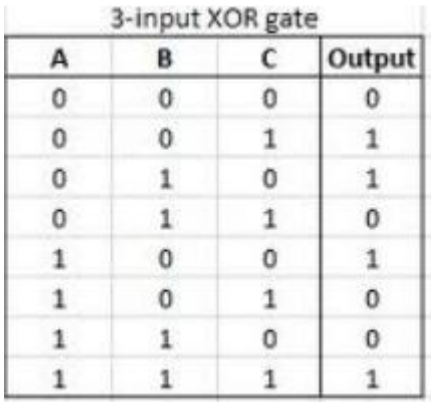
ML6 – FF Neural Network with Backpropagation BGD

Ohad Nir, Valentin Krasny

Initialization

Truth table initialization



3-input XOR gate

A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

1. The 3 first columns will be our inputs (X) and the last column will be the tags vector (t)
2. X and t are separated
3. X is transposed, and a first row of ones (for bias) is added.

1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1

Weights initialization:

1. Every node in the hidden layer has 4 weights – bias weight and 3 weights corresponding to every element in a given input vector (column of x). There are K nodes in the hidden layer so the dimensions of the hidden weights matrix is (KX4)

Hidden Layer Kode/weight	Z1	Z2	…	ZK
W0(bias)				
W1				
W2				
W3				

2. There is only one output node, that has K+1 weights – 1 for every node in the hidden layer connected to it, and 1 bias weight. dimensions: (1XK+1)

W0	W1	…	WK

The weights are filled with samples from gaussian distribution with 0 mean and 1 variance.

Node Values Calculating

Now, after we have weights and input data, we are able to calculate the values of the hidden layer nodes and the output node. The calculation is done simultaneously for all the inputs from the truth table (8 input vectors)

1. First, the activation for the hidden layer nodes is being calculated.

   $$a_j = (hidden - weights) \cdot X = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

   For every node, and for every input. Finally we get KX8 matrix (row for node, column for input)

2. the activation values are passed through sigmoid function to get the values of the hidden layer nodes:

   $$Z_j = h(a_j) = \sigma(a_j)$$

   Same dimensions as the activation matrix (every value just passed through sigmoid)

3. The Z matrix is biased with first row of ones, in order to calculate the output node Y in the next step. (It's now (K+1)X8)

4. The activation value for the output is being calculated.

   $$a_{out} = (output - weights) \cdot Z_{biased}$$

   For every input. Finally we get a 1X8 vector.
   (Multiplied 1X(K+1) with (K+1)X8)

5. The activation value for the output node is passed through the sigmoid to calculate the value for Y itself.

   $$Y = h(a_{out}) = \sigma(a_{out})$$

   Of course, same dimension as the activation vector – 1X8 for every input.

Batch Gradient Descent (BGD) by Backpropagation

1. Calculating the error δ_{out} for every input simultaneously by element wise multiplication:
$$\delta_{out} = Y * (1 - Y) * (Y - t^T)$$
Where * stands for element wise multiplication, and the relation:
$$h'(x) = \sigma'(x) = \sigma(x)[1 - \sigma(x)]$$
is used.
$$\delta_{out}\ dimensions\ are\ 1X8$$

2. Calculating the errors of the hidden layer nodes δ_{hidden}
$$\delta_{hidden} = [(output - w_{unbiased})^T \cdot \delta_{out}] * [Z_{unbiased} * (1 - Z_{unbiased})]$$
again, * is for element wise multiplication.
The resulting value δ_{hidden} is with dimensions of KX8, every row for hidden layer node, and every column for every input.

Input 1	…	Input 8
$\delta_{out1} w_1 z_{11}(1 - z_{11})$	…	$\delta_{out8} w_1 z_{18}(1 - z_{18})$
. . .	. . .	. . .
$\delta_{out1} w_K z_{K1}(1 - z_{K1})$	…	$\delta_{out8} w_K z_{K8}(1 - z_{K8})$

3. The new weights are being calculated (this is actually the BGD step):
$$(new - output - w - biased) = (old - output - w - biased) - \eta * \delta_{out} \cdot Z_{biased}^T$$
$$(new - hidden - w - biased) = (old - hidden - w - biased) - \eta * \delta_{hidden} \cdot X_{biased}^T$$
Dimensions are 1X(K+1) for output W and KX4 for the hidden W (just as before)

Main:

1. Initialize (Truth table and weights)
2. Calculate hidden layer nodes & output node values
3. (Optional): Calculate the loss (MSE between Y and t)
4. Make BGD step and update the weights.
5. Back to 2 (2000 iterations)