Madhura Nagle
PD 15

Assignment 5

* Aim : Implement Neural Network for any application.

* objective: To study and implement neural Network for any application.

* Theory :

→ Neural network architecture

Neural networks are complex structures made of artificial neurons that can take in multiple inputs to produce a single output. This is the primary job of a Neural Network - to transform input into a meaningful output. Usually, a neural network consists of an input and output layer with one or multiple hidden layers within. In Neural networks all the neurons influence each other, and hence they are all connected. The network can acknowledge and observe every aspect of the dataset at hand and how the different parts of data may or may not relate to each other. This is how neural networks are capable of finding extremely complex patterns in vast volumes of data.

→ Deep learning frameworks.

Deep learning frameworks offer building blocks for designing, traning and validating deep neural networks, through a high level programming interface. This eliminates the need to manage packages and dependencies or build deep learning frameworks from source.

→ commonly used three activation functions

(1) Sigmoid function
(2) Hyperbolic Tangent
(3) Softmax function
(4) Soft sign function

* FAQs

Q Which algorithm is used to train the neural network?

→ (1) One dimensional optimization
(2) Multidimensional optimization
(3) Gradient descent
(4) Newton method
(5) Conjugate gradient.

Q How to decide number of hidden layers in neural network?

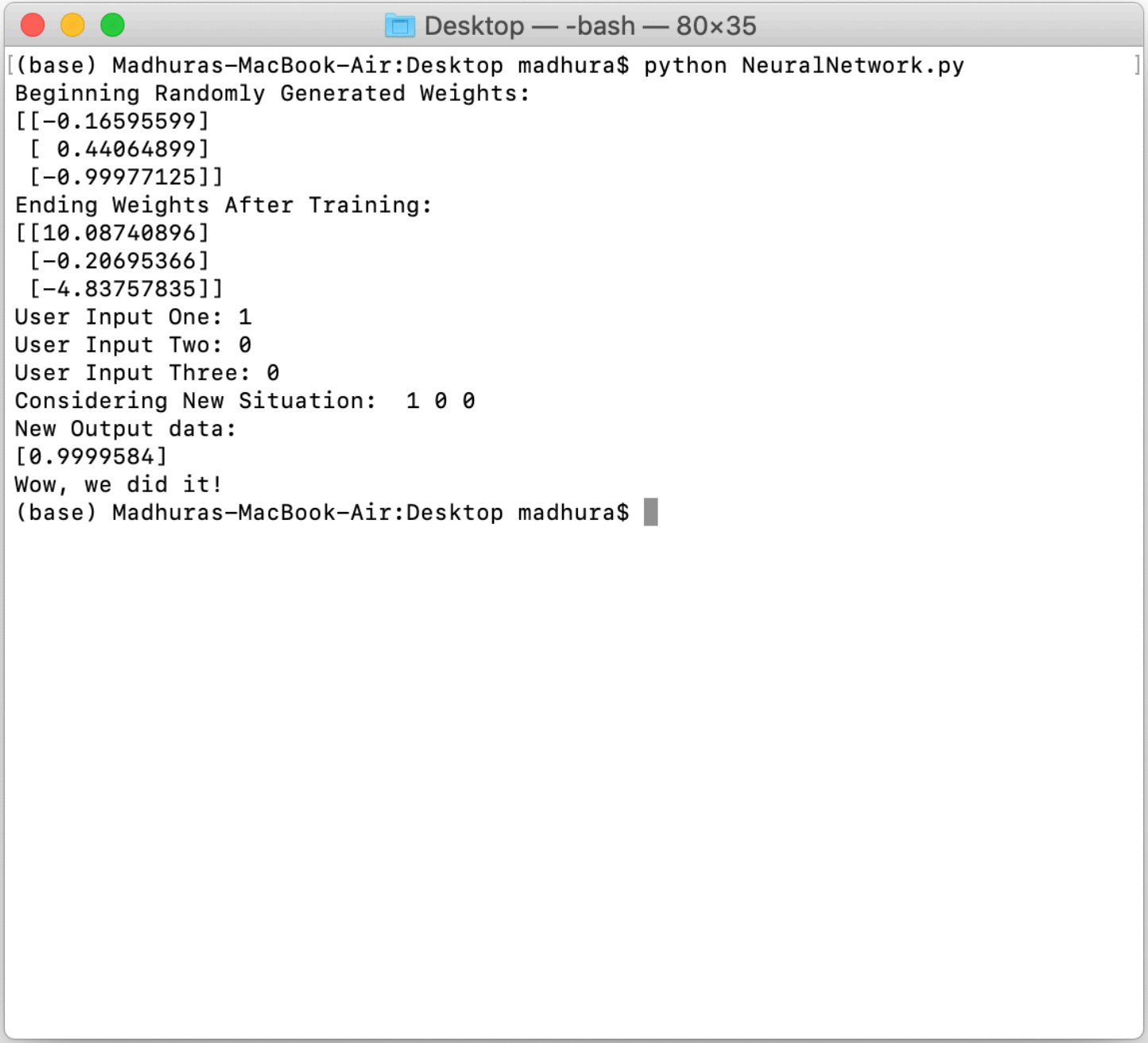
→ For most problems one could probably get decent performance (even without a second optimization step) by setting the hidden layer configuration using just two rules: (1) Number of hidden layers equals one and # (2) the number of neurons is that layer is the mean of the neurons in the input and output layers.

Q What is the drawback of deep learning?

→ It requires very large amount of data in order to perform better than other techniques. It is extremely expensive to train due to complex data models. Moreover deep learning requires expensive GPUs and hundreds of machines. It increases the cost to the users.

```
import numpy as np

class NeuralNetwork():

    def __init__(self):
        # seeding for random number generation
        np.random.seed(1)

        #converting weights to a 3 by 1 matrix with values from -1
to 1 and mean of 0
        self.synaptic_weights = 2 * np.random.random((3, 1)) - 1

    def sigmoid(self, x):
        #applying the sigmoid function
        return 1 / (1 + np.exp(-x))

    def sigmoid_derivative(self, x):
        #computing derivative to the Sigmoid function
        return x * (1 - x)

    def train(self, training_inputs, training_outputs,
training_iterations):

        #training the model to make accurate predictions while
adjusting weights continually
        for iteration in range(training_iterations):
            #siphon the training data via  the neuron
            output = self.think(training_inputs)

            #computing error rate for back-propagation
            error = training_outputs - output

            #performing weight adjustments
            adjustments = np.dot(training_inputs.T, error *
self.sigmoid_derivative(output))

            self.synaptic_weights += adjustments

    def think(self, inputs):
        #passing the inputs via the neuron to get output
        #converting values to floats

        inputs = inputs.astype(float)
        output = self.sigmoid(np.dot(inputs, self.synaptic_weights))
        return output


if __name__ == "__main__":

    #initializing the neuron class
    neural_network = NeuralNetwork()

    print("Beginning Randomly Generated Weights: ")
    print(neural_network.synaptic_weights)
```

```
    #training data consisting of 4 examples--3 input values and 1
output
    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

    #training taking place
    neural_network.train(training_inputs, training_outputs, 15000)

    print("Ending Weights After Training: ")
    print(neural_network.synaptic_weights)

    user_input_one = str(input("User Input One: "))
    user_input_two = str(input("User Input Two: "))
    user_input_three = str(input("User Input Three: "))

    print("Considering New Situation: ", user_input_one,
user_input_two, user_input_three)
    print("New Output data: ")
    print(neural_network.think(np.array([user_input_one,
user_input_two, user_input_three])))
    print("Wow, we did it!")
```


Desktop — -bash — 80×35
(base) Madhuras-MacBook-Air:Desktop madhura$ python NeuralNetwork.py
Beginning Randomly Generated Weights:
[[-0.16595599]
[0.44064899]
[-0.99977125]]
Ending Weights After Training:
[[10.08740896]
[-0.20695366]
[-4.83757835]]
User Input One: 1
User Input Two: 0
User Input Three: 0
Considering New Situation: 1 0 0
New Output data:
[0.9999584]
Wow, we did it!
(base) Madhuras-MacBook-Air:Desktop madhura$