Assignment_No_1

February 26, 2020

[1]:

```
#====================================================================================
# Assignment NO.1 : Back Propagation Algorithm
# Name: Mahesh Tatyasaheb Chavan
# Roll No.: BETB64
#====================================================================================
```

[2]:

```
import numpy as np
def tanh(x):
    return(np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x))
def tanh_derivative(x):
    return(1-((np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x)))**2)
```

[3]:

```
training_inputs = np.array([[0,0,1],
                            [1,1,1],
                            [1,0,1],
                            [0,1,1]])
training_outputs = np.array([[0,1,1,0]]).T
np.random.seed(1)
synaptic_weights = 2*np.random.random((3,1))-1
print('Random Starting synaptic weights: ')
print(synaptic_weights)
```

```
Random Starting synaptic weights:
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
```

[5]:

```
for iteration in range(1):
    input_layer = training_inputs
    outputs=tanh(np.dot(input_layer,synaptic_weights))
    error=training_outputs-outputs
    adjustments=error*tanh_derivative(outputs)
    synaptic_weights=synaptic_weights + np.dot(input_layer.T,adjustments)
print('Outputs after 1st iterations:')
print(outputs)
```

```
Outputs after 1st iterations:
[[0.96111914]
```

```
 [0.99998424]
 [0.99919809]
 [0.99922115]]
```

[6]:

```
for iteration in range(2):
    input_layer = training_inputs
    outputs=tanh(np.dot(input_layer,synaptic_weights))
    error=training_outputs-outputs
    adjustments=error*tanh_derivative(outputs)
    synaptic_weights=synaptic_weights + np.dot(input_layer.T,adjustments)
print('Outputs after 2nd iterations:')
print(outputs)
```

```
Outputs after 2nd iterations:
[[0.24055659]
 [0.99739234]
 [0.97564141]
 [0.87824804]]
```

[7]:

```
for iteration in range(10):
    input_layer = training_inputs
    outputs=tanh(np.dot(input_layer,synaptic_weights))
    error=training_outputs-outputs
    adjustments=error*tanh_derivative(outputs)
    synaptic_weights=synaptic_weights + np.dot(input_layer.T,adjustments)
print('Outputs after 10 iterations:')
print(outputs)
```

```
Outputs after 10 iterations:
[[-0.21819918]
 [ 0.95411964]
 [ 0.96412559]
 [-0.3340008 ]]
```

[9]:

```
for iteration in range(1000):
    input_layer = training_inputs
    outputs=tanh(np.dot(input_layer,synaptic_weights))
    error=training_outputs-outputs
    adjustments=error*tanh_derivative(outputs)
    synaptic_weights=synaptic_weights + np.dot(input_layer.T,adjustments)
print('Outputs after 1000 iterations:')
print(outputs)
```

```
Outputs after 1000 iterations:
[[-0.33026858]
 [ 0.99940135]
 [ 0.99959432]
 [-0.49127047]]
```

[10]:

```
for iteration in range(5000):
    input_layer = training_inputs
    outputs=tanh(np.dot(input_layer,synaptic_weights))
    error=training_outputs-outputs
    adjustments=error*tanh_derivative(outputs)
    synaptic_weights=synaptic_weights + np.dot(input_layer.T,adjustments)
print('Outputs after 5000 iterations:')
print(outputs)
```

```
Outputs after 5000 iterations:
[[-0.33024703]
 [ 0.99982755]
 [ 0.99988317]
 [-0.49133189]]
```

[11]:

```
for iteration in range(50000):
    input_layer = training_inputs
    outputs=tanh(np.dot(input_layer,synaptic_weights))
    error=training_outputs-outputs
    adjustments=error*tanh_derivative(outputs)
    synaptic_weights=synaptic_weights + np.dot(input_layer.T,adjustments)
print('Outputs after 2nd iterations:')
print(outputs)
```

```
Outputs after 2nd iterations:
[[-0.33023937]
 [ 0.99997876]
 [ 0.99998561]
 [-0.49135348]]
```

[ ]: