X-NOR Implementation using Neural Network (Back-Prop. Algo.)

Vishal Kumar, MIT2019090

Dataset Used:

[[[0,0], [1]], [[0,1], [0]], [[1,0], [0]], [[1,1], [1]]]

Introduction:

An artificial neural network is a supervised learning algorithm which means that we provide it the input data containing the independent variables and the output data that contains the dependent variable. In the beginning, the ANN makes some random predictions, these predictions are compared with the correct output and the error (the difference between the predicted values and the actual values) is calculated. The function that finds the difference between the actual value and the propagated values is called the cost function. The cost here refers to the error. Our objective is to minimize the cost function. Training a neural network basically refers to minimizing the cost function.

A neural network executes in two phases: Feed Forward phase and Back Propagation phase.

Feed forward - Sum of dot product of input and weights passed through activation function to bound it in between 0 and 1. Activation function - Sigmoid Function.

Backward Propagation - Minimize the cost using Gradient descent method

Model Description:

I have written the model of network in the following manner: There is an input layer and output layer, we have to give the input dimension (no of features) and output dimension (no of classes) to the network. (x, y), where x & y are number of neurons in hidden layer Bias term is not considered in this model. seed for cross-validation is 1 and seed for NN weight initialization is also 1.

Activation Function: Sigmoid function

Optimizer Used: Stochastic Gradient Descent

In the output layer if the output of neuron is less than 0.5, we predict it as 0 or else 1.

Training:

The model I have used have 2 inputs, 2 neurons in hidden layer and 1 neuron in output layer.

Observation:

For learning rate = 0.3 and the iteration stops at MSE<0.01,

Input Layer Weights: [[0.9476900966595706, 8.035272863999296], [0.9475395915200718, 8.033708260024492]]

Hidden Layer Weights: [[37.87071385509751], [-30.24152635061117]]

Training ended at iteration= 232052

Inputs: [0, 0] => Expected = 1, Actual = 0.9784289018180675

Inputs: [0, 1] => Expected = 0, Actual = 0.0501482044876278

Inputs: [1, 0] => Expected = 0, Actual = 0.0502021599227385

Inputs: [1, 1] => Expected = 1, Actual = 0.9359205687112073

As per the output we can see that our model is working perfectly for X-NOR.

Epoch vs loss graph: