PROGRAMMING TEST: LEARNING ACTIVATIONS IN NEURAL NETWORK

A PREPRINT

Harikrishnan NB
Consciousness Studies Programme,
National Institute of Advanced Studies, Indian Institute of Science Campus, Bengaluru, India.
harikrishnan.nb@nias.res.in, harikrishnannb07@gmail.com

June 5, 2022

ABSTRACT

In this programming test, a one hidden layer neural network model that adapts to the most suitable activation function according to the dataset is developed. The mathematical formulation of the activation function used in this method is $g(x) = k_0 + k_1 x$, where k_0, k_1 are learned from the data via gradient descent. The performance of the one hidden layer neural network model is evaluated on Breast Cancer Wisconsin, Iris, Bank Note Authentication, and MNIST datasets.

1 Background

In the Deep Learning literature, the selection of activation function is mostly based on brute-force stratergy. A popularly used activation function such as *ReLU*, *sigmoid*, etc. are incorporated in the model. If the activation function is not working, then repeat the whole process with a new activation function. This process is very time consuming especially when dealing with high dimensional data. Hence, it is important to develop a framework where a suitable activation function is discovered from the data - *Data Driven Discovery of Activation Function*. The possibilities of such an approach could not only save significant time and effort for tuning the model, but will also open up new ways for discovering essential features of not-so-popular activation functions.

2 Mathematical Framework

In this section, we shall describe the mathematical framework for the data driven discovery of activation function. For the sake of simplicity, we start with a binary classification problem. We shall develop a two layer neural network (one hidden layer and one output layer) for solving binary classification problem. Then we shall extend the framework for the multiclass classification problem.

2.1 Forward Pass and Backward Pass for a single data instance

In this section we concretely show the forward and backward pass for a single data instance. We consider a binary classification problem for mathematical easiness. Figure 1 depicts the neural network of our interest.

1. **Data:** Let the training data represented as X contains m data instances and two features ($n = 2$).
2. **Two Layer Neural Network Model:** We consider a two layer neural network with 2 nodes in the input layer, 2 nodes in the first hidden layer and 2 nodes in the output layer (Refer Figure 1).
3. **Forward Pass** To begin with, we shall show the forward and backward pass for a single data instance $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

   The following are the steps in the forward pass:

$$Z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ \\ z_2^{[1]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ \\ b_2^{[1]} \end{bmatrix} \tag{1}$$

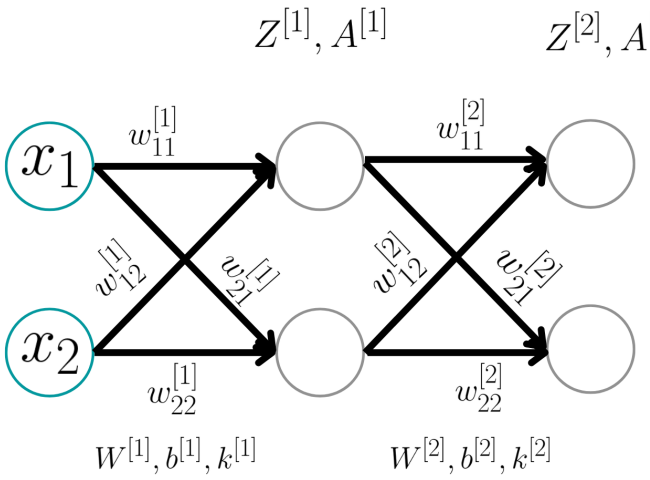

Figure 1: Two layer neural network with two neurons in the input layer, two neurons in the hidden layer and two neurons in the output layer.

$$A^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} & 1 \\ z_2^{[1]} & 1 \end{bmatrix} \begin{bmatrix} k_1^{[1]} \\ k_0^{[1]} \end{bmatrix}, \tag{2}$$

$$Z^{[2]} = \begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \end{bmatrix}, \tag{3}$$

$$A^{[2]} = \begin{bmatrix} a_1^{[2]} \\ a_2^{[2]} \end{bmatrix} = \begin{bmatrix} \frac{\exp^{Z_1^{[2]}}}{\exp^{Z_1^{[2]}} + \exp^{Z_2^{[2]}}} \\ \frac{\exp^{Z_2^{[2]}}}{\exp^{Z_1^{[2]}} + \exp^{Z_2^{[2]}}} \end{bmatrix}, \tag{4}$$

$A^{[2]}$ is passed to the categorical cross entropy loss function. Loss function calculates the error in classification. The categorical cross entropy (CCE) is defined as follows:

$$CCE = -\sum_{i=1}^{C} y_i \log(a_i^{[2]}), \tag{5}$$

where C is the total number of classes (in this case $C = 2$), y_i is the true label in one hot encoding format.

The output of a loss function is a scalar. The aim is to minimise the loss with respect to $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, k^{[1]}$. This involves computing the derivative with respect to above variables. Following that we shall apply gradient descent algorithm to find the optimum parameter values.

4. **Backward Pass:** We shall find the partial derivatives of the loss function with respect to the learnable parameters. This involves the usage of chain rule.

$$\frac{\partial L}{\partial w_{11}^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}} = a_1^{[1]}(a_1^{[2]} - y_1), \tag{6}$$

$$\frac{\partial L}{\partial w_{12}^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial w_{12}^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial w_{12}^{[2]}} = a_2^{[1]}(a_1^{[2]} - y_1), \tag{7}$$

$$\frac{\partial L}{\partial w_{21}^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} \cdot \frac{\partial z_2^{[2]}}{\partial w_{21}^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}} \cdot \frac{\partial z_2^{[2]}}{\partial w_{21}^{[2]}} = a_1^{[1]}(a_2^{[2]} - y_2), \tag{8}$$

$$\frac{\partial L}{\partial w_{22}^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} \cdot \frac{\partial z_2^{[2]}}{\partial w_{22}^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}} \cdot \frac{\partial z_2^{[2]}}{\partial w_{22}^{[2]}} = a_2^{[1]}(a_2^{[2]} - y_2), \tag{9}$$

The above can be compactly represented as follows:

$$\boxed{\frac{\partial L}{W^{[2]}} = (A^{[2]} - y) \cdot A^{[1]^T}.} \tag{10}$$

where '$\cdot$' represents matrix multiplication, T represents transpose and y represents the one hot label representation of the data instance x.

$$\frac{\partial L}{\partial b_1^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial b_1^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial b_1^{[2]}} = a_1^{[2]} - y_1, \tag{11}$$

$$\frac{\partial L}{\partial b_2^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} \cdot \frac{\partial z_2^{[2]}}{\partial b_2^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}} \cdot \frac{\partial z_2^{[2]}}{\partial b_2^{[2]}} = a_2^{[2]} - y_2, \tag{12}$$

The above can be compactly represented as follows:

$$\boxed{\frac{\partial L}{\partial b^{[1]}} = A^{[2]} - y.} \tag{13}$$

$$\frac{\partial L}{\partial k_1^{[1]}} = \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}}\right] \cdot \left[\frac{\partial z_1^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial k_1^{[1]}} + \frac{\partial z_1^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial k_1^{[1]}}\right] + \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}}\right] \cdot \left[\frac{\partial z_2^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial k_1^{[1]}} + \frac{\partial z_2^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial k_1^{[1]}}\right], \tag{14}$$

On simplification we get the following,

$$\frac{\partial L}{\partial k_1^{[1]}} = (a_1^{[2]} - y_1) \cdot \left[w_{11}^{[2]} \cdot z_1^{[1]} + w_{12}^{[2]} \cdot z_2^{[1]}\right] + (a_2^{[2]} - y_2) \cdot \left[w_{21}^{[2]} \cdot z_1^{[1]} + w_{22}^{[2]} \cdot z_2^{[1]}\right], \tag{15}$$

$$\frac{\partial L}{\partial k_0^{[1]}} = \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}}\right] \cdot \left[\frac{\partial z_1^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial k_0^{[1]}} + \frac{\partial z_1^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial k_0^{[1]}}\right] + \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}}\right] \cdot \left[\frac{\partial z_2^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial k_0^{[1]}} + \frac{\partial z_2^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial k_0^{[1]}}\right], \tag{16}$$

On simplification, we get the following:

$$\frac{\partial L}{\partial k_0^{[1]}} = (a_1^{[2]} - y_1) \cdot \left[w_{11}^{[2]} + w_{12}^{[2]}\right] + (a_2^{[2]} - y_2) \cdot \left[w_{21}^{[2]} + w_{22}^{[2]}\right], \tag{17}$$

The above can be compactly represented as follows:

$$\boxed{\frac{\partial L}{\partial k^{[1]}} = \begin{bmatrix} sum_e\left(W^{[2]} \cdot Z^{[1]} * (A^{[2]} - y)\right) \\ sum_e\left(W^{[2]^T} \cdot (A^{[2]} - y)\right) \end{bmatrix}.} \tag{18}$$

where sum_e represents the sum of all elements, and $*$ represents element-wise multiplication.

$$\frac{\partial L}{w_{11}^{[1]}} = \frac{\partial L}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{11}^{[2]} \cdot k_1^{[1]} \cdot x_1 + (a_2^{[2]} - y_2) \cdot w_{21}^{[2]} \cdot k_1^{[1]} \cdot x_1, \tag{19}$$

$$\frac{\partial L}{w_{12}^{[1]}} = \frac{\partial L}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial w_{12}^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{11}^{[2]} \cdot k_1^{[1]} \cdot x_2 + (a_2^{[2]} - y_2) \cdot w_{21}^{[2]} \cdot k_1^{[1]} \cdot x_2, \tag{20}$$

$$\frac{\partial L}{w_{21}^{[1]}} = \frac{\partial L}{\partial z_2^{[1]}} \cdot \frac{\partial z_2^{[1]}}{\partial w_{21}^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{12}^{[2]} \cdot k_1^{[1]} \cdot x_1 + (a_2^{[2]} - y_2) \cdot w_{22}^{[2]} \cdot k_1^{[1]} \cdot x_1, \tag{21}$$

$$\frac{\partial L}{w_{22}^{[1]}} = \frac{\partial L}{\partial z_2^{[1]}} \cdot \frac{\partial z_2^{[1]}}{\partial w_{22}^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{12}^{[2]} \cdot k_1^{[1]} \cdot x_2 + (a_2^{[2]} - y_2) \cdot w_{22}^{[2]} \cdot k_1^{[1]} \cdot x_2, \tag{22}$$

The above can be compactly represented as follows:

$$\boxed{\frac{\partial L}{\partial W^{[1]}} = sum_c \left(W^{[2]^T} \cdot (A^{[2]} - y) * k_1^{[1]} \right) \cdot x^T.} \tag{23}$$

where sum_c represents the column sum.

$$\frac{\partial L}{b_1^{[1]}} = \frac{\partial L}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial b_1^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{11}^{[2]} \cdot k_1^{[1]} + (a_2^{[2]} - y_2) \cdot w_{21}^{[2]} \cdot k_1^{[1]}, \tag{24}$$

$$\frac{\partial L}{b_2^{[1]}} = \frac{\partial L}{\partial z_2^{[1]}} \cdot \frac{\partial z_2^{[1]}}{\partial b_2^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{12}^{[2]} \cdot k_1^{[1]} + (a_2^{[2]} - y_2) \cdot w_{22}^{[2]} \cdot k_1^{[1]}, \tag{25}$$

The above can be compactly represented as follows:

$$\boxed{\frac{\partial L}{\partial b^{[1]}} = sum_c \left(W^{[2]^T} \cdot (A^{[2]} - y) * k_1^{[1]} \right).} \tag{26}$$

5. **Gradient Descent Algorithm**: Now, we shall use the gradient descent algorithm for updating the learnable parameters of the two layer neural network. Following parameters are updated:
   - $W^{[2]} := W^{[2]} - \alpha \cdot \frac{\partial L}{W^{[2]}}$
   - $b^{[2]} := b^{[2]} - \alpha \cdot \frac{\partial L}{b^{[2]}}$
   - $K^{[1]} := K^{[1]} - \alpha \cdot \frac{\partial L}{K^{[1]}}$
   - $W^{[1]} := W^{[1]} - \alpha \cdot \frac{\partial L}{W^{[1]}}$
   - $b^{[1]} := b^{[1]} - \alpha \cdot \frac{\partial L}{b^{[1]}}$

   where α is the learning rate.

3 Dataset Description

3.1 Breast Cancer Wisconsin

Breast Cancer Wisconsin ([1,2]) dataset consists of two class. They are: class 'M' (refers to a malignant infection level) and class 'B' (refers to a benign infection level). There are to total of 569 data instances with 31 attributes such as radius, perimeter, texture, smoothness, etc. for each cell nucleus. The class distribution for training and testing is provided in Table 1.

3.2 Iris

Iris ([3,4]) is a three class classification problem of three types of iris plantss: Iris Setosa, Iris Versicolour, and Iris Virginica. There are 150 data instances in this dataset with four attributes in each data instance: sepal length, sepal width, petal length, and petal width. All attributes are in *cms*. The specified class distribution provided in Table 1 is in the following order: (Setosa, Versicolour, Virginica).

3.3 Bank Note Authentication

Bank-note Authentication ([1,5]) is a binary classification problem involving the classification of Genuine and Forgery banknotes. A Genuine class refers to an authentic banknote denoted by '0', while a Forgery class refers to a forged banknote denoted by '1'. The dataset was curated by extracting four features from the images of banknotes belonging to the two classes. These features are obtained through wavelet transformation. There are a total of 1372 data instances. The class distribution for training and testing is provided in Table 1.

3.4 MNIST

MNIST is a publicly available computer vision dataset corresponding to handwritten digits from 0 to 9. This is a 10 class classification task. Each image in their respective classes has a dimension of 28 pixels $\times$ 28 pixels. There is a total of 70000 images in MNIST database. The class distribution for training and testing is provided in Table 1.

Table 1: Train-Test split in experiments. High training sample regime corresponds to an 80% and 20% split in training and testing respectively. In the Imbalanced data column, 'Y' implies yes and 'N' implies no.

Dataset	**Classes**	**Features**	**Training samples /class**	**Testing samples /class**	**Imbalanced data (Y/N)**
Breast Cancer Wisconsin	2	31	(169, 286)	(43, 71)	Y
Iris	3	4	(40, 41, 39)	(10, 9, 11)	N
Bank Note Authentication	2	4	(614, 483)	(148, 127)	Y
MNIST	10	784	(5560, 6277, 5610, 5708, 5529, 5040, 5480, 5790, 5468, 5538)	(1343, 1600, 1380, 1433 1295, 1273, 1396, 1503 1357, 1420)	Y

4 Experiments and Results

A three fold crossvalidation on the training set to determine the best hyperparameters (number of nodes in the hidden layer) for the two layer neural network is carried out. For this, train data is split into training and validation instances. The average F1-Score of three fold crossvalidation is found. We choose the hyperparameter which gives maximum average F1-Score. Now, we retrain the model with the entire train data. The model is then tested on test data (the test data is visited only once). The system configuration used to perform the experiments are the following: Manufacturer and model: Acer, Predator G3-571, Processor: Intel(R) Core(^{TM}) i7-7700HQCPU 2.80GHz, System type: 64 bit Operating System, x64-based processor. The platform used to implement the code is Python 3.7.

4.1 Breast Cancer Wisconsin

4.1.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[60, 90, 100, 110, 500]$ and learning rate (α) = 0.21, three fold crossvalidation has been performed on the training set. A maximum average macro F1-score = 0.91 has been obtained for the validation data in the three-fold crossvalidation experiment, when the number of nodes in the hidden layer is 100. For testing, we choose $\alpha = 0.21$, and $n_h = 100$ (number of nodes in the hidden layer). Figure 2 depicts the three-fold crossvalidation on breast cancer wisconsin data for varying number of nodes in the hidden layer.

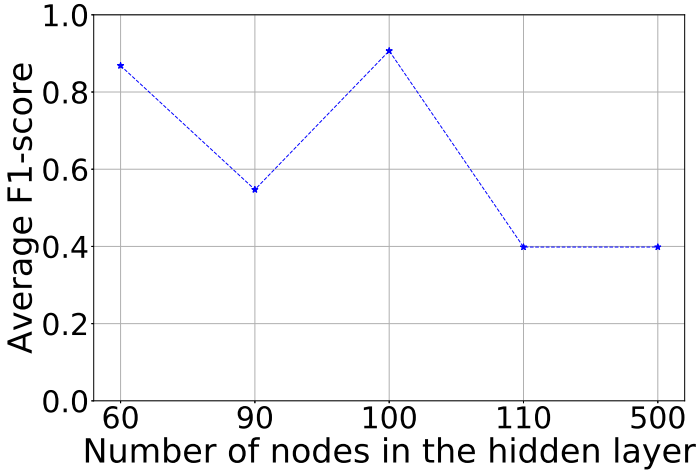

Figure 2: Breast Cancer Wisconsin: Three-fold crossvalidation.

4.1.2 Training and Testing

The specifics of the two layer neural network architecture is the following: the number nodes in the input layer is 30, the number of nodes in the first hidden layer is 100, the number of nodes in the output layer is 2. The train-test distribution is provided in Table 1. Training was done for 6 epochs with $\alpha = 0.21$. The training loss is provided in Figure 3.

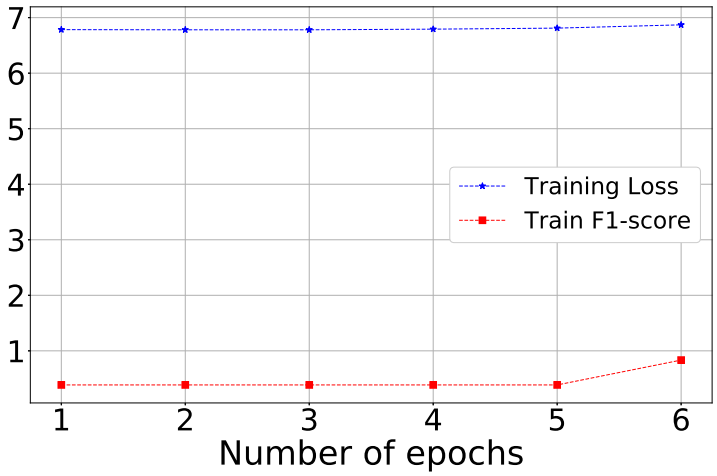

Figure 3: Breast Cancer Wisconsin: Training Loss and Train F1-score vs number of epochs

We choosed the parameters of the model $(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, k^{[1]})$ corresponding to the best training F1-score. From Figure 3, we choose the parameters of the network corresponding to epoch 6. Table 2 provides the training accuracy and F1-score of Breast cancer wisconsin data.

4.2 Iris

4.2.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[10, 30, 60, 90, 100, 110]$ and learning rate $(\alpha) = 0.249$, three fold crossvalidation has been performed on the training set. A maximum average macro F1-score = 0.53 has been obtained has been obtained for the validation data in the three-fold crossvalidation experiment, when the number of nodes in the hidden layer is 100. For testing, we choose $\alpha = 0.249$, and $n_h = 100$. Figure 4 depicts the three-fold crossvalidation on Iris data for varying number of nodes in the hidden layer.

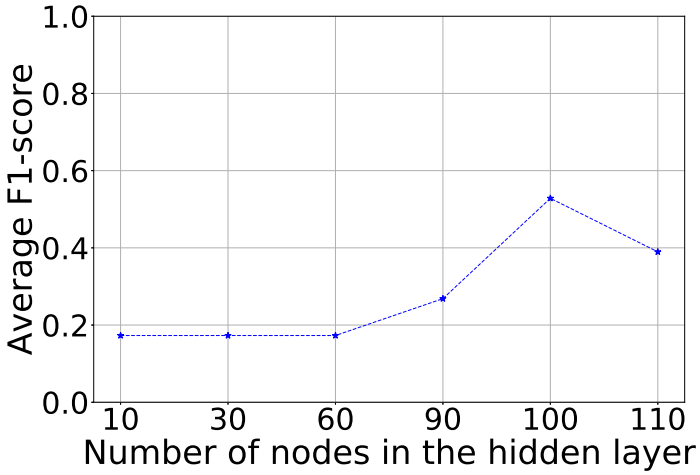

Figure 4: Iris: Three-fold crossvalidation.

4.2.2 Training and Testing

The specifics of the two layer neural network architecture is the following: the number nodes in the input layer is 4, the number of nodes in the first hidden layer is 100, the number of nodes in the output layer is 3. The train-test distribution is provided in Table 1. Training was done for 500 epochs with $\alpha = 0.249$. The training loss is provided in Figure 5.

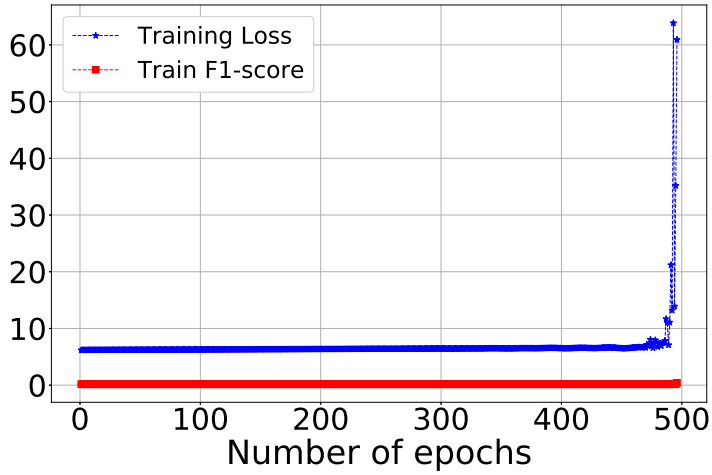

Figure 5: Iris: Training Loss and Train F1-score vs number of epochs

We choosed the parameters of the model corresponding to the best training F1-score. From Figure 5, we choose the parameters of the network corresponding to epoch 500. Table 2 provides the training accuracy and F1-score of Iris data.

4.3 Bank Note Authentication

4.3.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[5, 10, 30, 60, 90, 100, 110]$ and learning rate (α) = 0.291, three fold crossvalidation has been performed on the training set. A maximum average macro F1-score = 0.357 has been obtained for all the above values of nodes in the three-fold crossvalidation experiment. For testing, we choose $\alpha = 0.249$, and $n_h = 5$ (number of nodes in the hidden layer). Figure 6 depicts the three-fold crossvalidation on bank note authentication data for varying number of nodes in the hidden layer.

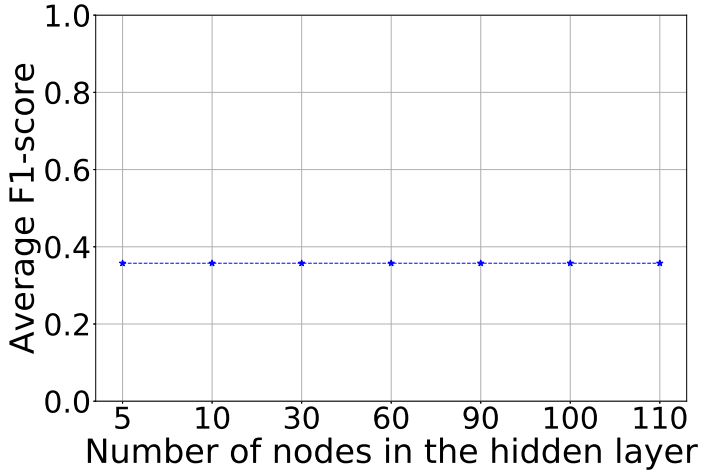

Figure 6: Bank Note Authentication: Three-fold crossvalidation.

4.3.2 Training and Testing

The specifics of the two layer neural network architecture is the following: the number nodes in the input layer is 4, the number of nodes in the first hidden layer is 5, the number of nodes in the output layer is 2. The train-test distribution is provided in Table 1. Training was done for 500 epochs with $\alpha = 0.291$. The training loss is provided in Figure 7.

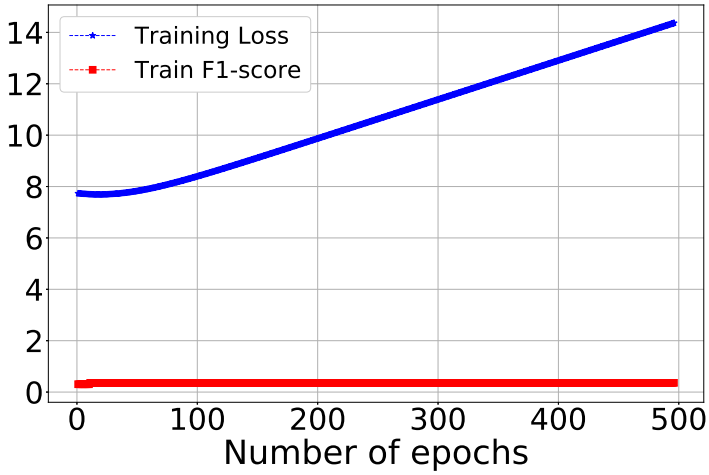

Figure 7: Bank Note Authentication: Training Loss and Train F1-score vs number of epochs

We choosed the parameters of the model corresponding to the best training F1-score. From Figure 7, we choose the parameters of the network corresponding to epoch 11. Table 2 provides the training accuracy and F1-score of Bank Note Authentication data.

4.4 MNIST

4.4.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[790, 1024, 4000]$ and learning rate (α) = 0.198, three fold crossvalidation has been performed on the training set. A maximum average macro F1-score = 0.40 has been obtained for $n_h = 4000$ nodes in the three-fold crossvalidation experiment. For testing, we choose $\alpha = 0.198$, and $n_h = 4000$. Figure 8 depicts the three-fold crossvalidation on mnist data for varying number of nodes in the hidden layer.

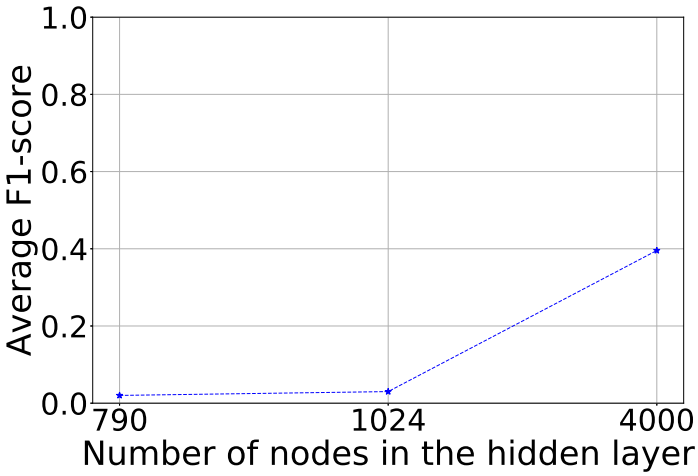

Figure 8: MNIST: Three-fold crossvalidation.

Table 2: Testdata results for the two layer neural network.

Dataset	Train Accuracy	Train F1-score	Test Accuracy	Test F1-score
Breast Cancer Wisconsin	85.93%	0.83	90.35%	0.89
Iris	50%	0.41	56.7%	0.45
Bank Note Authentication	55.97%	0.36	53.82%	0.35
MNIST	40.20%	0.35	40.77%	0.35

4.4.2 Training and Testing

The specifics of the two layer neural network architecture is the following: the number nodes in the input layer is 784, the number of nodes in the first hidden layer is 4000, the number of nodes in the output layer is 10. The train-test distribution is provided in Table 1. Training was done for 25 epochs with $\alpha = 0.198$. The training loss is provided in Figure 9.

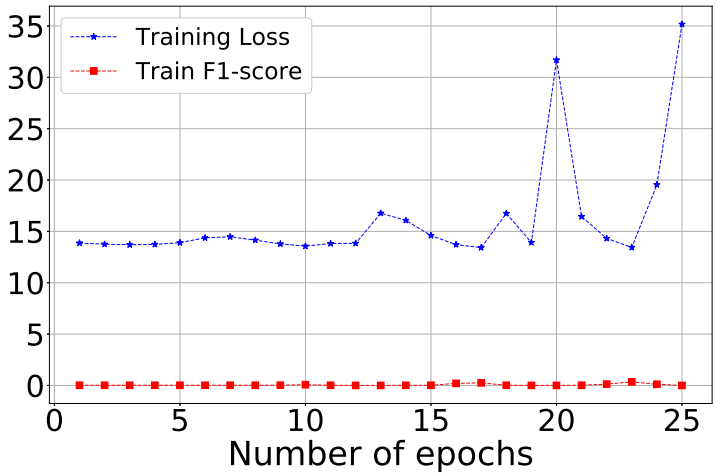

Figure 9: MNIST: Training Loss and Train F1-score vs number of epochs

We choose the parameters of the model corresponding to the best training F1-score. From Figure 9, we choose the parameters of the network corresponding to epoch 11. Table 2 provides the training accuracy and F1-score of mnist data.

5 Summary

In the case of Breast Cancer Wisconsin data, we get highest performance on testdata (test F1-score = 0.89), whereas in all the remaining cases the performance is very poor. Another important observation noted during the experiments is the sensitivity of the neural network to initial random weights and learning rate. One reason for the poor performance of the two neural network is the absence of inherent nonlinearity in the network. Eventhough the parameters of the activation function are learned from data, in effective the activation function is doing only an affine transformation. This seems to be a huge restriction for the neural network.

Finding an optimal learning rate for a data is also an interesting problem to look at. The concept of data-driven discovery of activation function can be applied to data-driven discovery of learning rate.

Going forward, implementing the network with multiple hidden layers and nonlinear activation seems to be a promising direction for future research. The code used in this research is available in the following GitHub link: `https://github.com/HarikrishnanNB/programming_test`.

Acknowledgements

I thank Dr. Snehanshu Saha (Associate Professor, Department of Computer Science and Information Systems, KK Birla Goa Campus) for giving me this opportunity.

References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. SPIE, 1993.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[5] Eugen Gillich and Volker Lohweg. Banknote authentication. 11 2010.

Supplementary Material

The intermediate steps used in computing the derivatives of the loss function with respect to learnable parameters is provided below:

$$\frac{\partial L}{\partial a_1^{[2]}} = \frac{-y_1}{a_1^{[2]}}. \tag{27}$$

$$\frac{\partial L}{\partial a_2^{[2]}} = \frac{-y_2}{a_2^{[2]}}. \tag{28}$$

$$\frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} = a_1^{[2]}(1-a_1^{[2]}). \tag{29}$$

$$\frac{\partial a_2^{[2]}}{\partial z_2^{[2]}} = a_2^{[2]}(1-a_2^{[2]}). \tag{30}$$

$$\frac{\partial a_2^{[2]}}{\partial z_1^{[2]}} = -a_1^{[2]}a_2^{[2]}. \tag{31}$$

$$\frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} = -a_1^{[2]}a_2^{[2]}. \tag{32}$$

$$\frac{\partial z_1^{[2]}}{\partial a_1^{[1]}} = w_{11}^{[2]}. \tag{33}$$

$$\frac{\partial z_1^{[2]}}{\partial a_2^{[1]}} = w_{12}^{[2]}. \tag{34}$$

$$\frac{\partial z_2^{[2]}}{\partial a_1^{[1]}} = w_{21}^{[2]}. \tag{35}$$

$$\frac{\partial z_2^{[2]}}{\partial a_2^{[1]}} = w_{22}^{[2]}. \tag{36}$$

$$\frac{\partial a_1^{[1]}}{\partial z_1^{[1]}} = k_1^{[1]}. \tag{37}$$

$$\frac{\partial a_1^{[1]}}{\partial k_1^{[1]}} = z_1^{[1]}. \tag{38}$$

$$\frac{\partial a_2^{[1]}}{\partial k_1^{[1]}} = z_2^{[1]}. \tag{39}$$

$$\frac{\partial a_1^{[1]}}{\partial k_0^{[1]}} = 1. \tag{40}$$

$$\frac{\partial a_2^{[1]}}{\partial k_0^{[1]}} = 1. \tag{41}$$

$$\frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}} = a_1^{[1]}. \tag{42}$$

$$\frac{\partial z_1^{[2]}}{\partial w_{12}^{[2]}} = a_2^{[1]}. \tag{43}$$

$$\frac{\partial z_2^{[2]}}{\partial w_{21}^{[2]}} = a_1^{[1]}. \tag{44}$$

$$\frac{\partial z_2^{[2]}}{\partial w_{22}^{[2]}} = a_2^{[1]}. \tag{45}$$

$$\frac{\partial z_1^{[2]}}{\partial b_1^{[2]}} = 1. \tag{46}$$

$$\frac{\partial z_2^{[2]}}{\partial b_2^{[2]}} = 1. \tag{47}$$

$$\frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}} = x_1. \tag{48}$$

$$\frac{\partial z_1^{[1]}}{\partial w_{12}^{[1]}} = x_2. \tag{49}$$

$$\frac{\partial z_2^{[1]}}{\partial w_{21}^{[1]}} = x_1. \tag{50}$$

$$\frac{\partial z_2^{[1]}}{\partial w_{22}^{[1]}} = x_2. \tag{51}$$

$$\frac{\partial z_1^{[1]}}{\partial b_1^{[1]}} = 1. \tag{52}$$

$$\frac{\partial z_2^{[1]}}{\partial b_2^{[1]}} = 1. \tag{53}$$

$$\frac{\partial L}{\partial z_1^{[1]}} = \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}}\right] \cdot \left[\frac{\partial z_1^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial z_1^{[1]}}\right] + \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}}\right] \cdot \left[\frac{\partial z_2^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial z_1^{[1]}}\right], \tag{54}$$

On simplification we get,

$$\frac{\partial L}{\partial z_1^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{11}^{[2]} \cdot k_1^{[1]} + (a_2^{[2]} - y_2) \cdot w_{21}^{[2]} \cdot k_1^{[1]}. \tag{55}$$

$$\frac{\partial L}{\partial z_2^{[1]}} = \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_1^{[2]}}\right] \cdot \left[\frac{\partial z_1^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial z_2^{[1]}}\right] + \left[\frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_2^{[2]}} + \frac{\partial L}{\partial a_2^{[2]}} \cdot \frac{\partial a_2^{[2]}}{\partial z_2^{[2]}}\right] \cdot \left[\frac{\partial z_2^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial z_2^{[1]}}\right], \tag{56}$$

On simplification we get,

$$\frac{\partial L}{\partial z_2^{[1]}} = (a_1^{[2]} - y_1) \cdot w_{12}^{[2]} \cdot k_1^{[1]} + (a_2^{[2]} - y_2) \cdot w_{22}^{[2]} \cdot k_1^{[1]}. \tag{57}$$