Assignment 3

NEURAL NETWORKS - BITS F312

GROUP 2

Gargi Balasubramaniam
2016A7PS0365G

Alish Dipani
2016A7PS0005G

Sugam Budhraja
2016A7PS0064G

Mehul Rastogi
2016A7PS0087G

Contents

1 Introduction **2**

2 Review of the paper **3**
2.1 What is the paper about? 3
2.2 Dataset 4
2.3 The Proposed Model 5
2.4 Training the model 7
2.5 Results 8
2.6 Final Comments 11

3 Review of Neural Network Models **12**
3.1 Data set 12
3.1.1 About the Data set 12
3.1.2 Prepossessing the Data set 13
3.2 Implementation 14
3.3 Experiments 15
3.4 Conclusion 18

1 Introduction

The paper is titled " Construction of Feed Forward Multilayer Perceptron Model for Genetic Dataset in Leishmaniasis using Cognitive Computing".

This paper specifies the technical details of an Artificial Neural Network model developed by the authors in order to predict whether a given human being has Leishmaniasis or not, based on their genetic details.

In the first part of this report, we have attempted to provide a critical summary of their paper - details of their dataset, neural network model, results and comments on the overall quality of the paper- clarity and conciseness.The second part of this report talks about an ANN model built by us, following a similar workflow adopted by the authors of the above paper, along with the relevant details.

2 Review of the paper

2.1 What is the paper about?

This paper is a technical report on developing a neural network model in order to predict whether a given human being has Leishmaniasis or not.The genetic dataset used by them is collected from the **Gene Expression Omnibus Database**, and is claimed to have been processed there onwards.They have developed the model in **Python** using the **TensorFlow** library. The code has been made available on a Github[1] link provided.

Some points to consider :

- The paper includes in its title and its keywords, the term "Cognitive Computing". It broadly refers to software or hardware that attempts to mimic the human brain, and there is no universally accepted definition of this term. Thus, using such a term as a buzzword may not have been the right thing to do, given that they have built a simple ANN model .It is misleading in some sense.
- Certain lines in the introduction tend to be confusing -

  "The cognitive model of the trained network is interpreted using the maps and mathematical formula of the influencing parameters"

  These references to "maps" is unspecified in the entire paper.
- There is a long description of the background of this disease which seems unnecessary in the given technical context.

[1] https://github.com/shailzasingh/Machine-Learning-code-for-analyzing-genetic-dataset-in- Leishmaniasis

2.2 Dataset

The authors have obtained their training database from the **Gene Expression Omnibus (GEO)** database which is a public repository hosting genomic data of various diseases.

We present some points concerning their statements about the dataset-

- Out of the many datasets available on the GEO database, they have not named **which** dataset they have used.
- They have collected genetic data of 33 human beings. These comprise of 5 healthy and 28 infected human beings. This training dataset is small as well as skewed which could pose a problem for training. They could have used techniques like oversampling or SMOTE to balance the dataset.
- In context of the numbers mentioned earlier, the github link states that the dataset used is a collection of 166 datasets comprising of datasets of 38 healthy people, 78 infected people, 10 lesion samples and 40 Leishmaniasis patients. This is inconsistent with the details provided in the paper.
- They mention that the dataset has 45033 rows (features,should ideally be columns) but they fail to mention that the dataset originally has 45031 rows and the 45033 rows are achieved after one-hot encoding of the label in the dataset. This has been shown in the github code.
- The paper claims - *"The variations in the genetic dataset is normalized using log2 normalization function by leaving the least reproducing data".* However, the equation mentioned in the paper is the method for min-max scaling. Moreover, this is nowhere visible in the code as well.
- There is absolutely no information on how the data was pre-processed, say for e.g. if the data had NaNs.

  There is a line mentioning- "RNN is used to recognize the variance in the genetic dataset of leishmaniasis"

  This is nowhere manifested in the code and we have no means of verifying how this was done.

Thus, we realize that there are many inconsistencies between the code and the paper.

2.3 The Proposed Model

The authors have used an **Artificial Neural Network (ANN)**.

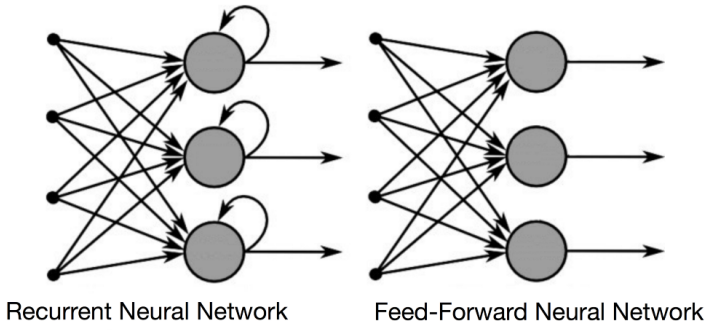
- They explain that the ANN mimics the working of a biological neuron. They state that they have used an ANN rather than the conventional multiple regression model because of their ability to handle non linear data.
- The model used in the paper is a Feed Forward Multilayer Perceptron. The neural network has **4 hidden layers each with 45 neurons** . The paper says *"multilayer perceptron is activated using sigmoid function"*. However, in the code, they have used the sigmoid activation function for the first 3 hidden layers and ReLU activation for the fourth. This is not explicitly mentioned in the paper.
- According to the authors-

  "The 45033 features of the individual sample are loaded into the 45 neurons of the input layer." It is not clear what the author means by these lines.
- The output is decided as follows - *" It is then characterized into the two classes in the output layer. The value of the output class crossing above the threshold value is given as an output of the given test data."*

  The threshold here is not specified. Neither is it clear why they have chosen such a decision process.
- They say- *" The random structure of hidden layer used in constructing the model increases the efficacy of the model, which is proved with increased accuracy of the model."*

  By *random structure* it is not clear whether they mean the random weights, or the number of neurons in each hidden layer (possibly couldn't be random because of each layer has the same number of neurons)

The authors claim to have used Recurrent Neural Network (RNN) in the model.



- According to them:

  "the activation of each layer for RNN is dependent on the output of the previous layer".

  This is not a RNN but simply a Feed Forward Neural Network.

- They also say that=

  "the highly reproducible data is captured by long short term memory of the model".

  These claims are inconsistent with the actual definitions of these models.

Recurrent Neural Networks (RNN) are feed forward neural networks that in addition to input from previous layers also use what they have learned from previous inputs.[1]

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks that have longer memory. The memory can be seen as a gated cell - based on the importance assigned to the information, it is either stored or deleted.

[1]Source: https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5

2.4 Training the model

They have presented the following specifications :

1. The model has been trained using **back propoagation**. This has been implemented using the `Gradient Descent Optimizer` function in TensorFlow.

2. 80% of the dataset is used for training and the remaining 20% is used for testing.

3. The model is trained at the learning rate of *0.001* for *500 epochs*. **How these values were decided is not specified.**

4. The paper says

   "the loss function (or) cost of the model is calculated by the prediction of deviations in the mean squared error between the actual and predicted output value".

   Yet, in the code they use

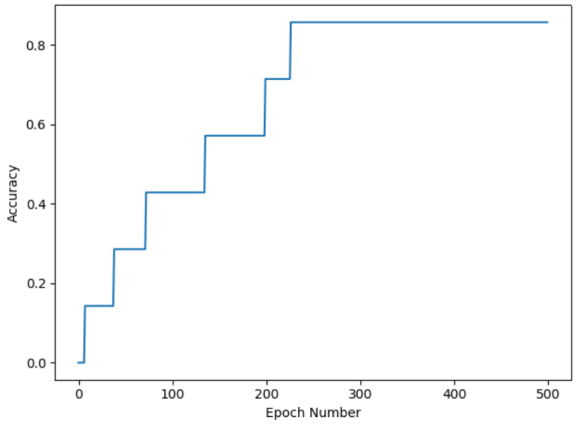
   `softmax_cross_entropy_with_logits` as the loss function.This loss function first converts the real values into a probability distribution (softmax) and then calculates the cross entropy loss.

5. Mean square error is used as the error function during gradient descent. (This is inconsistent with the error used in the code, as previously stated).

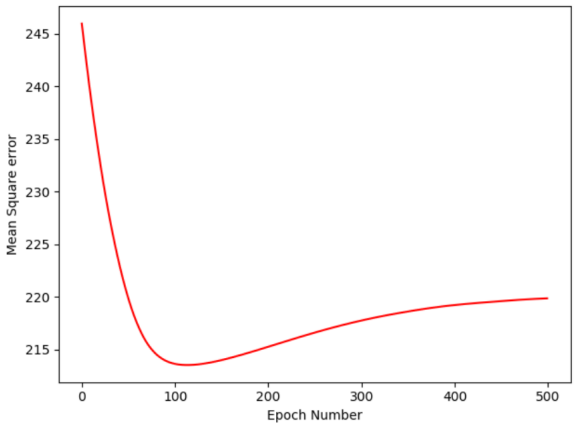
2.5 Results

We summarize the results as follows :

1. The accuracy of the model is initially 0%. This implies that the outcomes for all 6 human beings were wrongly predicted initially. It increases in steps as the model learns the outcome of each human being.



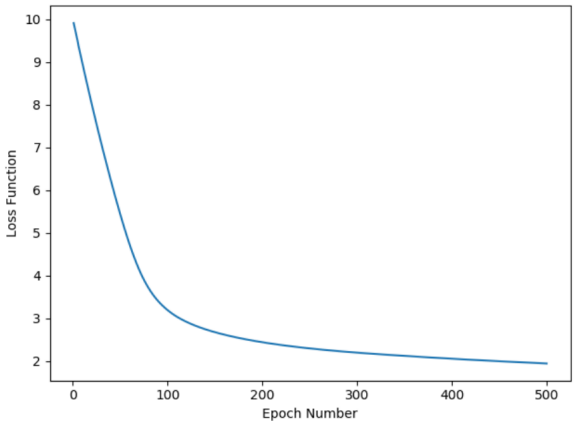
2. The mean squared error varies as follows:



From the graph we can see that the learning rate is too big. The weights overshoot when the gradient descent approaches the minima. Hence, we see the MSE increases after decreasing. This problem can be solved by decaying the learning rate.

The final accuracy achieved is 85.71%.

3. The loss is found to be 9.91 at the start of the training process. It changes as follows:



   This seems to be a reasonable result.

We now mention some lines as it is, and present our thoughts:

1. *"The increased accuracy of the model is due to the use of increased neurons in the hidden layer. The increased number of hidden layer, decreases the error of the model to the process "*

   This is true but can also cause overfitting (on increasing the number of hidden layers).

2. *" This dependency structure of the matrix made the model to learn the highest data volume of the genetic dataset and also it increases the cognitive effect of the model "*

   This line shows the unnecessary usage of buzz words and doesn't convey anything meaningful.

3. *"The accuracy of 85.71% shows it to be a good model for predicting the more reliable experimental result for the given genetic dataset of leishmaniasis in human beings. "*

   This claim should not be made given that their testing dataset consists of only 6 human beings.

4. *"The increasing value of the accuracy denotes the decreasing state of the error in the model, with the increase in regression coefficient.*

   What they mean by regression coefficient is perhaps the L2 Norm , (which is what they might have meant when they said Log2 Normalization).

5. *"The usage of back propagation along the feed forward neural network have also proved to be the better option than using spike timing-dependent plasticity model for biological datasets."*

   They claim to have achieved better results as compared to STDP learning on spiking neural networks, which has not been verified anywhere.(The citation shows better results on spiking neural networks).

2.6 Final Comments

After reviewing the paper, a highlight of the paper is redundancy (*genetic dataset, cognitive model, cognitive computing*) as well as the lack of clarity. The authors seem to have a knowledge gap regarding definitions and usage of terms with respect to what they have actually implemented. Another point to note is the inconsistency between the code cited and the paper specifications.

Some scope for improvement:

- The paper should also explain more about the data itself, i.e. the nature of the data (since we are dealing with complex genetic data with a large number of parameters) in order to facilitate our understanding of the feature set.
- More light should be thrown on the ways of preprocessing the dataset (which is a major aspect before training).
- The analysis of graphs could be made more clear with greater reasoning. The exact influence of parameters such as the learning rate could also be illustrated.
- They could also show data visualization by using Principle Component Analysis to reduce the feature space and then extract meaning out of the data.

3 Review of Neural Network Models

3.1 Data set

We have chosen Wisconsin Diagnostic Breast Cancer (WDBC) dataset. The data contains measurements of cells in suspicious lumps in a woman's breast. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. All samples are classified as either benign or malignant.[1]

3.1.1 About the Data set

The features of the dataset from a digitized image of fine needle aspirate(FNA) of breast mass. The dataset has clean information about the cell nuclei present in the image. Some sample images can be found at http://www.cs.wisc.edu/ street/images/. The dataset has 569 instances and 32 attributes. 30 of the attributes are real valued and other two define the instance ID and whether the cancer is malignant or benign. The class distribution of the dataset is as follows:-[2]

1. Benign:357
2. Malignant:212

The Attributes information in the dataset is as follows:-

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. Ten real-valued features are computed for each cell nucleus:
   (a) radius (mean of distances from center to points on the perimeter)
   (b) texture (standard deviation of gray-scale values)
   (c) perimeter
   (d) area
   (e) smoothness (local variation in radius lengths)
   (f) compactness ($perimeter^2/area - 1.0$)
   (g) concavity (severity of concave portions of the contour)
   (h) concave points (number of concave portions of the contour)
   (i) symmetry
   (j) fractal dimension ("$coastlineapproximation$" $- 1$)

[1] Source:https://rdrr.io/cran/kdevine/man/wdbc.html

[2] Further information on the data set can be found at https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names

Using the dataset we will be predicting if the cancer tumor is benign or malignant. This dataset is a real life dataset of cancer patients.We are using this dataset to test how well our model performs on a such a real life scenario. This dataset was first published in 1995. Being an old dataset this has a lot of available literature and has proven useful to test new models. Hence, we will also be using this to test our model and our code. After training the model on this dataset using our code we can also predict if the cancer tumor is benign or malignant in real life.

3.1.2 Prepossessing the Data set

As a prepossessing step we first ensured that the data set did not have any NULL values. We also ensured that all the features had have only float values so as to remove any inconsistencies. We further did one hot encoding on the target labels(the target variable is "Diagnosis").

3.2 Implementation

For building a neural network we have made a **fully extendable and modular** code. The code along with the documentation can be found at
https://github.com/mehulrastogi/Neural-Network-Assignments

We have used the numpy to build our neural network from scratch. NumPy is the fundamental package for scientific computing with Python. It contains among other things:
- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- useful linear algebra, optimized matrix multiplications[1]

We have implemented a large variety of loss functions,activation function and metrics.The implementation being object oriented and modular the user can choose these when and where in the architecture the user builds.Being modular the user can easily add any activation function,loss function or metrics following the documentation.

The neural network implementation is flexible in the sense that the user can define any number of neurons and any number of layers in the network.The user can also easily change the activation function, loss functions or metrics according to need.

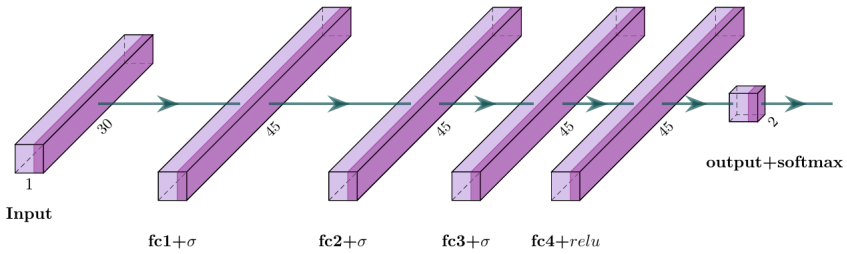
The documentation for the code is available at:-
https://mehulrastogi.github.io/Neural-Network-Assignments/

We have also included a small tutorial using a dummy data set for the user to easily modify the example to fit a custom model and make predictions accordingly.

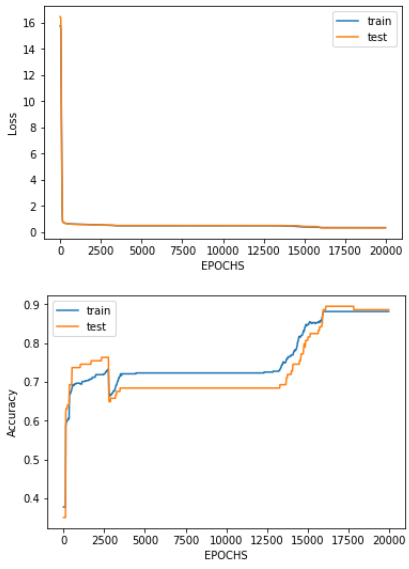
[1]Source : www.numpy.org

3.3 Experiments

We made and used our code for neural network model implementations on the dataset. The dataset is randomly divided into training and testing dataset with 80% of the data as training data and rest as testing data.
First, we experiment the neural network architecture described by the paper i.e. a neural network with four hidden layers, each having 45 neurons. The input layer has 30 neurons corresponding to number of features in the dataset. The output layer has two neurons corresponding to malignant and benign. The first three hidden layers in the neural network have the sigmoid activation function and the last hidden layer has ReLU activation function.

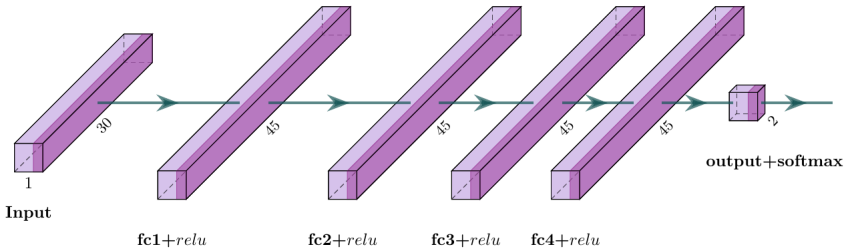


This model is trained for 20000 epochs with a learning rate of 0.00000001 and the standard cross entropy loss and accuracy is reported.

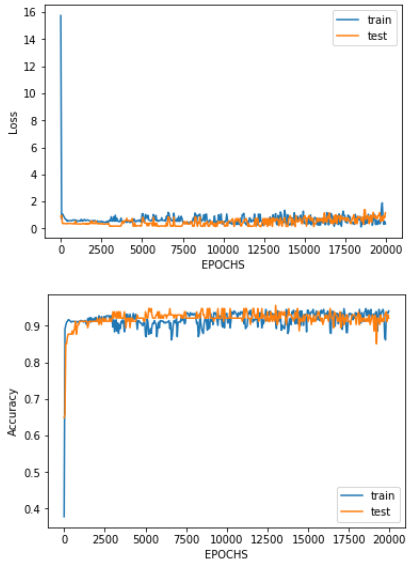


This model achieved an accuracy of 88.6% on the test set and a loss of approximately 0.3 on the test set.

These results can be improved and hence we propose two simple models which give better results and have other advantages as well.
The First model which we propose has the same architecture as the previous model but every activation is changed to ReLU. So this model has an input layer having 30 neurons and four hidden layers having 45 neurons each and finally an output layer having two neurons. Each hidden layer has an activation ReLU.

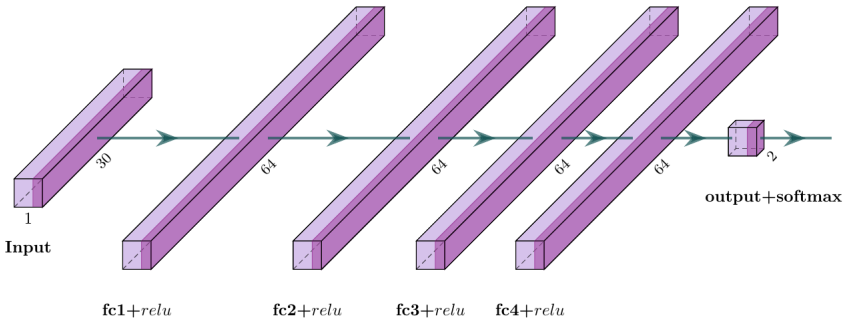


This model is trained for 20000 epochs with a learning rate of 0.00000001 and the standard cross entropy loss and accuracy is reported.

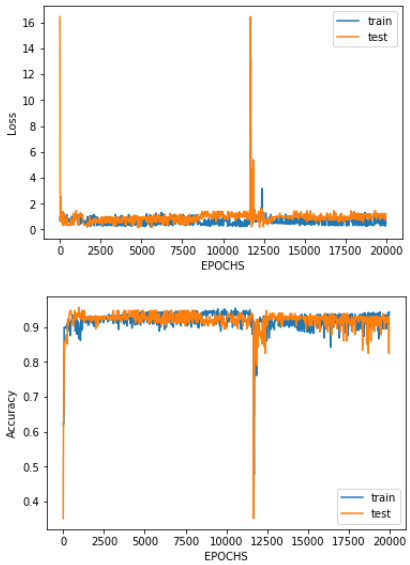


This model achieved an accuracy of 91.2% on the test set and a loss of approximately 0.5 on the test set. This model achieves better accuracy than the previous model and is also computationally less expensive as ReLU activation function requires less computational power and hence this model is more accurate and compuationally cheap which makes it better than the model proposed in the paper.

Now we change the architecture of the previous model by changing the number of neurons to hidden layers to 64 and hence the model has an input layer having 30 neurons, four hidden layers having 64 neurons and finally an output layer having 2 neurons.Each hidden layer has an activation ReLU.



This model is trained for 20000 epochs with a learning rate of 0.00000001 and the standard cross entropy loss and accuracy is reported.



This model achieved an accuracy of 92.9% on the test set and a loss of approximately 1 on the test set. This model has a significant increase in accuracy as compared to the model proposed in the paper and hence is better than it.

3.4 Conclusion

The model they propose does perform fairly good on the Diagnostic Breast Cancer (WDBC) dataset but the results can be significantly improved just by minor changes like changing activation function or changing the number of neurons in layers. We proposed two models which give better results and have other advantages like being computationally inexpensive as well.
The choice of parameters is not mentioned in the paper which leads us to believe that these are not the optimal parameters as they do not provide state-of-the-art results and hence we contribute by modifying the models which provide much better results.