Priyaman Venu Saranyan (50134222) Tanvi Bhavsar (50133125) Karamveer Choudhary (50134943)

Handwritten Digits Classifier using a Multilayer Perceptron Network

1 ABSTRACT

The aim of this report is to demonstrate the ability of a Multi-Layer Perceptron Network in classifying Handwritten Digits. In this report, we use the MNIST dataset to benchmark the performance of the classifier. We also report the variation in classification results with respect to the regularization Hyper-parameter (ƛ) and the number of hidden layer units.

2 INTRODUCTION

Artificial Neural Networks have come into extensive use nowadays due to the increased processing power available. In the following section, we describe the basic structure of the neural network and its working.

2.1 DESIGN

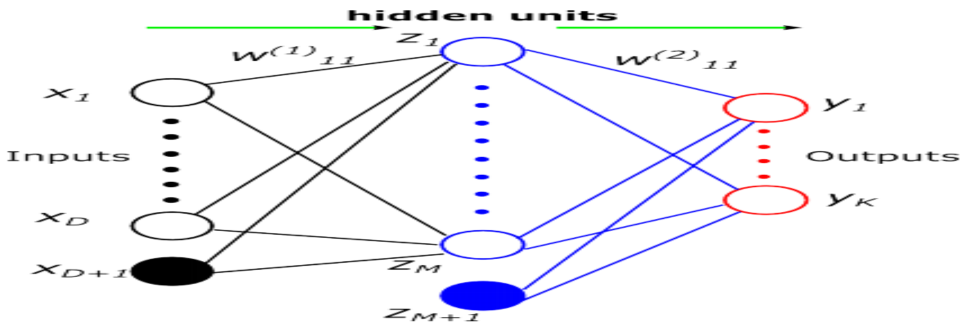


Figure 1: Neural network

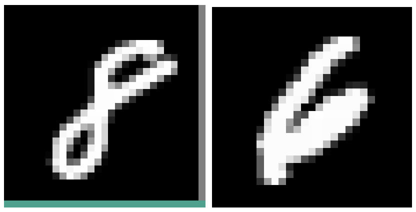
Figure one describes the structure of a neural network. It consist of 2 layers the input layer and the hidden layer of sizes D and respectively. We add an extra node at each layer which is referred to as the 'Bias'. The input feature vector 'X' that represent the 'D' dimensional input feature that we need to classify as one of the 'k-classes' in the output layer. The input features are 28x28 grayscale images of handwritten digits from the MNIST database. The image-matrices are reshaped from 28x28 to form 1x784 vectors which is supplied as input to the classifier.

The output (O) of the network is classified as W^TX at each layer where W is the weights of the perceptrons at each of the layers. This is referred to as Feed-Forward. During the learning phase, the error in output is compared to the true labels and weights of the nodes are adjusted as required.This is called 'backpropagation'.

2.2 Feature Reduction

Since, the input images might have a lot of redundant data from which the classifier cannot gain any useful information and the existence of such information would only make the training process slower, we prune the training data to remove redundant data.

For example,

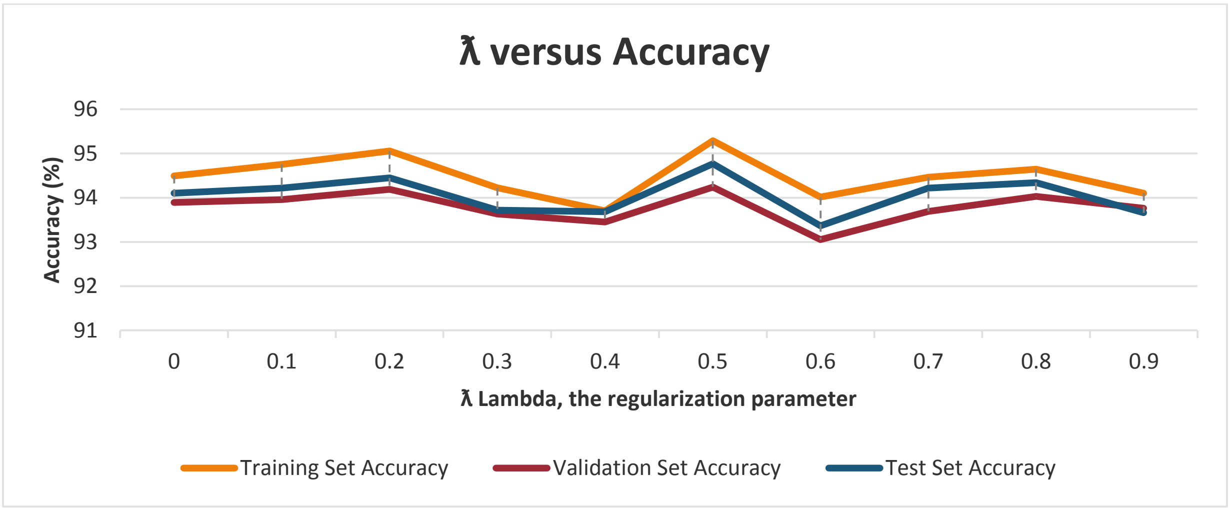


These are highly enlarged images of an eight and a six from the MNIST database. As you can see the boundaries of the image is black and contain no useful information to classify the images. Hence we present a technique to remove this redundant data.

We stack all the input images from the MNIST database into a NxF matrix where N is the number of input images and F is the size of each images (eg: 5000x784). Then we see remove those columns that are the identical in the matrix as they are the same for all inputs and don't give any useful information to the classifier.

2.3 Tuning the Regularization Hyper-Parameter ƛ

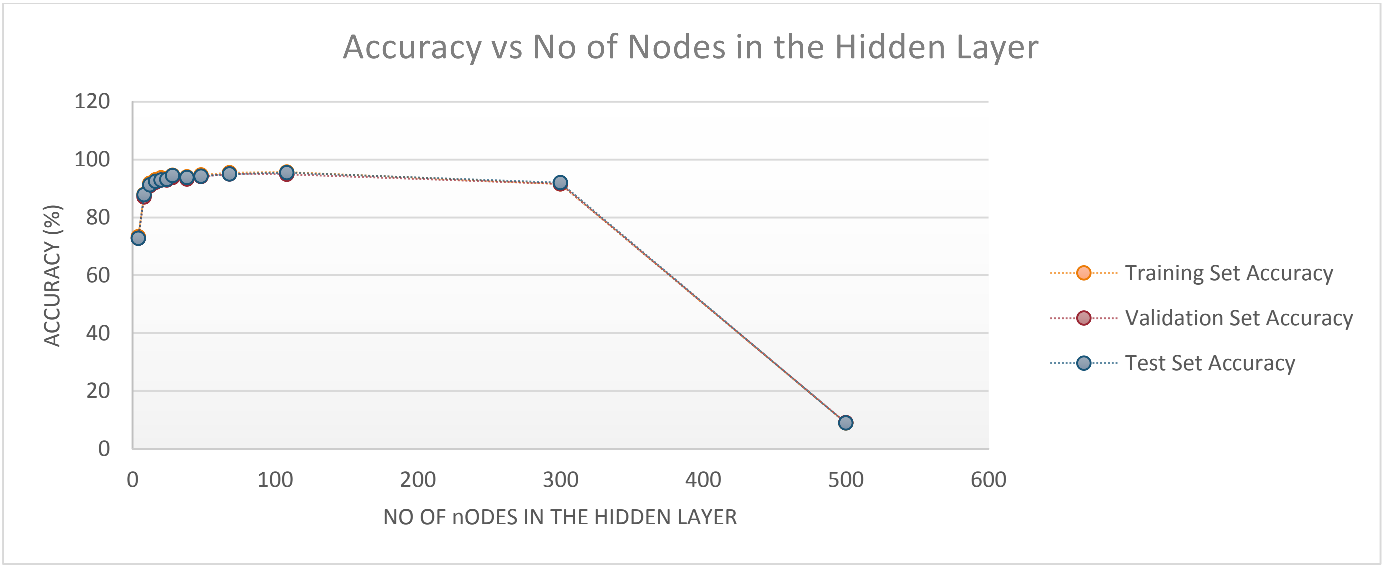
The Hyper Parameter ƛ is used to obtain good generalization and regularization and thereby avoid the problem of over-fitting. In this section, we report the classification results with respect to different values of Lambda. The graph below was plotted for 50 hidden nodes.



From the above graph, we can see that the efficiency peaks at a Lambda value of 0.5 at 94.77% percent on the test data. So, empirically we determine the best value of lambda.

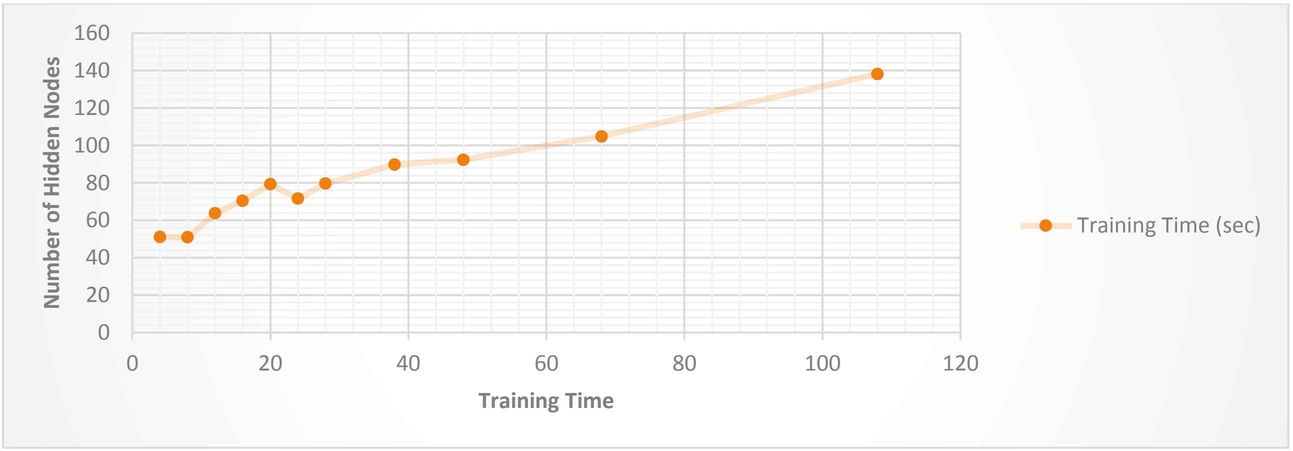
2.4 Varying the Number of Nodes in the Hidden Layer

2.4.1 Performance versus Number of Nodes in the Hidden Layer



From the results of the graph, we can see that increasing the number of nodes in the hidden layer increases the accuracy up to a certain point after which the accuracy starts dropping.

2.4.2 Time taken versus Number of Hidden nodes



From the above graph, we can see that the training time increases as the number of nodes in the hidden layer increases. More nodes implies more weights to be updated, so the training time increases.

2.5 Conclusion

Various aspects of the Multi-layer Perceptron Network such as the Hyper-parameter for Regularization, number of nodes in the hidden layer and training time were analyzed. From our results we can infer that the accuracy peaks for a certain value of Lambda. The training time increases as the number of nodes in the hidden layer increases and that the performance increases as the number of hidden nodes increases until a point after which it drops, so we choose the value at which it peaks as the value of number of hidden nodes, ie: empirically from test data.