Sharif University of Technology
Electrical Engineering School

Computational Intelligence CHW1

REGRESSION, MULTI LAYER PERCEPTRON

Armin Panjehpour
arminpp1379@gmail.com

Supervisor(s): Dr. Hajipour
Sharif University, Tehran, Iran

26/04/2022

Part.3 - MLP with BackPropagation Algorithm

Part.3.a - Derivatives of Activation Functions – Derivative of logistic function:

$$\sigma(\mathrm{x}) = \frac{1}{1+e^{-\mathrm{x}}} \qquad \frac{d(\sigma(\mathrm{x}))}{dx} = \frac{(e^{-\mathrm{x}})}{(1+e^{-\mathrm{x}})^2}$$

Figure 1: logistic function and its derivative

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

def sigmoid_prime(x):
    return np.exp(-x)/(1+np.exp(-x))**2
```

– Derivative of logistic function:

$$\frac{d}{dx}\tanh x = 1 - \tanh^2 x$$

Figure 2: tanh function and its derivative

```
def tanh(x):
    return np.tanh(x)

def tanh_prime(x):
    return (1-(np.tanh(x))**2)
```

Part.3.b - Create the Input and Output for XOR function Here's the input and corresponding outputs of XOR gate for MLP:

```
    X = np.array([[0,0],[0,1],[1,0],[1,1]])

    y = np.array([0,1,1,0])
```

Part.3.c - Updating the Weights Here's the completed code for updating the weights:(delta is error * activation prime value and layer is the input)

```
    self.weights[i] += learning_rate*np.dot(layer.T,delta);
```

Part.3.d - Run the Algorithm Here we run the algorithm with activation function, tanh, learning rate = 0.2 and 100000 epochs. Here are the outputs:

```
    [0 0] [0.00092276]
    [0 1] [0.99984882]
    [1 0] [0.99979804]
    [1 1] [0.00054346]
```

As you can see, by just setting a threshold to make the outputs binary, we have the desired outputs and our MLP has performed well!

Part.3.e - Error Vs Epoch Number Here, we plot the error values vs the epoch numbers:

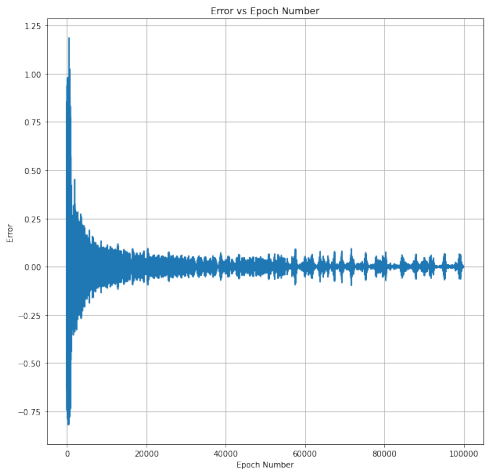

Figure 3: Error Vs Epoch Num, XOR, activationFunctino = tanh

As you can see, the error decreases as the epochs pass which is logical, because at first, our MLP will have random values for its weights and then it will move on to and try to decrease the output error.

Part.3.f - OR & AND Functions Here we try our model for OR & AND gates and here's the results:

```
[0 0] [0.01099317]
[0 1] [0.99994796]
[1 0] [0.99994872]
[1 1] [0.99998358]
```

As you can see, by just having a one in our inputs, the output will be one and our model is performed well for OR gate too. By just setting a threshold, we can make our outputs binary.

If we run the code several times, we always get this result close to one and close to zero which by setting a threshold they will turn to binary numbers. Here's the error vs epoch number for OR gate and it has a decreasing behaviour:

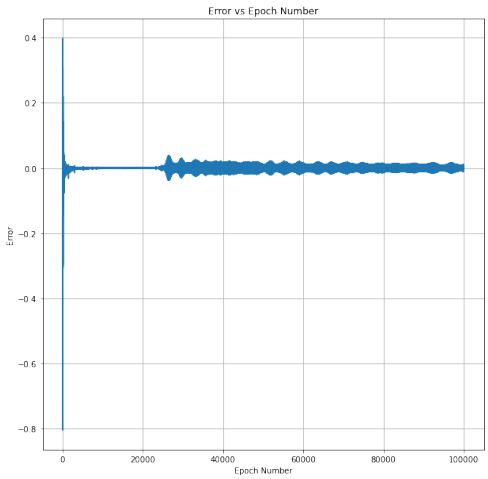

Figure 4: Error Vs Epoch Num, OR, activationFunctino = tanh

```
[0 0] [-0.00037174]
[0 1] [-0.00182712]
[1 0] [0.00085602]
[1 1] [0.99978485]
```

As you can see, when both of the inputs are one, the output will be one and in other conditions, the output would be zero. So the model describes the AND gate well too. By running the code several times, the model again perform the AND gate. The outputs would be different but again so much close to zero and one and by setting a threshold, they will be binary. Here's the error vs epoch number for AND gate and it has a decreasing behaviour:

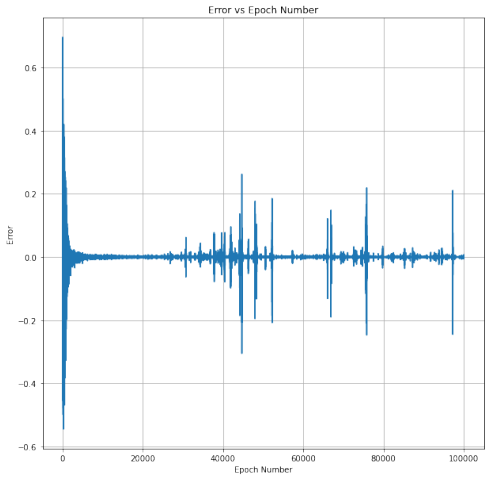

Figure 5: Error Vs Epoch Num, AND, activationFunctino = tanh

Part.3.g - XOR & OR & AND Functions with Logistic Activation Function Here we repeat previous parts using logistic activation function instead of tanh:

– XOR:

Again, XOR is described well:

```
[0 0] [0.01216228]
[0 1] [0.98583924]
[1 0] [0.98585893]
[1 1] [0.01541345]
```

the values are not that close to 0 and 1 as it was when we had tanh as our activation function but they are so much close. By running several times and having a threshold we always get the desired outputs for XOR. Here's the error vs epoch number plot:

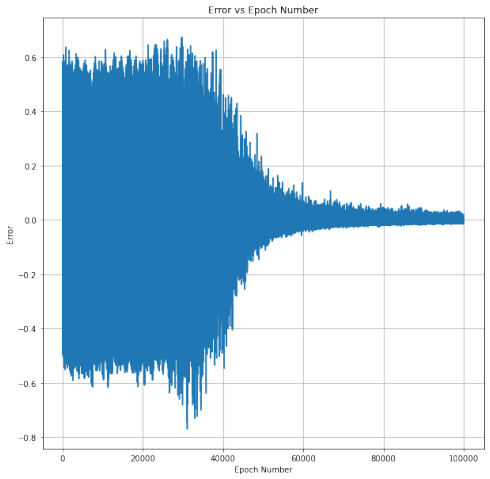

Figure 6: Error Vs Epoch Num, XOR, activationFunctino = logistic

As you can see, error has this decreasing behaviour again but it takes more time to decay.

– OR:

Again, OR is described well:

```
[0 0] [0.00086969]
[0 1] [0.9995786]
[1 0] [0.9995564]
[1 1] [0.99997104]
```

Here again our model performs well. By running several times and having a threshold we always get the desired outputs for OR gate. Here's the error vs epoch number plot:

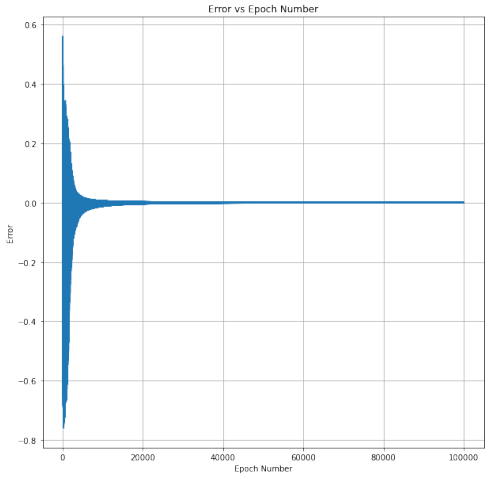

Figure 7: Error Vs Epoch Num, OR, activationFunctino = logistic

– AND:
for AND our model with default values won't perform that much good for all runs, but if we take the maximum output as one and the others as zero, it describes AND correctly. Here's the output for two times of running:

```
[0 0] [3.58367935e-11]
[0 1] [0.00190268]
[1 0] [0.00191638]
[1 1] [0.99623512]
```

```
[0 0] [3.97831596e-25]
[0 1] [0.23135754]
[1 0] [0.23295288]
[1 1] [0.5]
```

As it seems, sigmoid function doesn't perform as good as tanh for AND gate but still works and by several times of running, the maximum output is always created by the [1 1] input.

– In all previous parts, if we had put a threshold, and make the output binary, errors would have been zero from somewhere up to the end epoch.