Double Moon Classifier

Aditya N. Govardhan

Abstract: In this report, the problem of double moon classification is analysed using neural network approach. A uniformly distributed dataset is generated using random number generation functions of MATLAB, which is also used for implementing a multilayer neural network based on backpropagation algorithm. It has been found that a linear classifier is useful only till the two half moons are linearly separable, otherwise the error in classification increases. Increasing the number of layers and number of neurons in each layer can result in desired output, however this can lead to overfitting issues. Thus, it can be concluded that a single layer of hidden neurons with a sufficiently fast learning rate and backpropagation training algorithm should suffice to solve double moon classification problem. This report aims to put light on the preliminary design aspects to be taken into consideration while designing neural networks. These aspects can be applied to neural networks of any scale and form the basis of solving problems using deep learning techniques.

***Index Terms*—backpropagation, classification, gradient descent, overfitting, stopping criteria**

I. Introduction

Pattern recognition is one of the important classes of supervised learning in which data is classified into desired set of labels or groups. Double moon classification is one of the representative problems for pattern recognition in which data points are situated in space as shown in Fig. 1. As the length d between half moons is varied, the data points can be made linearly separable or inseparable. As long as the two areas in the figure don't merge for a given value of d, they can be made nonlinearly classifiable as well.

Artificial neural networks are one of the approaches for implementing learning algorithms. This paper analyses the aspects of implementing neural network based learning algorithm for solving double moon classification problem.

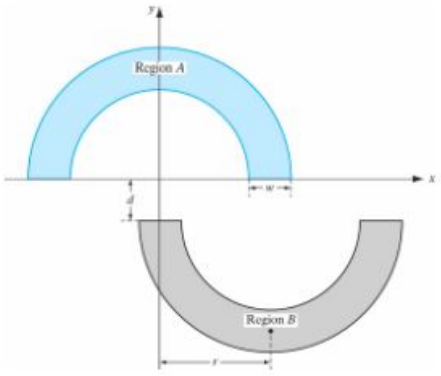

Fig. 1. Double Moon Classification Space

Aditya N. Govardhan is with the Arizona State University, Tempe, AZ 85281 USA (e-mail: Aditya.Govardhan@asu.edu).

II. Method

A. Data Generation

In this setup, 1400 data points are generated where each data point are (x, y) cartesian coordinates of the point. Out of these 1400 points, 1000 are used for training the neural network, 200 are used for validation of the model and the remaining 200 are used for testing the model.

Any method can be used to generate these data points. In this report, MATLAB is used to create uniformly distributed set of data points. Initially, random points are generated from a square bordering the double moon figure completely. Then these points are tested if they lie within the boundaries of the two half moon figures. If they do, corresponding targets are labelled 0 or 1 if they lie in lower half moon or upper half moon respectively. This process is repeated until we collect a set of 700 points lying in upper half moon and lower half moon each.

The distance d between the two half moons can be varied and linearly separable (see fig. 2(a)) and inseparable cases can be obtained. In this report, the classification problem is analyzed for d = -6 (linearly inseparable, see fig. 2(b)).

B. Neural Network Design

Given the power of neural networks to draw inferences which are not apparent to humans, double moon classification is an inherently simple problem for a neural network. Hence, a "shallow" neural network is used. To design the neural network, MATLAB neural network toolbox is used because it simplifies prototyping process.

The neural network consists of two layers, hidden layer and output layer, with one neuron each. Multiple neurons are avoided in the hidden layer in an effort to avoid *overfitting*. The hidden neuron uses tansigmoid (MATLAB function 'tansig') transfer function while the output neuron uses linear (MATLAB function 'purelin') transfer function.

These transfer functions are chosen in order to avoid abrupt outputs since discontinuous functions like hard limiting function have binary outputs and although the expected output is binary, given the training process explained in the following section, the process can fail to converge because of major change in the output for even a small change in input and weights. fig. 3. shows a structural view of the designed neural network.

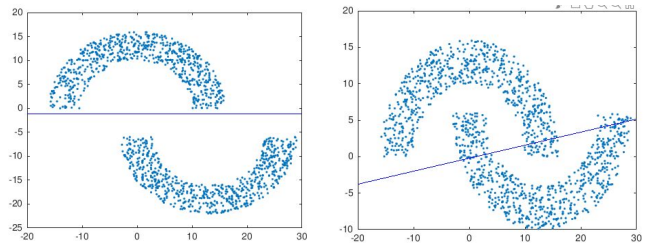

Fig. 2. (a) Linearly separable (b) Linearly inseparable

C. Training and Stopping Criteria

Obtained neural network design is trained using the data generated. The weights are initialized with values between -0.2 and 0.2. The training function used is gradient descent with adaptive learning rate backpropagation (MATLAB function 'traingda'), with an initial learning rate of 0.01. The stopping criteria for the training process is defined in two ways, whichever occurs first. First, if training and validation is completed for 1000 epochs, the process is stopped. However, if during validation, the network doesn't classify correctly for 10 data points then the training process is stopped. This training process is iterated 20 times in order to obtain average values for the process.

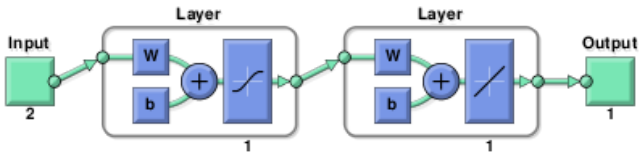

Fig. 3. Neural Network Design

III. Results

After carrying out the iterative process, average values of system parameters are obtained. Fig. 4. displays the confusion matrix for one of the iterations of testing data. As shown in Fig. 5, the training stops after 77 epochs since stopping criteria for 10 validation errors is reached. However the optimal (least) error is obtained at 67th epoch with value 0.07069. In the remaining iterations, the number of validation errors increase and eventually the stopping criterion is met. The average error over 20 iterations is found to be 0.0776 and average number of epochs required before stopping criterion is met is 76 epochs.

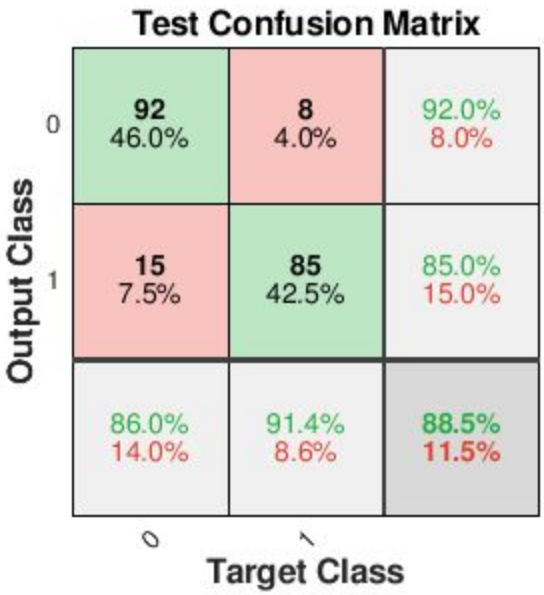

Fig. 4. Test Data Confusion Matrix

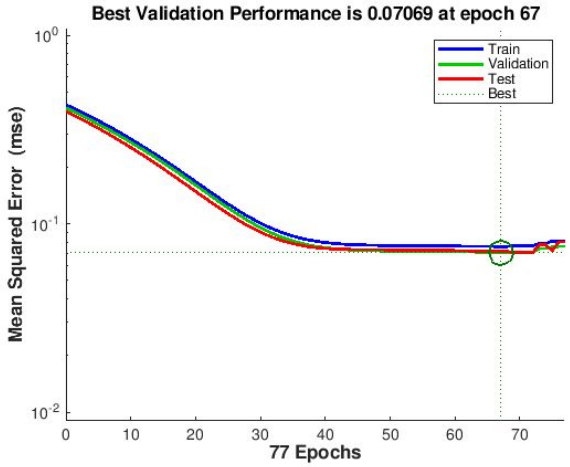

Fig. 5. Training Error and Testing Error

IV. Conclusion

Initial weights, order of data, learning rate, number of layers, neurons in each layer and the learning algorithm chosen, all influence the number of epochs required for the neural network to classify both half moon. When the learning rate is increased, convergence is approached faster however there is a chance of overshooting the minima. Increase in number of layers and neurons can classify faster however can lead to overfitting. The training algorithm used is chosen as per the class of problem and determines the speed of convergence. Stopping criteria is used to keep validation error under check as well as the maximum number of epochs performed.

References

[1] MATLAB Neural Network Toolbox Documentation