Weekly Updates Report

Satwik Dudeja

Delhi Technological University

Project Trainee, Centre for Artificial Intelligence and Robotics, DRDO

Introduction

Neural Networks

Artificial Neural Networks are computing systems that are meant to simulate the working of a human brain. With the help of experience and data, a neural network improves itself using artificial neurons. ANN consist of elementary units called as neurons which takes one or more inputs and produces an output. At each node, the following computations are carried out.

$$Z^{[i]} = W^{[i]} * A^{[i-1]} + b^{[i]}$$
$$A^{[i]} = f\left(Z^{[i]}\right)$$

$$Where - W^{[i]} - Weights\ of\ the\ connection$$
$$A^{[i-1]} - Output\ from\ the\ previous\ layer$$
$$b^{[i]} - bias\ of\ the\ connection$$
$$f(x) - Non\ linear\ activation\ function$$

A neural network comprises of input, hidden and output layers. A layer is a group of parallel neurons without any interaction between them. Neurons in the input layer are connected to the layer of hidden units, which are then connected to neurons in the output layer.

Gradient descent based back propagation technique is used to tune the weights and the bias to improve the accuracy of the model.

Backpropagation and Gradient Descent

Backpropagation is the algorithm in supervised learning of neural networks for backward propagation of errors. After every feed-forward epoch, the calculation of the gradient is done in a reverse manner, where weights of the final layer are calculated and used for subsequent calculations for the layers moving backwards. This enables efficient calculation of gradient, and optimisation of weights and biases.

The equations used in the backpropagation algorithm for gradient and and loss computations is listed -

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

Week 1

Comparison of Neural Network Implementation of a 10 input AND Gate using MATLAB and Keras API

1. AND Gates

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a certain logic.

A high output results only if all the inputs to the AND gate are high. If none or not all inputs to the AND gate are high, low output results. Basic truth table –

A	*B*	*Output*
1	1	1
1	0	0
0	1	0
0	0	0

Because the Boolean expression for the logic AND function is defined as (.), which is a binary operation, AND gates can be cascaded together to form any number of individual inputs.

2. Dataset

Both models on MATLAB and Python use the same dataset for a 10 input AND gate truth table.

The dataset is generated on Python using the *ttg* package, and the training, validation and testing splits are made the exact same for both models. Dataset dimensions - 1024

3. Model Parameters and Training Configuration

The Sequential Model from the Keras API of Pythons Tensorflow library is used. For MATLAB, the Neural Net Pattern Recognition Application is used.

Adam Optimizer in Keras and *learnscg* in MATLAB work on the same principle of Stochastic Gradient Descent with Adaptive Learning.

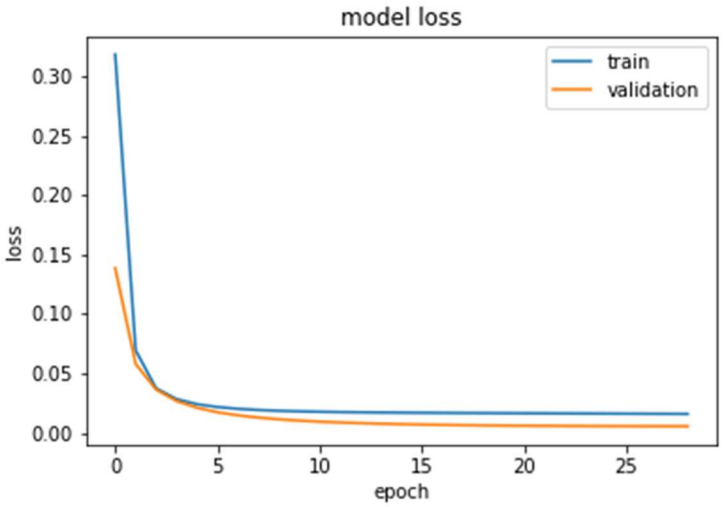
No shuffling of dataset ensures same weight initialization for both models.

The following settings were kept the same -

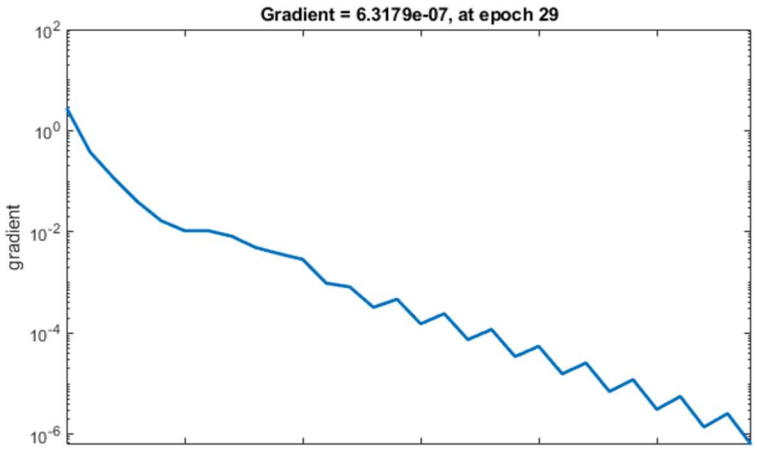
Parameter	*Keras*	*MATLAB*
Train Split	70%	70%
Validation Split	15%	15%
Test Split	15%	15%
Network Type	Feed-Forward Backpropagation	Feed-Forward Backpropagation
Number of hidden layers	1	1
Number of neurons in hidden layer	100	100
1st Activation Function	*tanh*	*tanh*
2nd Activation Function	*sigmoid*	*sigmoid*
Optimizer	*Adam*	*learnscg*
Loss Function	*binary_crossentropy*	*crossentropy*
Max. Epochs	1000	1000
System Architecture	x64 Windows	x64 Windows
Shuffle	False	False

4. Computations and Results

Keras



MATLAB



Confusion Matrix

The shown figures are the visualisation of the gradient descent of both algorithms. Adam Optimizer works on Stochastic Gradient Descent whereas the MATLAB model is trained on Scaled Conjugate Gradient which performs faster. Also shown, is the confusion matrix showing 100% accuracy.

5. Conclusion

Both MATLAB and Keras are able to classify the AND Gate output into the linearly separable classes with perfect accuracy (100% in MATLAB and 99.51% in Keras). The gradient descent is different for both models due to minor differences in training methods, but the outcome is correct for both.