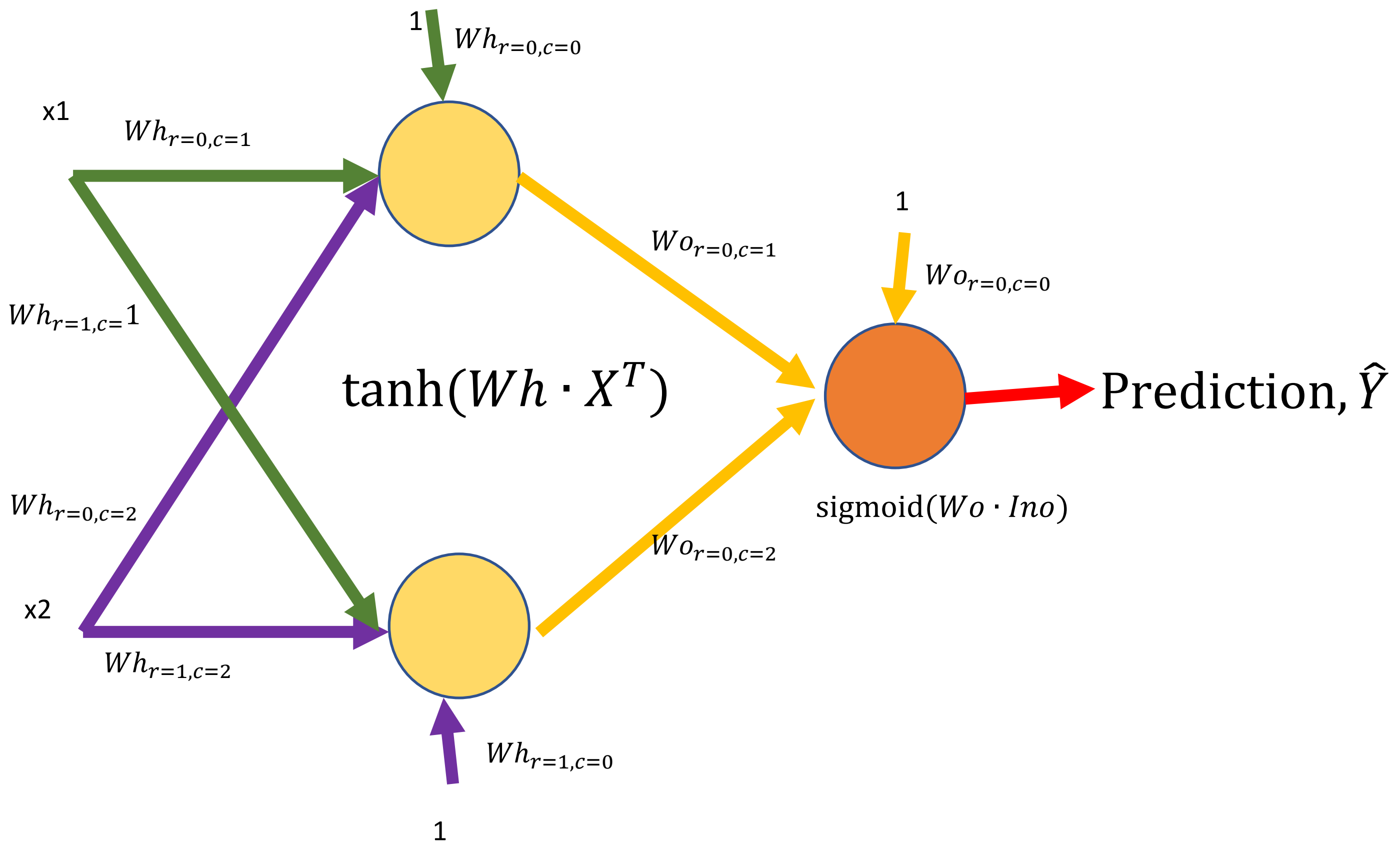
Abstract

Artificial neural networks (ANNs) are computing systems vaguely inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

In this assignment, we learn to model an XOR gate with a neural network. Acceptable results were obtained with an alpha of 0.5, an iteration count of 1000 and with the XOR table as the training set.

Section 1 ANN for XOR



Section 1 ANN for XOR(continued)

$$Let\ X\ be\ a\ single\ sample\ input.$$

$$X = \begin{bmatrix} 1 & x1 & x2 \end{bmatrix}$$

$$Let\ Wh\ be\ \ the\ weights\ of\ the\ hidden\ layer.$$

$$Wh\ be\ a\ 2x3\ Matrix, initialised\ as\ \begin{bmatrix} 0 & rand & rand \\ 0 & rand & rand \end{bmatrix}$$

$$The\ 2\ rows\ of\ Wh\ specifies\ two\ neurons, and\ the\ columns$$

$$specify\ the\ number\ of\ inputs\ along\ with\ a\ bias\ for\ each\ neuron.$$

$$\mathrm{Oh} = \tanh(WhX^T)\ will\ give\ a\ 2x1\ matrix, where\ each\ value\ is\ the\ output\ of$$

$$the\ neurons\ from\ the\ hidden\ layer.$$

$$\tanh\ is\ a\ non-linear\ activation\ function\ which\ will\ produce$$

$$\text{outputs in } [-1,1]$$

$$A\ 1\ is\ appended\ to\ the\ top\ of\ Oh\ to\ provide\ as\ the\ bias\ for\ the\ neuron$$

$$in\ the\ output\ layer.$$

$$Ino = \begin{bmatrix} 1 \\ Oh \end{bmatrix}$$

$$Ino, is\ the\ input\ to\ the\ neuron\ in\ the\ output\ layer.$$

$$The\ output\ of\ the\ neural\ network\ will\ be,$$

$$Output = \ sigmoid(Wo \cdot Ino)$$

$$Where\ Wo\ is\ 1x3\ matrix, initialised\ as\ \begin{bmatrix} 0 & rand & rand \end{bmatrix}$$

$$The\ Output\ will\ be\ a\ value\ in\ [0,1]$$

$$From\ Wh\ and\ Wo, we\ can\ see\ that\ there\ will\ be\ a\ total\ of\ 9\ parameters$$

Section 2 Back Propagation

$$Error\ Function = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2$$

Where m is the number of training examples.

To update the weights of the neural networks, we need to compute the derivatives w.r.t the error function, then update the weights with a factor of alpha.

The equation to update the weights for the output layer,

$$Wo = Wo + \ \alpha \cdot (\delta o \cdot Ino^T)/4$$

Where δo is,

$$\delta o = (Y^T - Oo)^\circ Oo^\circ(1 - Oo)$$

$$^\circ : the\ element\ wise\ product$$

$$\delta o\ will\ be\ a\ (1xm)\ matrix, as\ there\ is\ one\ neuron\ in\ the\ output\ layer\ and\ m\ examples$$

To update the weights for the hidden layer, we make use of $\delta o\ again$, to first calculate,

$$\delta h = (\overline{Wo}^T \cdot \delta o)^\circ(1 - \mathrm{Oh}^\circ\mathrm{Oh})$$

Where δh is a 2xm matrix, as there are 2 neurons in the hidden layer and m examples.

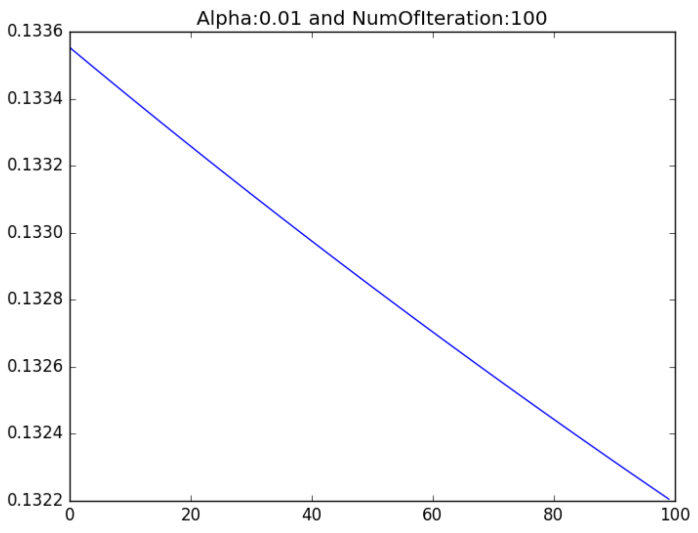
Update the weights in the hidden layer with,

$$Wh = Wh + \ \alpha \cdot (\delta o \cdot X)/4$$

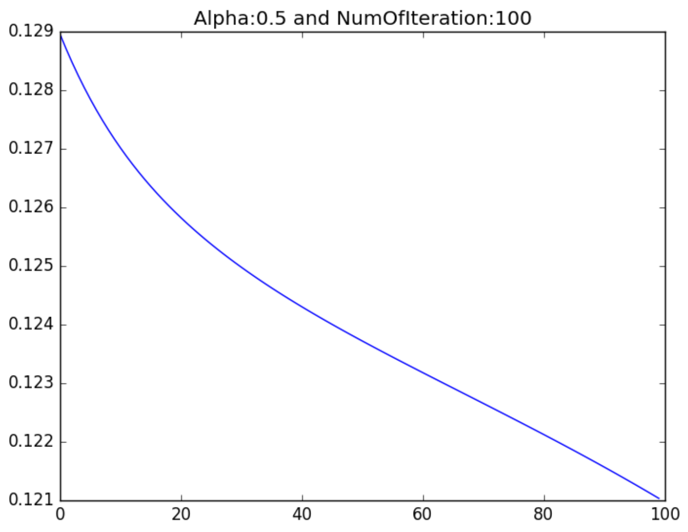
Section 3 Experimental Result

Each graph below is accompanied by a table of results produced by weights after training.

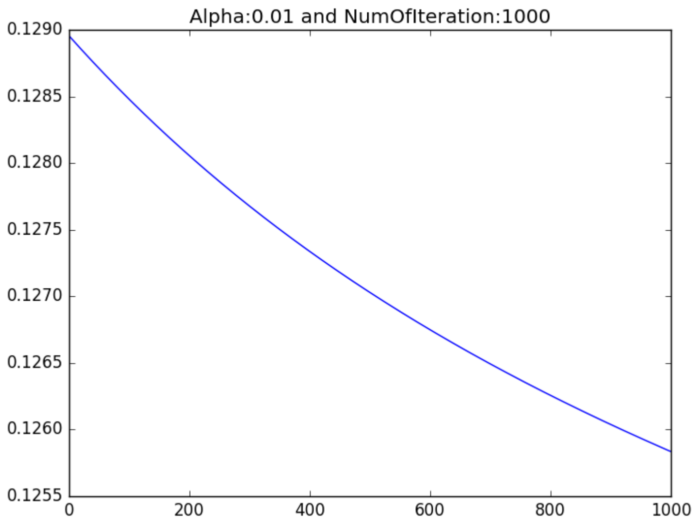
X-axis of the graph represents the iteration, and Y-axis represents the error.



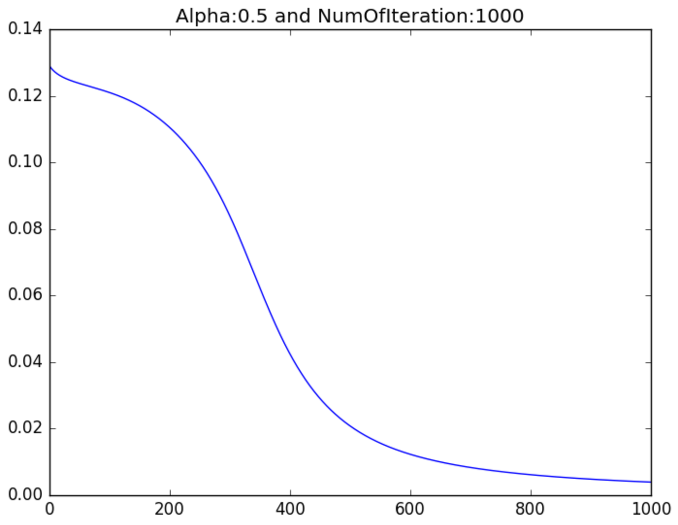
0	0	0.330358
0	1	0.528235
1	0	0.220033
1	1	0.342779



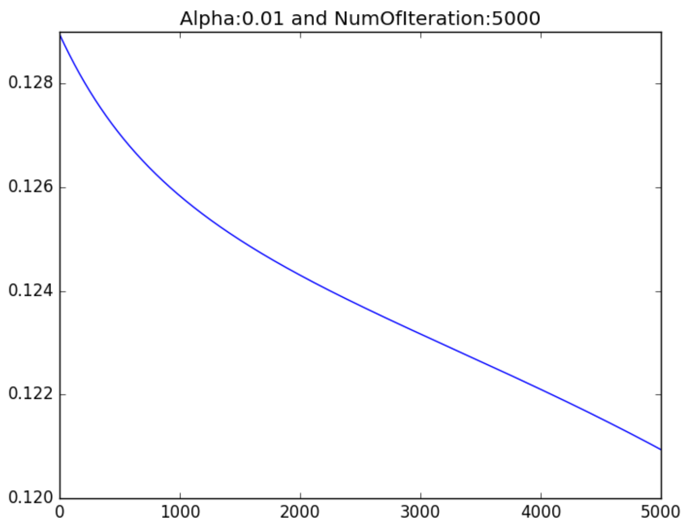
0	0	0.477898
0	1	0.555161
1	0	0.432006
1	1	0.467862



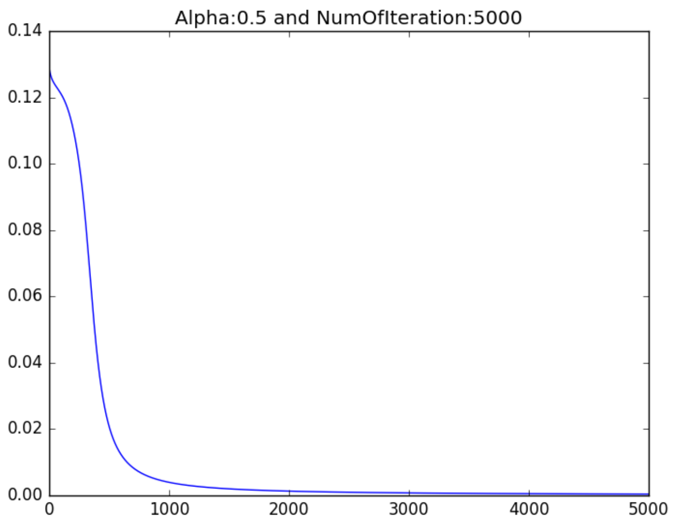
0	0	0.482905
0	1	0.556237
1	0	0.425257
1	1	0.496214



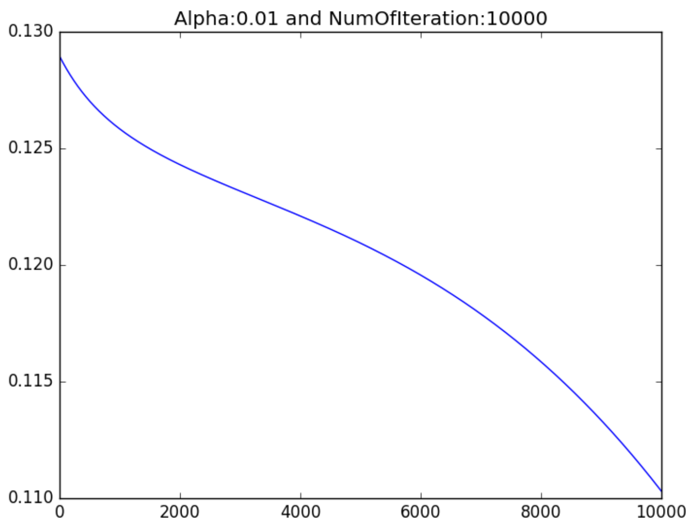
0	0	0.080393
0	1	0.900501
1	0	0.898655
1	1	0.070495



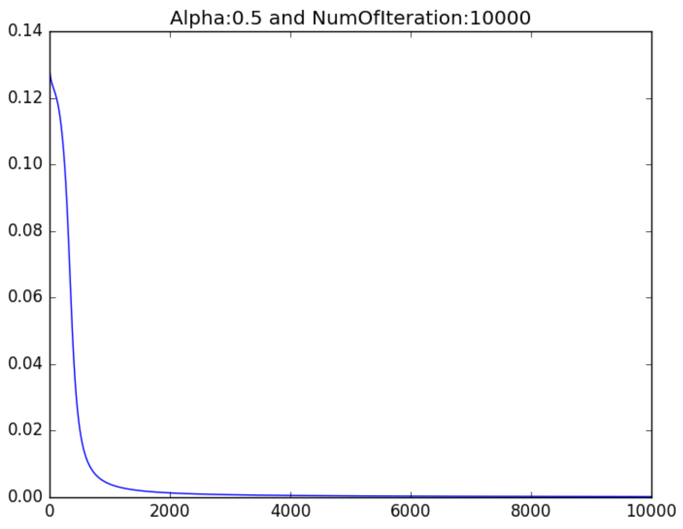
0	0	0.477637
0	1	0.555483
1	0	0.431621
1	1	0.467674



0	0	0.02523
0	1	0.967721
1	0	0.967566
1	1	0.022289



0	0	0.439671
0	1	0.601092
1	0	0.418127
1	1	0.437504



0	0	0.016899
0	1	0.978226
1	0	0.978153
1	1	0.014972

From the graphs above, we can see that a learning rate of 0.5 constantly outperforms a learning rate of 0.1. Although the error of 0.1 decreases as we increase the number of iterations, it still fails to converge. On the contrary, when using an alpha of 0.5, we see acceptable results as early as 1000 iterations. Thus the learning rate places an important role in neural networks and machine learning as a whole.