EXPERIMENT – VI

Developed By: Vishruth Khare
Roll Number: 2K18/CO/393

IDE: PyCharm
Compiler: Terminal (Python 3.6)
Source Code:

```
import numpy as np

np.random.seed(0)

class OneHiddenLayerNetwork:

    #
    # activation functions
    #

    @staticmethod
    def tang(y):
        return np.tanh(y)

    @staticmethod
    def derivative_tang(y):
        return 1.0 - y ** 2

    @staticmethod
    def sigmoid(y):
        return 1 / (1 + np.exp(-y))

    @staticmethod
    def derivative_sigmoid(y):
        return y * (1 - y)

    def __init__(self, learning_rate=0.1):

        #
        # neural network architecture
        #
        self.learning_rate = learning_rate
        self.output = None

        # weights with random values
        self.weights = [
            np.random.uniform(low=-0.2, high=0.2, size=(2, 2)),  # input
layer
            np.random.uniform(low=-2, high=2, size=(2, 1))  # hidden layer
        ]

    def activation(self, activation_type, y):
        if activation_type == 'sigmoid':
            return self.sigmoid(y)
        if activation_type == 'tang':
            return self.tang(y)
```

```
        raise ValueError('Undefined activation function:
{}'.format(activation_type))

    def derivative_activation(self, activation_type, y):
        if activation_type == 'sigmoid':
            return self.derivative_sigmoid(y)
        if activation_type == 'tang':
            return self.derivative_tang(y)

        raise ValueError('Undefined derivative activation function:
{}'.format(activation_type))

    def feed_forward_pass(self, x_values):

        # forward
        input_layer = x_values
        hidden_layer = self.activation('tang', np.dot(input_layer,
self.weights[0]))
        output_layer = self.activation('tang', np.dot(hidden_layer,
self.weights[1]))

        self.layers = [
            input_layer,
            hidden_layer,
            output_layer
        ]

        # last layer is an output
        return self.layers[2]

    def backward_pass(self, target_output, actual_output):

        err = (target_output - actual_output)

        # propagate gradients using chain rule
        for backward in range(2, 0, -1):
            err_delta = err * self.derivative_activation('tang',
self.layers[backward])

            # update weights using computed gradient
            self.weights[backward - 1] += self.learning_rate *
np.dot(self.layers[backward - 1].T, err_delta)

            # previous layer weights update
            err = np.dot(err_delta, self.weights[backward - 1].T)

    def train(self, x_values, target):
        self.output = self.feed_forward_pass(x_values)
        self.backward_pass(target, self.output)

    def predict(self, x_values):
        return self.feed_forward_pass(x_values)

X = np.array(([0, 0], [0, 1], [1, 0], [1, 1]), dtype=float)
y = np.array(([0], [1], [1], [0]), dtype=float)

network = OneHiddenLayerNetwork(learning_rate=0.1)
iterations = 5000

for i in range(iterations):
    network.train(X, y)
```

```
    ten = iterations // 10
    if i % ten == 0:
        print('-' * 10)
        print("Iteration number: " + str(i) + ' / ' +
              "Squared loss: " + str(np.mean(np.square(y -
network.output))))

for i in range(len(X)):
    print('-' * 10)
    print('Input value: ' + str(X[i]))
    print('Predicted target: ' + str(network.predict(X[i])))
    print('Actual target: ' + str(y[i]))
```

OUTPUT

```
(base) Vishruths-MacBook-Air-3:Desktop vishruthkhare$ python Experiment6_ML_Lab.py
----------
Iteration number: 0 / Squared loss: 0.4799042135840691
----------
Iteration number: 500 / Squared loss: 0.042950494381573286
----------
Iteration number: 1000 / Squared loss: 0.013212838667124893
----------
Iteration number: 1500 / Squared loss: 0.0070132325078057466
----------
Iteration number: 2000 / Squared loss: 0.004629345031713999
----------
Iteration number: 2500 / Squared loss: 0.003412401497355063
----------
Iteration number: 3000 / Squared loss: 0.0026854848967451344
----------
Iteration number: 3500 / Squared loss: 0.0022061432569868943
----------
Iteration number: 4000 / Squared loss: 0.0018679274404916754
----------
Iteration number: 4500 / Squared loss: 0.0016172802953130104
----------
Input value: [0. 0.]
Predicted target: [0.]
Actual target: [0.]
----------
Input value: [0. 1.]
Predicted target: [0.9470363]
Actual target: [1.]
----------
Input value: [1. 0.]
Predicted target: [0.94703627]
Actual target: [1.]
----------
Input value: [1. 1.]
Predicted target: [0.00936281]
Actual target: [0.]
(base) Vishruths-MacBook-Air-3:Desktop vishruthkhare$
```