ARTIFICIAL NEURAL NETWORK

ASHNA SWAIKA	***MUDIT CHATURVEDI***	***ROHAN SACHAN***
(2018A7PS0027H)	(2018A7PS0248H)	(2018B3A70992H)

INTRODUCTION

Artificial Neural Networks are models inspired by biological neural networks that are found in animal brains. It is based on a network of nodes (neurons) which are basically functions that take in some input and give out an output. They fire an output when according to the nature of the function and these functions are known as activation functions. There is a feedback mechanism called back propagation which adjusts the parameters in the networks for the best possible output and least possible loss, which is calculated according to the actual output and predicted output given by the model. This basic machine learning paradigm is called supervised learning. There are various tasks which can be accomplished through this paradigm and in this particular assignment, we perform a multi class classification using an Artificial Neural Network on the given dataset.

MODEL DESCRIPTION AND IMPLEMENTATION

The Model uses sigmoid function as an activation function and categorical cross entropy as the loss function. We have implemented it as a class named ANN where forward propagation and backward propagation functions have been made alongside train and test functions. The activation and loss functions are defined separately and not in the model. The hyper-parameters such as the number of nodes per layer and weights are attributes of the class itself. The single hidden layer neural network and double hidden layer neural network have been implemented as separate classes namely – ANN1 and ANN2. The dataset is divided into train and test for Hold-out cross validation using 70:30 split. For pre-processing we use min-max normalization on the dataset. We have used Batch gradient Descent for training the model. Best weights of the model are stored. The output is converted into a one hot vector for training according to the number of classes that are present which is 10. The training has been done using Google Colaboratory for better speed.

INPUT :

1. *Input matrix, '* $X_{N\ x\ M}$ *',* where - 'N' training instances each with 'M' features, here M = 6 as there are 6 features in the dataset

2. *Output vector, '* $T_{N\ x\ C}$ *'* which gives the correct labels of N training features as one hot encoding vector.

HYPER-PARAMETERS :

	SINGLE HIDDEN LAYER	TWO HIDDEN LAYERS
ITERATIONS	4000	4000
HIDDEN LAYER N1	256	256
HIDDEN LAYER N2	_	128
LEARNING RATE	0.1, 0.05, 0.01	0.1, 0.05, 0.01

OUTPUT :

1. *Weight vector 1 – between input and hidden layer 1 '* $W_{M\ x\ N1}$ *',*
2. *Weight vector 2 – between hidden layer 1 and hidden layer 2 '* $W_{N1\ x\ N2}$ *',*
3. *Weight vector 3 – between hidden layer 2 and output '* $W_{N2\ x\ C}$ *',*

All the best weights are stored in two .hd5 files according to best accuracy found.

TRAINING :

During training, the first step is forward propagation in each step. Considering a single layer in the model, where x is the input and a is the output, equations (1) and (2) give us the computations taking place at each node. The cost function is calculated using equation 3 which is the categorical cross entropy loss function.

$$z = xw + b \rightarrow z = f(w) \quad (1)$$

$$a = sig(z) or\ softmax(z) \rightarrow a = f(z) \quad (2)$$

$$cost\ = -(y * log(a3)) \quad (3)$$

For the Back propagation step we start with finding the derivative of cost function with respect to w3 (or w2 for single hidden layer) that is the weight of last layer in order to get the best weights using eqn. (4) and (5).

$$\frac{d(cost)}{d(w3)} \rightarrow \frac{d(cost)}{d(a3)} . \frac{d(a3)}{d(z3)} . \frac{d(z3)}{d(w3)} = (a3 - y) * a2 \qquad (4)$$

$$w3 = w3 - \frac{d(cost)}{d(w3)}) \qquad (5)$$

$$b3 = b3 - \frac{d(cost)}{d(b3)} [\frac{d(cost)}{d(b3)} \rightarrow (a3 - y)] \qquad (6)$$

For the Back propagation step of hidden layers, we find derivative of cost function with respect to hidden layer weights (w1, w2) using eqn. (7) and (8).

$$\frac{d(cost)}{d(w2)} \rightarrow \frac{d(cost)}{d(a2)} . \frac{d(a2)}{d(z2)} . \frac{d(z2)}{d(w2)} = (a3 - y) * w3 * sig\_der(z2) * a1 \qquad (7)$$

$$\frac{d(cost)}{d(w1)} \rightarrow \frac{d(cost)}{d(a1)} . \frac{d(a1)}{d(z1)} . \frac{d(z1)}{d(w1)} = \left(\frac{d(cost)}{d(a2)} . \frac{d(a2)}{d(z2)}\right) * w2 * sig\_der(z1) * x \qquad (8)$$

$$w2 = w2 - \frac{d(cost)}{d(w2)}) , \quad w1 = w1 - \frac{d(cost)}{d(w1)}) \qquad (9) , (10)$$

Using the above equations we make the back propagation function and use it for weight update and bias update for minimum possible loss.

TESTING :

During testing accuracy only the forward propagation steps are used to get the one hot encoding of the predicted output which is given by a3. Later the one hot vector is converted to normal form so that it can be compared with actual output and accuracy can be calculated

RESULTS - FINAL

Learning rate = 0.1	SINGLE HIDDEN LAYER	TWO HIDDEN LAYERS
TRAINING ACCURACY	74.93%	75.14%
TESTING ACCURACY	75.33%	75.5%
TRAINING LOSS	0.68	6.09
TESTING LOSS	0.68	0.59

Learning rate = 0.05	SINGLE HIDDEN LAYER	TWO HIDDEN LAYERS
TRAINING ACCURACY	73.35%	74.79%
TESTING ACCURACY	73.5%	75%

Learning rate = 0.01	SINGLE HIDDEN LAYER	TWO HIDDEN LAYERS
TRAINING ACCURACY	69.0%	71.36%
TESTING ACCURACY	68.83%	71.33%

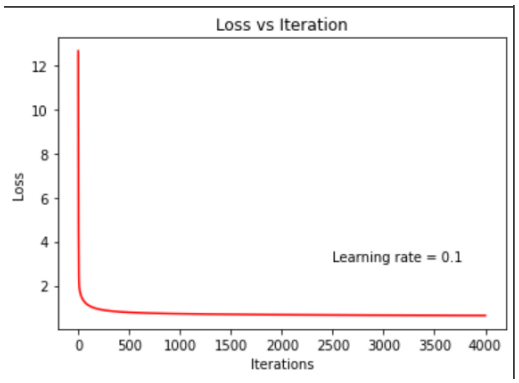
Note : The accuracy and loss for learning rate 0.1 have been considered final loss and accuracy

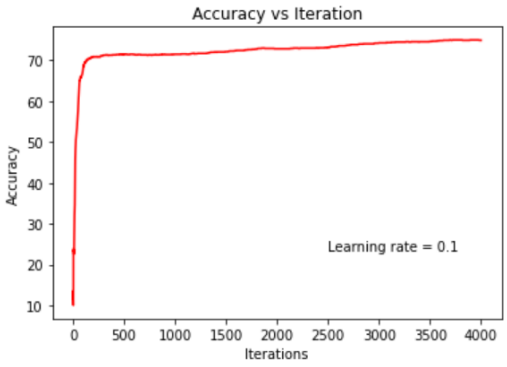
CONCLUSION

ANN with two hidden layers is more likely to give better results than ANN with single layer because both the training and testing accuracy is higher for ANN with two hidden layers. It ***takes two times longer to train the double hidden layered ANN*** (with given number of neurons) which took around 230 seconds per 500 iterations compared to single layered which took 30 seconds per 500 iterations. For the same number of iterations and other hyper-parameters higher learning rate gives better results of accuracy because model learns faster. We ***need higher amount of iterations for lower learning rate***.

1. *ACCURACY and LOSS PLOTS FOR SINGLE HIDDEN LAYER*

A. Learning rate = 0.1

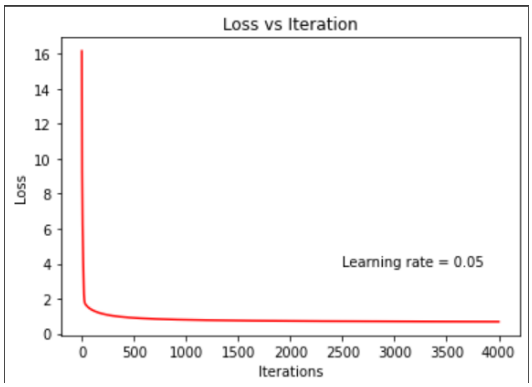


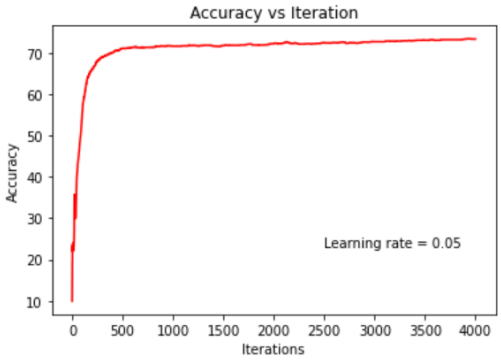


1.1 Loss vs Iteration Plot

1.2 Accuracy vs Iteration Plot

B. Learning rate = 0.05

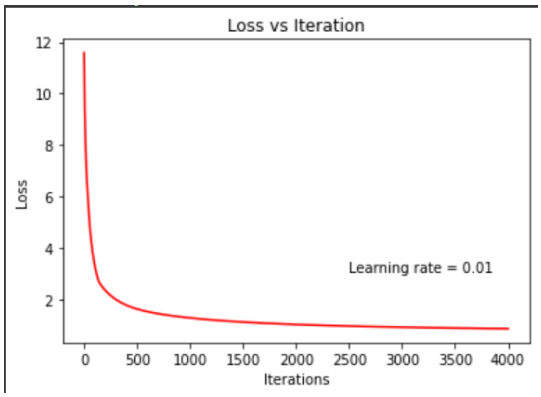


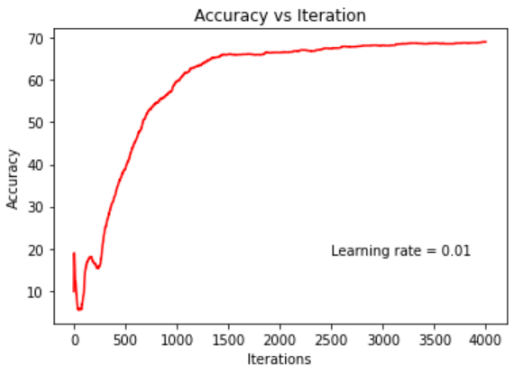


1.3 Loss vs Iteration Plot

1.4 Accuracy vs Iteration Plot

C. Learning rate = 0.01



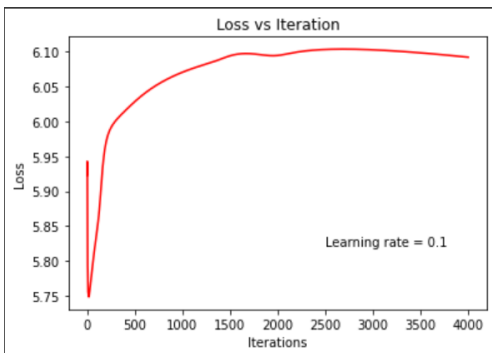


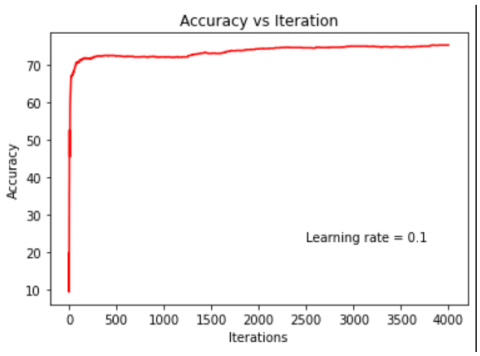
1.5 Loss vs Iteration Plot

1.6 Accuracy vs Iteration Plot

2. *ACCURACY and LOSS PLOTS FOR TWO HIDDEN LAYERS*

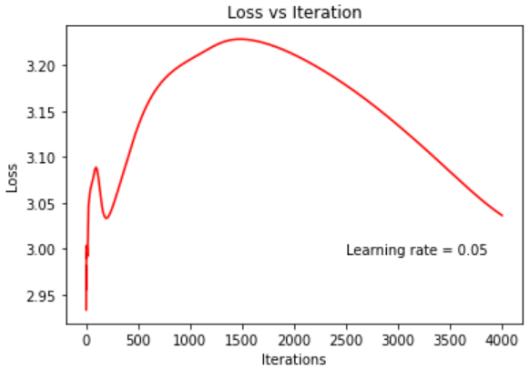
A. Learning rate = 0.1

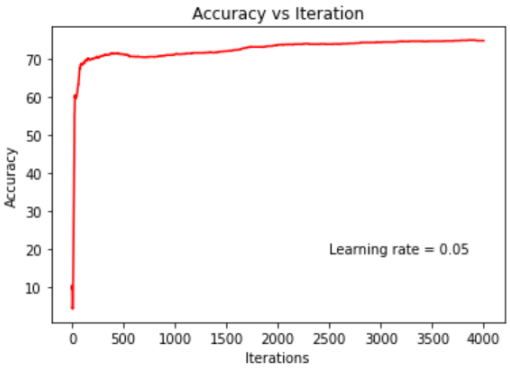




2.1 Loss vs Iteration Plot 2.2 Accuracy vs Iteration Plot

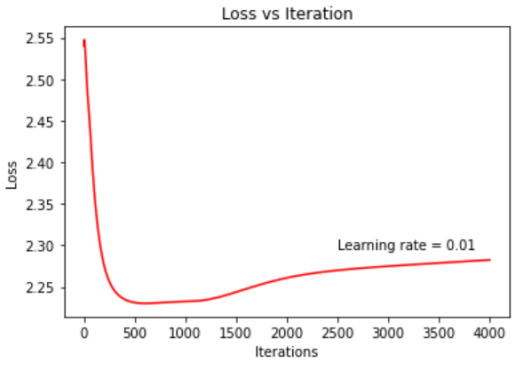
B. Learning rate = 0.05

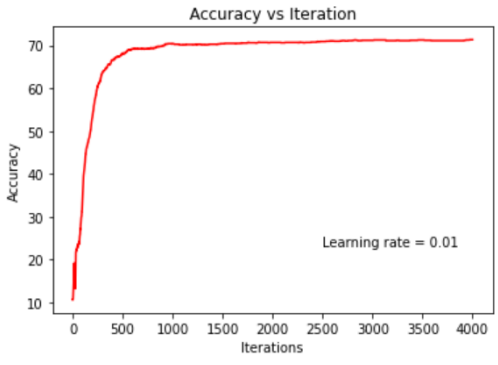




2.2 Loss vs Iteration Plot 2.4 Accuracy vs Iteration Plot

C. Learning rate = 0.01





2.5 Loss vs Iteration Plot 2.6 Accuracy vs Iteration Plot