Implementation of MLPs for XOR function with Single Hidden Layer

Lalit Singh, Oleg Borisov, Bishwa Thapa May 18, 2017

1 MLP Structure

This implementation of MLP for XOR function has following structure. It contains single hidden layer with a variable number of nodes. The input layer contains two input nodes which accepts input parameters and a single bias node. The output layer consists of single node which gives the estimation for XOR function. In the algorithm presented below, the calculations are shown for 3 nodes and a single bias node in hidden layer.

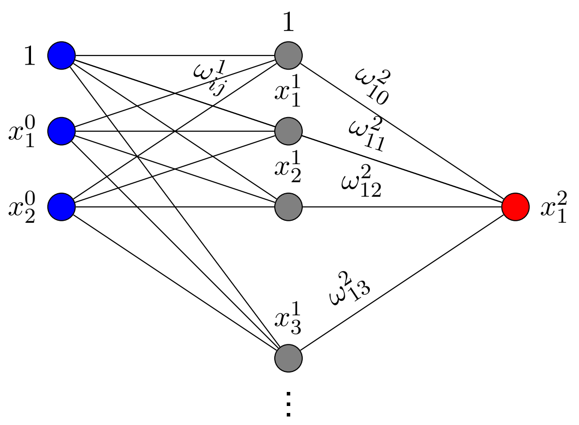

Figure 1: Schema of MLP for XOR function with a single Hidden layer.

2 Algorithm

Forward Pass:

The activation $x^1_{i=1,2,\dots,n}$ of layer 1 is given by

$$x_i^m = \sigma\left(\sum_{j=1}^{L^{m-1}} w_{ij}^m x_j^{m-1} + w_{i0}^m\right) \tag{1}$$

The above equation can be written in matrix form as below.

For Layer 1:

$$\begin{bmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \\ \dots \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{10}^1 & w_{11}^1 & w_{12}^1 \\ w_{20}^1 & w_{21}^1 & w_{22}^1 \\ w_{30}^1 & w_{31}^1 & w_{32}^1 \\ \dots & \dots & \dots \end{bmatrix} \times \begin{bmatrix} 1 \\ x_1^0 \\ x_2^0 \end{bmatrix}\right)$$

Similarly, for layer 2:

$$\begin{bmatrix} x_1^2 \end{bmatrix} = \begin{bmatrix} w_{10}^2 & w_{11}^2 & w_{12}^2 & w_{13}^2 & \vdots \end{bmatrix} \begin{bmatrix} 1 \\ x_1^1 \\ x_2^1 \\ x_3^1 \\ \dots \end{bmatrix}$$

The number of nodes in the hidden layer can be extended to i = n by adding the weights $w_{i=4,\dots,n}^{1,2}$ and the activations $x_{3,\dots,n}^1$ in the space represented by $\vdots$ and $\dots$ in above matrices.

The output of the **XOR** function is represented by the activation x_1^2 of node 1 of output layer.

$$i.e.\hat{y} = \mathcal{N}_\theta(u) = x_1^2 \tag{2}$$

Backward Pass:

Now,
The δ for i^{th} node of layer m is defined as

$$\delta_i^m = \frac{\partial L(\mathcal{N}_\theta(u), y)}{\partial a_i^m} \tag{3}$$

where L is the loss function and a_i^m represents the potential of the i^{th} node of layer m.

The DELTA for output layer is given by

$$\delta_1^2 = 2(\hat{y} - y_i) \tag{4}$$

And, the DELTA for the hidden layer can be computed by the formula given below:

$$\delta_i^m = \sigma'(a_i^m) \sum_{l=1,\dots L^{m+1}} \delta_l^{m+1} w_{li}^{m+1} \tag{5}$$

So the DELTA of hidden layer can be written in matrix form as given below:

$$\begin{bmatrix} \delta_0^1 \\ \delta_1^1 \\ \delta_2^1 \\ \delta_3^1 \end{bmatrix} = \sigma\left(\begin{bmatrix} a_0^1 & a_1^1 & a_2^1 & a_3^1 \end{bmatrix}\right) \times \begin{bmatrix} w_{10}^2 & \vdots \\ w_{11}^2 & \vdots \\ w_{12}^2 & \vdots \\ w_{13}^2 & \vdots \end{bmatrix} \times \begin{bmatrix} \delta_1^2 \\ \dots \end{bmatrix}$$

The gradiant descent can be calculated is given by

$$\frac{\partial L(\mathcal{N}_\theta(u), y)}{\partial w_{ij}^m} = \delta_i^m x_j^{m-1} \tag{6}$$

Also, in matrix form for layer 1:

$$\begin{bmatrix} \dot{w}_{10}^1 & \dot{w}_{20}^1 & \dot{w}_{30}^1 & \vdots \\ \dot{w}_{11}^1 & \dot{w}_{21}^1 & \dot{w}_{31}^1 & \vdots \\ \dot{w}_{12}^1 & \dot{w}_{22}^1 & \dot{w}_{32}^1 & \vdots \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^0 \\ x_2^0 \end{bmatrix} \times \begin{bmatrix} \delta_1^1 & \delta_2^1 & \delta_3^1 \end{bmatrix}$$

In matrix form for layer 2:

$$\begin{bmatrix} \dot{w}_{10}^2 & \vdots \\ \dot{w}_{11}^2 & \vdots \\ \dot{w}_{12}^2 & \vdots \\ \dot{w}_{13}^2 & \vdots \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^1 \\ x_2^1 \\ x_3^1 \end{bmatrix} \times \begin{bmatrix} \delta_1^2 \end{bmatrix}$$

Now the θ is updated by moving little bit in the direction of steepest decrease of the error:

$$\theta^{(n+1)} = \theta^{(n)} - \lambda \times \nabla R^{emp}(\mathcal{N}) \tag{7}$$

where λ is learning rate. In this implimentation the value used for learning rate is

$$\lambda = 0.1853$$

In matrix form the weights can be updated as below:

$$\begin{bmatrix} w_{10}^1 & w_{11}^1 & w_{12}^1 \\ w_{20}^1 & w_{21}^1 & w_{22}^1 \\ w_{30}^1 & w_{31}^1 & w_{32}^1 \\ \dots & \dots & \dots \end{bmatrix}^{(n+1)} = \begin{bmatrix} w_{10}^1 & w_{11}^1 & w_{12}^1 \\ w_{20}^1 & w_{21}^1 & w_{22}^1 \\ w_{30}^1 & w_{31}^1 & w_{32}^1 \\ \dots & \dots & \dots \end{bmatrix}^{(n)} - \lambda \times \left(\begin{bmatrix} w_{10}^1 & w_{11}^1 & w_{12}^1 \\ w_{20}^1 & w_{21}^1 & w_{22}^1 \\ w_{30}^1 & w_{31}^1 & w_{32}^1 \\ \dots & \dots & \dots \end{bmatrix}^{(n)} - \begin{bmatrix} \bar{\dot{w}}_{10}^1 & \bar{\dot{w}}_{20}^1 & \bar{\dot{w}}_{30}^1 & \vdots \\ \bar{\dot{w}}_{11}^1 & \bar{\dot{w}}_{21}^1 & \bar{\dot{w}}_{31}^1 & \vdots \\ \bar{\dot{w}}_{12}^1 & \bar{\dot{w}}_{22}^1 & \bar{\dot{w}}_{32}^1 & \vdots \end{bmatrix}^{(n)\mathbf{T}} \right)$$

$$\begin{bmatrix} w_{10}^2 & \vdots \\ w_{11}^2 & \vdots \\ w_{12}^2 & \vdots \\ w_{13}^2 & \vdots \end{bmatrix}^{(n+1)} = \begin{bmatrix} w_{10}^2 & \vdots \\ w_{11}^2 & \vdots \\ w_{12}^2 & \vdots \\ w_{13}^2 & \vdots \end{bmatrix}^{(n)} - \lambda \times \left(\begin{bmatrix} w_{10}^2 & \vdots \\ w_{11}^2 & \vdots \\ w_{12}^2 & \vdots \\ w_{13}^2 & \vdots \end{bmatrix}^{(n)} - \begin{bmatrix} \dot{w}_{10}^2 & \vdots \\ \dot{w}_{11}^2 & \vdots \\ \dot{w}_{12}^2 & \vdots \\ \dot{w}_{13}^2 & \vdots \end{bmatrix}^{(n)} \right)$$

3 Result

The implementation of the algorithm is run by initializing the wights of the link with some random values. Initially the neural networks gives wrong output for most of input. With the increase in number of epoch the error in output decreases. After certain number of epoch (*the exact value depends upon randomly initialized weight values*), the error value start oscillating around certain value(*again this values also depends upon the initial weights*). The algorithm is, then stopped, and the obtained weights are taken as optimal value for the given model.

The graph below shows the change in the error value of the with the increase in the number of epoch.

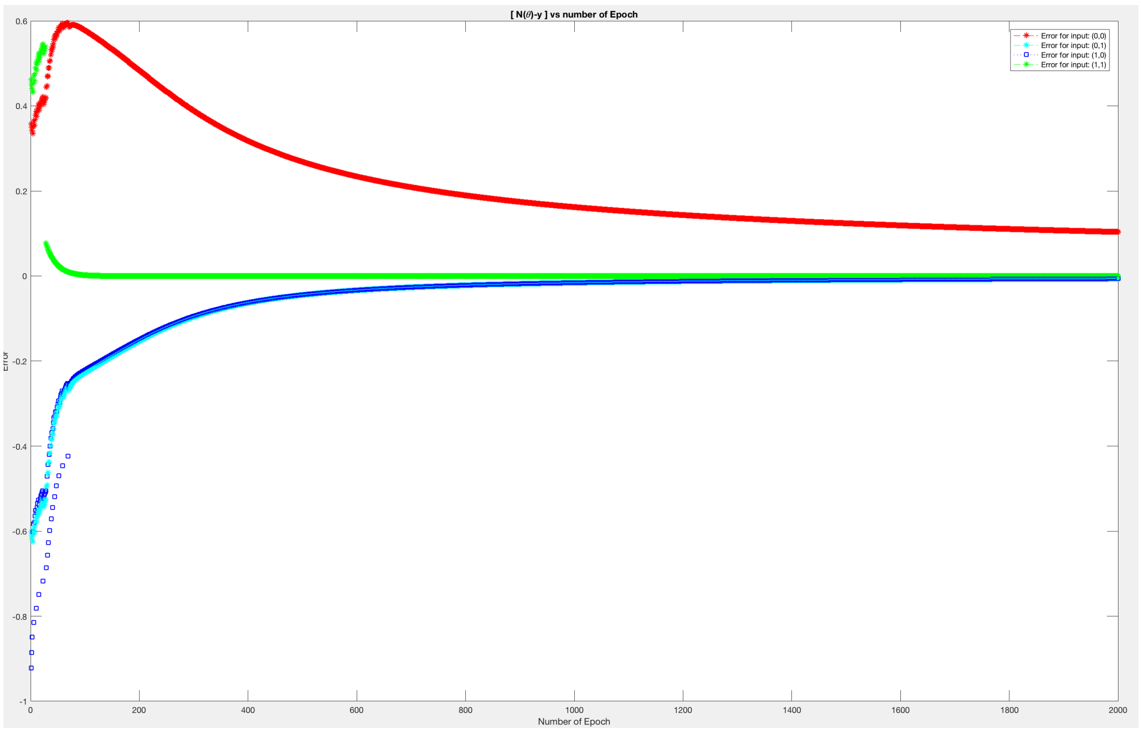

Figure 2: $\mathcal{N}_\theta(u) - y$ vs. number of epochs.The red colored stars represent error in output for input (0,0) Similarly blue squares are for error in output for input (1,0), greens are for error in output for input (1,1) and the cyan for error in output for input (0,1)

.

The optimal weights value for the given model after 2000 epoches are given below:

$$\begin{bmatrix} w^1_{10} & w^1_{11} & w^1_{12} \\ w^1_{20} & w^1_{21} & w^1_{22} \\ w^1_{30} & w^1_{31} & w^1_{32} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0.0019 & -0.0015 & -0.0015 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} w^2_{10} & w^2_{11} & w^2_{12} & w^2_{13} & \end{bmatrix} = 10^{-3} \begin{bmatrix} 0.0000 & 0 & -0.4532 & 0 \end{bmatrix}$$

References

[1] Herbert Jaeger's Machine Learning Lecture Notes for Spring 2017
`http://minds.jacobs-university.de/sites/default/files/uploads/teaching/lectureNotes/LN_ML4IMS.pdf`