515097
the total number of batches run through in the training:  263


Inputs: [0, 0] --> [0.9843976113348563]     Target [1]
Inputs: [1, 0] --> [0.8707987877158202]     Target [1]
Inputs: [0, 1] --> [0.8819107651684323]     Target [1]
Inputs: [1, 1] --> [0.4308813433201841]     Target [0]

Final weights:
Theta 1:
[-3.3284440271771856, 0.161344117635836231]
[-3.445590469460113, -0.7507966125665964]
[3.412236238784956, 0.1924105256974033]
Theta 2:
[4.904401489899515]
[-1.100949443575389]
```

Find the best one:

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 13.5
Please input the target error: 0.1

Initial weights:
Theta1:
[0.6888437030500962, 0.5159088058806O5]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.408161731044
final error  0.0557070274915
the total number of batches run through in the training:  26


Inputs: [0, 0] --> [0.9741428620766073]     Target [1]
Inputs: [1, 0] --> [0.9123403789968815]     Target [1]
Inputs: [0, 1] --> [0.8305550389593628]     Target [1]
Inputs: [1, 1] --> [0.1413619774362927]     Target [0]

Final weights:
Theta 1:
[-4.272617544786997, 1.1484788843605491]
[-4.979201621096145, 0.5357307966799636]
[5.526093415723924, 2.570219402255462]
Theta 2:
[5.471547994318959]
[-1.9602093953335826]
```

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 13.4
Please input the target error: 0.1

Initial weights:
Theta1:
[0.6888437030500962, 0.5159088058806 05]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.190131725099171 42]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.407572102623
final error  0.0412365573006
the total number of batches run through in the training:  27


Inputs: [0, 0] --> [0.9840656181280987]     Target [1]
Inputs: [1, 0] --> [0.9718743090999047]     Target [1]
Inputs: [0, 1] --> [0.45193560237759284]     Target [1]
Inputs: [1, 1] --> [0.1575557476028034]     Target [0]

Final weights:
Theta 1:
[-3.566432828181904, 1.2239192885205004]
[-6.951149558197828, 0.6021420747600211]
[5.897211468429037, 2.580544632388623]
Theta 2:
[5.769472927242636]
[-1.7539323682579742]
```

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 13.6
Please input the target error: 0.1

Initial weights:
Theta1:
[0.6888437030500962, 0.515908805880605]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.190131725099171 42]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.408741542335
final error  0.050960917836
the total number of batches run through in the training:  29


Inputs: [0, 0] --> [0.9812615625367586]     Target [1]
Inputs: [1, 0] --> [0.9622611796895307]     Target [1]
Inputs: [0, 1] --> [0.9403764509692161]     Target [1]
Inputs: [1, 1] --> [0.11950787766653459]     Target [0]

Final weights:
Theta 1:
[-5.1214707288619525, 1.2515556609279814]
[-5.822343206595036, 0.652503681564927]
[7.27132682763478, 2.6631102289639488]
Theta 2:
[5.98892336231091]
[-2.1678105894468316]
```

When the learning rate is 13.5, it has the lowest total number of batches run through in the training. So it is the best one.

(3)

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 0.5
Please input the target error: 0.02

Initial weights:
Theta1:
[0.6888437030500962, 0.515908805880605]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.190131725099171 42]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.164695634571
final error  0.019841296076
the total number of batches run through in the training:  596


Inputs: [0, 0] --> [0.9826296445620369]     Target [1]
Inputs: [1, 0] --> [0.8959942194529078]     Target [1]
Inputs: [0, 1] --> [0.9050502595236514]     Target [1]
Inputs: [1, 1] --> [0.19606489672635066]     Target [0]

Final weights:
Theta 1:
[-3.315075843133763, 0.9053106659984235]
[-3.545398618860515, -0.04055597800533122]
[4.213404955011452, 0.688566187062957]
Theta 2:
[5.550569680479396]
[-2.154553732639574]
```

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 1
Please input the target error: 0.02

Initial weights:
Theta1:
[0.6888437030500962, 0.515908805880605]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.179213976801
final error  0.0197119255818
the total number of batches run through in the training:  309


Inputs: [0, 0] --> [0.9813459418299816]     Target [1]
Inputs: [1, 0] --> [0.8940016747782182]     Target [1]
Inputs: [0, 1] --> [0.9011711605234614]     Target [1]
Inputs: [1, 1] --> [0.1918494872536116]     Target [0]

Final weights:
Theta 1:
[-3.2483646294854824, 0.9324670750704473]
[-3.49017680125415, -0.0008717872912126032]
[4.187559949383308, 0.7356369725669053]
Theta 2:
[5.522278933629905]
[-2.184520146944271]
```

Find the best one

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 9
Please input the target error: 0.02

Initial weights:
Theta1:
[0.6888437030500962, 0.515908805880605]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.367630524656
final error  0.0180657468408
the total number of batches run through in the training:  33


('Inputs:', [0, 0], '-->', [0.9753866275951452], '    Target', [1])
('Inputs:', [1, 0], '-->', [0.91402742423192], '    Target', [1])
('Inputs:', [0, 1], '-->', [0.881137719324144], '    Target', [1])
('Inputs:', [1, 1], '-->', [0.14182301999462066], '    Target', [0])

Final weights:
Theta 1:
[-3.748137748912163, 1.1939289655633085]
[-4.171609659592526, 0.45985985628994624]
[5.071572384190708, 2.0304884213105843]
Theta 2:
[5.625115065728524]
[-2.1613020766527926]
```

```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 10
Please input the target error: 0.02

Initial weights:
Theta1:
[0.6888437030500962, 0.5159088058806O5]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.379749158088
final error  0.0170812739176
the total number of batches run through in the training:  31


('Inputs:', [0, 0], '-->', [0.9775452962179334], '    Target', [1])
('Inputs:', [1, 0], '-->', [0.9276672412930285], '    Target', [1])
('Inputs:', [0, 1], '-->', [0.889170667013532], '    Target', [1])
('Inputs:', [1, 1], '-->', [0.13959932697051047], '    Target', [0])

Final weights:
Theta 1:
[-3.9829378573106715, 1.2032564606711313]
[-4.503472886958584, 0.4955135909596904]
[5.4109076722372045, 2.1764873680609607]
Theta 2:
[5.696565947229504]
[-2.112956776452243]
```

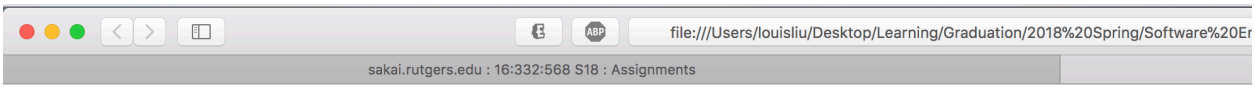
```
Louis:Assignment5 louisliu$ python BP.py
Please input the learning rate: 11
Please input the target error: 0.02

Initial weights:
Theta1:
[0.6888437030500962, 0.515908805880605]
[-0.15885683833831, -0.4821664994140733]
[0.02254944273721704, -0.19013172509917142]
Theta2:
[0.5675971780695452]
[-0.3933745478421451]
first-batch error  0.389801150042
final error  0.0135081009215
the total number of batches run through in the training:  31


('Inputs:', [0, 0], '-->', [0.9806330452178399], '    Target', [1])
('Inputs:', [1, 0], '-->', [0.9588676930556637], '    Target', [1])
('Inputs:', [0, 1], '-->', [0.8988690453809529], '    Target', [1])
('Inputs:', [1, 1], '-->', [0.12451950134101425], '    Target', [0])

Final weights:
Theta 1:
[-3.9110595993358865, 1.219187849679111]
[-5.071985304321048, 0.5351898643416794]
[6.026411220735808, 2.2997997594369557]
Theta 2:
[6.017424600980937]
[-2.2867125344886547]
```

When the learning rate is 10 and 11, the number of batches is the same. And they are both smaller than when the rate is 9. The best learning rate is: 10

Question3:

```
<!DOCTYPE html>
<html>
<head>
<title>Volume Calculation</title>
</head>
<body>
<h2> This web site will find the volume for a Cylinder, Spherre, or Cone</h2>

<form id="Form">
<p>Select the units (English or SI)<br />
<input type="radio" name="shape" value="English" onclick="click(this)" checked />English
<input type="radio" name="shape" value="SI" onclick="click(this)" />SI
</p>

<p>Select the shape
<select id="Shape" onclick="type_click(this)">
    <option value="Cylinder">Cylinder</option>
    <option value="Sphere">Sphere</option>
    <option value="Cone">Cone</option>
</select><br></p>

<p>Enter the radius <input id="r" style="text-align:right" oninput="radius_input(this)"></input><br /></p>
<p>For the cylinder and cone, Enter the height <input id="h" style="text-align:right" oninput="height_input(this)"></input><br />
</p>
</form>
<button onclick="reset()">reset the form</button><br>
<h2> Results</h2>
<p>You selected to use <span id="unit_show">English</span> units<br></p>

<p>You selected to find the value for a <span id="type_show">cylinder</span> shape<br> </p>
<table border="3">
    <thead>
    <tr>
        <td>Shape</td>
        <td>Radius</td>
        <td>Height</td>
        <td>Volume</td>
    </tr>
     </tr>
     </thead>
     <tbody>
     <tr>
         <td> </td>
         <td>(<span id ="cal_unit1">ft</span>)</td>
         <td>(<span id ="cal_unit2">ft</span>)</td>
         <td>(<span id ="cal_unit3">ft</span>^3)</td>
     </tr>
     <tr>
         <td><span id="type_show1">cylinder</span></td>
         <td><span id="radius"></span></td>
         <td><span id="height"></span></td>
         <td><span id="vol"></span></td>
     </tr>
     </tbody>
</table>

<button onclick="Calculate()">Click to calculate<br> the results</button><br>


<script type="text/javascript">
    function reset() {
        var x = document.forms["Form"];
        var obj = document.getElementById('Shape');
        x.r.value = "";
        x.h.value = "";
        obj.selectedIndex = 0;
        document.getElementById('vol').innerHTML = "";
        document.getElementById('radius').innerHTML = "";
        document.getElementById('height').innerHTML = "";
        document.getElementById("cal_unit1").innerHTML = "ft";
        document.getElementById("cal_unit2").innerHTML = "ft";
        document.getElementById("cal_unit3").innerHTML = "ft";
        document.getElementById('type_show').innerHTML = "Cylinder";
        document.getElementById('type_show1').innerHTML = "Cylinder";
    }
```

```
    }
    function radius_input(obj) {
        document.getElementById('radius').innerHTML = obj.value;
    }
    function height_input(obj) {
        document.getElementById('height').innerHTML = obj.value;
    }

    function click(obj){
        var unit = obj.value;
        document.getElementById('unit_show').innerHTML = unit;
        if (unit == "English") {
            document.getElementById("cal_unit1").innerHTML = "ft";
            document.getElementById("cal_unit2").innerHTML = "ft";
            document.getElementById("cal_unit3").innerHTML = "ft";
        } else {
            document.getElementById("cal_unit1").innerHTML = "m";
            document.getElementById("cal_unit2").innerHTML = "m";
            document.getElementById("cal_unit3").innerHTML = "m";
        }
    }
    function type_click(obj){
        var index = obj.selectedIndex;
        var type = obj.options[index].value;
        document.getElementById('type_show').innerHTML = type;
        document.getElementById('type_show1').innerHTML = type;
    }
</script>
<script type="text/javascript">
    function Calculate() {
        var x = document.forms["Form"];
        var v;
        var index = x.Shape.selectedIndex;
        var type = x.Shape.options[index].value;
        if (x.r.value == "") {
            window.alert("Please enter your radius!");
            return;
              return;
         }
         switch(type) {
             case "Cylinder":
                 if (x.h.value == "") {
                     window.alert("Please enter your height!");
                     return;
                 }
                 v = 3.1415926 * x.r.value * x.r.value * x.h.value;
                 break;
             case "Sphere":
                 v = 4/3 * x.r.value * x.r.value * x.r.value * 3.1415926;
                 break;
             case "Cone":
                 if (x.h.value == "") {
                     window.alert("Please enter your height!");
                     return;
                 }
                 v = 1/3 * x.r.value * x.r.value * 3.1415926 * x.h.value;
                 break;
         }
         document.getElementById('vol').innerHTML = v;
         document.getElementById('radius').innerHTML = x.r.value;
         document.getElementById('height').innerHTML = x.h.value;
     }
</script>

</body>
</html>
```




# This web site will find the volume for a Cylinder, Spherre, or Cone

Select the units (English or SI)
(•) English ( ) SI

Select the shape Cylinder

Enter the radius 1

For the cylinder and cone, Enter the height 1

reset the form

## Results

You selected to use English units

You selected to find the value for a cylinder shape

| Shape | Radius | Height | Volume |
|---|---|---|---|
| | (ft) | (ft) | (ft^3) |
| cylinder | 1 | 1 | 3.1415926 |

Click to calculate the results