Implementation of Artificial Neural Networks

Lab Assignment 3

Sumedha Janani Siriyapuraju
Dept. of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur, India
sumedhasjs@gmail.com

***Abstract*—Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.**

In this work we tried to use analytical solution to fit a curve to the Matlab_cancer.mat Dataset.

I. Introduction

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

II. Method

A. Data Preprocessing

The given file was loaded and data and headers were extracted. The raw data was converted into a dataframe using initial preprocessing.

The entire dataframe was then divided into 3 groups: training, validation and testing dataset in the ratio of 60:20:20. The validation dataset was used to lessen overfitting after the ANN was trained on the training dataset. Testing on the testing dataset was the last step in determining the correctness.

B. Part A: Feedforward Network

The weights are produced for ANN utilising back propagation over a large number of iterations. The ANN network is constructed with two weights. The weights between the input layer and the hidden layer are one example. The weights between the concealed layer and the output layer are the other. A random weight is used as the baseline initially. With that, after each iteration, the weights are updated. The derivative of output is calculated for each iteration with regard to output and weight. The new weights are found using the formula $W_{ih} = W_{ih} + X^{\mathrm{T}}.\Delta H1$ and $W_{ho} = W_{ho} + H1.\Delta o$ This computation of weights is done on the training dataset. Next, these weights are implemented on the ANN with the validation dataset as the input.

C. Part B: Reducing overfitting

If the number of iterations is raised, overfitting may become an issue. It happens when the training dataset's curve is overly modified, making the training dataset's error small compared to the error in any new dataset. We employ early stopping, or stopping iterations when the error in the validation data set becomes minimal, to avoid this. Until the mean square error is minimised, a loop is conducted. The iterations are terminated and the weights obtained at this point are regarded as the final weights for the ANN after the error has reached its minimal value.

D. Part D: Efficiency

In all the above cases, the confusion matrix is created using the inbuilt function of python. The sensitivity and specificity are found out using the formula $sensitivity = tp/(tp + fn)$ and $specificity = tn/(tn + fp)$ respectively. The ROC and AUROC curve are also plotted using inbuilt functions of python.

III. Results

```
epoch: 1, loss = 0.6962, accuracy = 0.4244185984134674
epoch: 2, loss = 0.5958, accuracy = 0.819767415523529
epoch: 3, loss = 0.5192, accuracy = 0.8604651093482971
epoch: 4, loss = 0.4511, accuracy = 0.8662790656089783
epoch: 5, loss = 0.3929, accuracy = 0.8779069781303406
epoch: 6, loss = 0.3466, accuracy = 0.8779069781303406
epoch: 7, loss = 0.3130, accuracy = 0.8662790656089783
epoch: 8, loss = 0.2908, accuracy = 0.854651153087616
epoch: 9, loss = 0.2775, accuracy = 0.8662790656089783
epoch: 10, loss = 0.2702, accuracy = 0.8720930218696594
epoch: 11, loss = 0.2660, accuracy = 0.8720930218696594
epoch: 12, loss = 0.2623, accuracy = 0.8720930218696594
epoch: 13, loss = 0.2582, accuracy = 0.8837209343910217
epoch: 14, loss = 0.2532, accuracy = 0.8895348906517029
epoch: 15, loss = 0.2468, accuracy = 0.9011628031730652
epoch: 16, loss = 0.2391, accuracy = 0.9011628031730652
epoch: 17, loss = 0.2306, accuracy = 0.9011628031730652
epoch: 18, loss = 0.2216, accuracy = 0.9069767594337463
epoch: 19, loss = 0.2123, accuracy = 0.9127907156944275
epoch: 20, loss = 0.2031, accuracy = 0.9186046719551086
```

Fig. 1. Model Training for ANN with hidden layer nodes = 512

```
accuracy: 0.9318
```

Fig. 2. Accuracy of Model 1

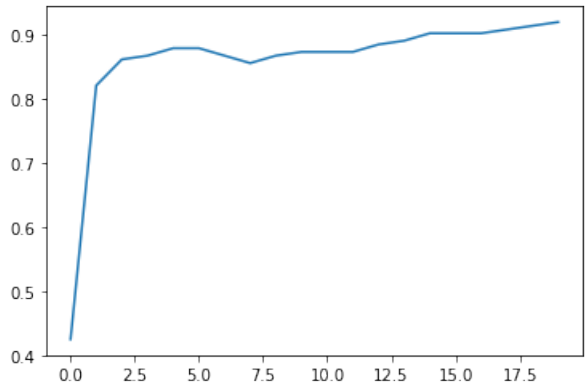

Fig. 3. Accuracy Plot of Model 1

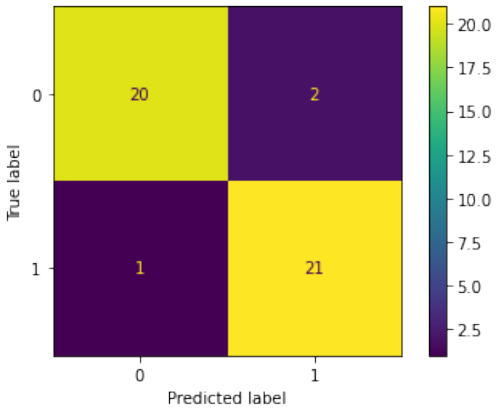

Fig. 4. Confusion Matrix of Model 1

```
Sensitivity :  0.9090909090909091
Specificity :  0.9545454545454546
```

Fig. 5. Sensitivity and Specificity of Model 1

```
AUROC Score:  0.9318181818181819
```

Fig. 6. AUROC Score of Model 1

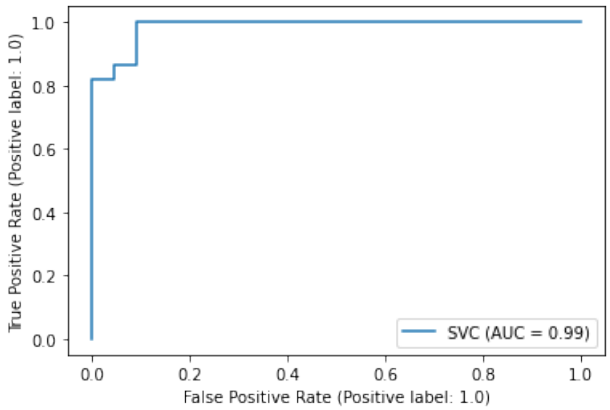

Fig. 7. ROC Curve of Model 1

```
epoch: 1, loss = 0.6951, accuracy = 0.4360465109348297
epoch: 2, loss = 0.6425, accuracy = 0.7790697813034058
epoch: 3, loss = 0.5989, accuracy = 0.7906976938247681
epoch: 4, loss = 0.5588, accuracy = 0.819767415523529
epoch: 5, loss = 0.5199, accuracy = 0.8313953280448914
epoch: 6, loss = 0.4821, accuracy = 0.8720930218696594
epoch: 7, loss = 0.4456, accuracy = 0.8720930218696594
epoch: 8, loss = 0.4109, accuracy = 0.8720930218696594
epoch: 9, loss = 0.3787, accuracy = 0.8895348906517029
epoch: 10, loss = 0.3502, accuracy = 0.8895348906517029
epoch: 11, loss = 0.3259, accuracy = 0.8779069781303406
epoch: 12, loss = 0.3061, accuracy = 0.8720930218696594
epoch: 13, loss = 0.2905, accuracy = 0.8720930218696594
epoch: 14, loss = 0.2788, accuracy = 0.8604651093482971
epoch: 15, loss = 0.2705, accuracy = 0.8604651093482971
epoch: 16, loss = 0.2644, accuracy = 0.8720930218696594
epoch: 17, loss = 0.2598, accuracy = 0.8720930218696594
epoch: 18, loss = 0.2561, accuracy = 0.8779069781303406
epoch: 19, loss = 0.2527, accuracy = 0.8837209343910217
epoch: 20, loss = 0.2492, accuracy = 0.8779069781303406
```

Fig. 8. Model Training for ANN with hidden layer nodes = 256

```
accuracy: 0.9091
```

Fig. 9. Accuracy of Model 2

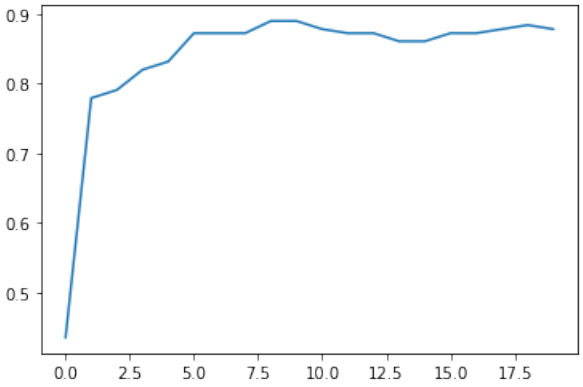

Fig. 10. Accuracy Plot of Model 2

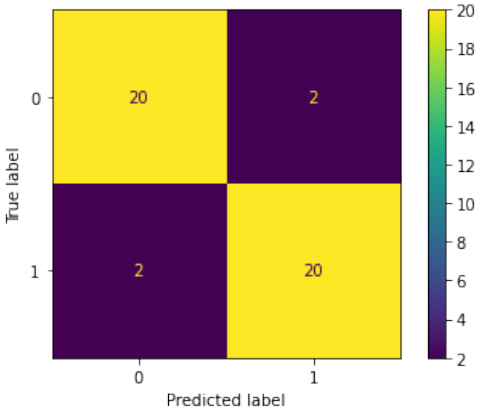

Fig. 11. Confusion Matrix of Model 2

```
Sensitivity :  0.9090909090909091
Specificity :  0.9090909090909091
```

Fig. 12. Sensitivity and Specificity of Model 2

```
AUROC Score:  0.9090909090909091
```

Fig. 13. AUROC Score of Model 2

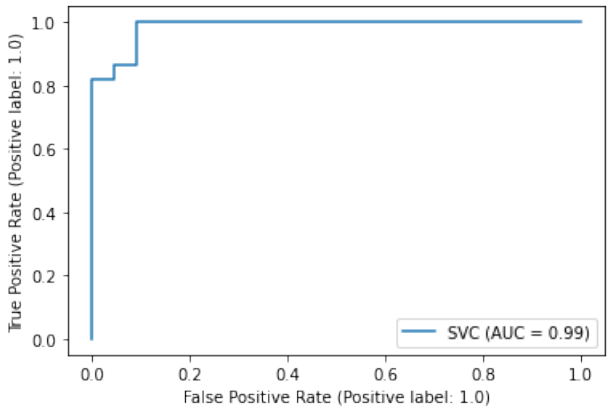

Fig. 14. ROC Curve of Model 2

IV. Discussion

- We have used 256 and 512 hidden layers to train the model.
- The learning rate was 0.001 with batch size of 4,
- The number of epochs was 20.
- ReLu was used as the Activation Functions.
- We can see that the efficiency for 512 hidden layer model is 93.18% and 256 hidden layer model is 90.91%.
- The weights get more and more optimised with increase in the number of iterations.
- Too much increase in number of iterations lead to over-fitting in the training dataset. This is curbed by using validation dataset to remove over fitting using early stopping.

V. Conclusion

Artificial Neural Network has been implemented on the given Matlab_cancer Data-set and has been studied.

Appendix (Code)

The code is available at here