Namal College, Mianwali

# Breast cancer classification using Neural Network Approach

An attempt to come up with a better performing neural network for the classification of breast cancer.

Shahrukh - 14031276
12-2-2017

# Introduction

Breast cancer starts when cells in the breast begin to group out of control, causing a tumor. History shows, breast cancer mainly affects women and is the second leading cause of death for women since ancient Egyptian times [1] [12]. Its diagnosis, treatment and prevention have also been taken care of since then.

Medical organizations all around the world uses many data storage techniques to store their patients' medical records containing the symptoms of the disease and also the related medications. Medical organizations, Researchers and Scientists have been working on this problem for so long to defeat this disease. All of the work on breast cancer classification conducted as yet results in making accurate predictions and those by computer machines. Computer machines are dumb; how could they make predictions? Artificial Intelligence and data mining play a major role in this.

We have to train a neural network on breast cancer data which was obtained from the University of Wisconsin Hospitals, Madisons from Dr. William H. Wolberg and we have Neural Networks from Artificial Intelligence domain.

# Background

## Breast Cancer Classification

Medical application of the classification of the breast cancer is a great challenge for researchers and scientists [3]. There are two classes of cancer type that we will be identifying, Malignant (Cancerous) and Benign (Non-cancerous). A tumor is malignant when cells start to metastasize to other parts of the body and invade surrounding tissues and always considered as cancerous. A tumor is benign when cells do not invade surrounding tissues and such a tumor is not cancerous [2][3].

## Neural Network Models

All of the existing Neural Network Models are inspired by the human brain and are computer programs designed to simulate human brain processing. Neural Networks learn through experience by detecting the patterns and relationships in provided dataset [4].

Any neural network can be thought of as a network of "neurons" which is organized in layers and those layers are defined as follows:

1. Input Layer: It takes predictors/inputs and attaches coefficients to them which are called "Weights".
2. Hidden Layer: It makes the neural network non-linear and improves the performance. Neural network can be linear with no hidden layer.
3. Output Layer: It outputs the results generated from weighted predictors passed through hidden layers, if any.

A general diagram of a linear and non-linear neural network is as follows.

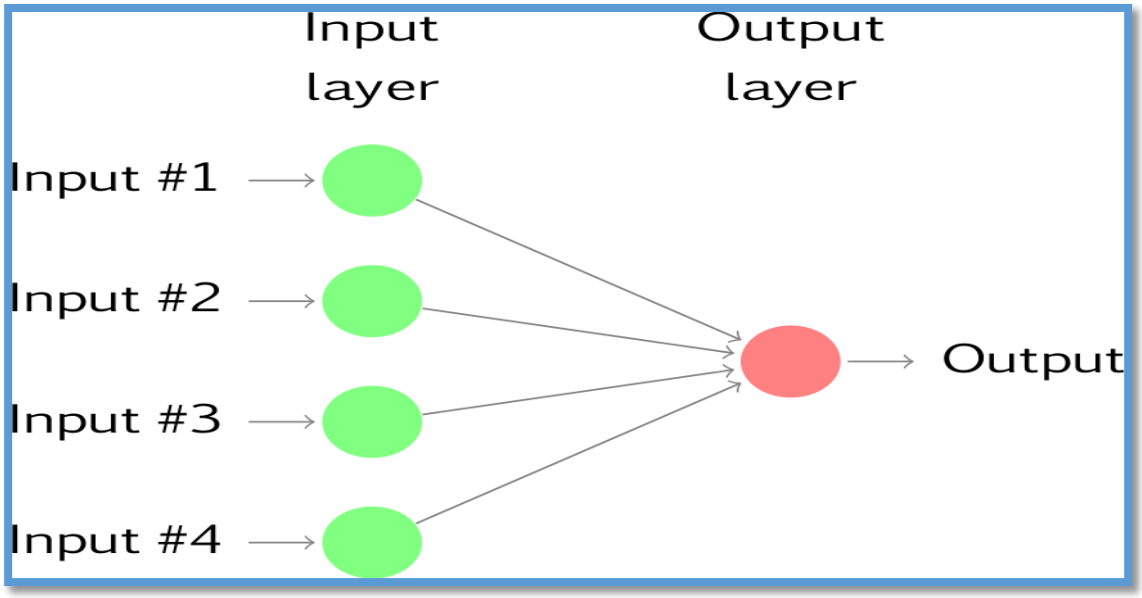



*Figure 1: Linear Neural Network [5]*

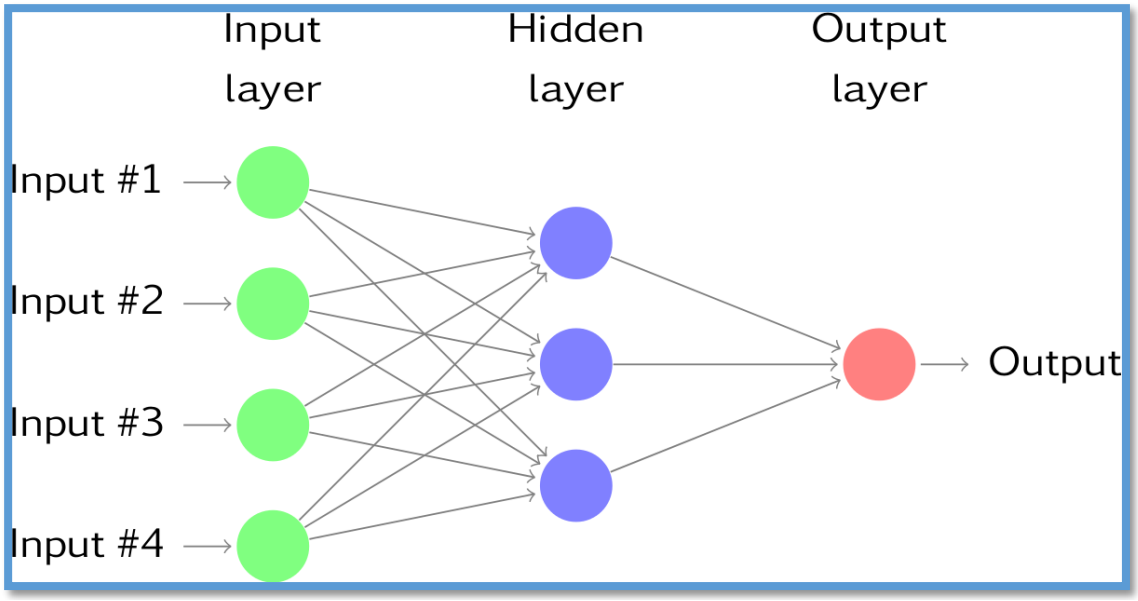



*Figure 2: Non-linear Neural Network [6]*

Linear Neural Network follows the concept of Linear Regression [9]. Non-linear Neural Network is also known as multilayer feed-forward network where each layer of nodes receives inputs from previous layers (hidden layers) [7].

Neural Networks' behavior is determined by its neurons' particular transfer functions, by learning rule and architecture [4]. Transfer function, also called Activation functions, defines the output of given input or set of inputs to a node of neurons.

Applications of the neural networks can be summarized into classification or pattern recognition, prediction and modelling.

## Basic concepts before going through later material

Matlab has been used as a development tool to solve the problem. Reader should have Matlab installed on his computer machine and know the basics of Matlab programming. Following concepts are very useful to understand how the problem has been solved in the later material.

## Importing data from a file

The 'importdata' method has been used to import data from the file. It receives an argument of type string which is the name or path of the file and returns a multidimensional array. You can also get its details in Matlab by using "help" command.

## Array Manipulation

### *Array Creation*

Array creation is as simple as it is in other programming languages. You create a variable and assign it set of values as follows:

**Array = [0,1,2,3,4,5,6,7,8,9]**

### *Array Indexing*

Indexing an array in Matlab starts from 1, so indexing first element of array can be done as Array(1).

Getting particular values from an array is very interesting. If you want to get values from particular row of particular columns, you just define the ranges as follows:

**Array1 = Array(1,2:4)**

where 1 is row number and 2:4 is column range. (Try this in Matlab)

## Creating, Setting, Training, and Testing Neural Network

### *Creating Neural Network with newff*

Following command is used to create network, which can be configured.

**net = newff(Input_Data, Targeted_Data, Hidden_Layers, TF, BTF, BLF, PF);** where [10]:

Input_Data is the dataset we want to train our network on.
Targeted_Data is the resultant of Input_Data for mapping of the outputs.
Hidden_Layers is the variable which defines how many H.L. Neurons to create.
TF is Transfer Function. We set it to {'tansig' 'tansig'}.
BTF is Backpropagation Training Function. We set it to 'trainr'.
BLF is Backpropagation Learning Factor. We set it to 'learngd'.
PF is Performance Function. We set it to 'mse'.

### *Setting Parameters of the network, returned by newff*

There are many parameters of the network we get from newff but in later material, only following parameters are set.

**net.trainParam.epochs** is used to set maximum numbers of epochs/iterations to train the network [11].

**net.trainParam.goal** is used to set the performance goal [11].

**net.trainParam.max_fail** is used to set maximum validation failures [11].

These parameters are used while the training of the network.

### *Training of the configured network*

Following command is used to train the network.

**net = train(net, Input_Data, Targeted_Data);**

This command returns a trained network on the provided dataset which further can be used for testing or real application.

### *Testing the trained network*

Following command is used to test the trained network which returns the set of outputs corresponding to the set of inputs.

**results = net(Testing_Data);**

where: Testing_Data is the dataset on which we want to test our neural network.

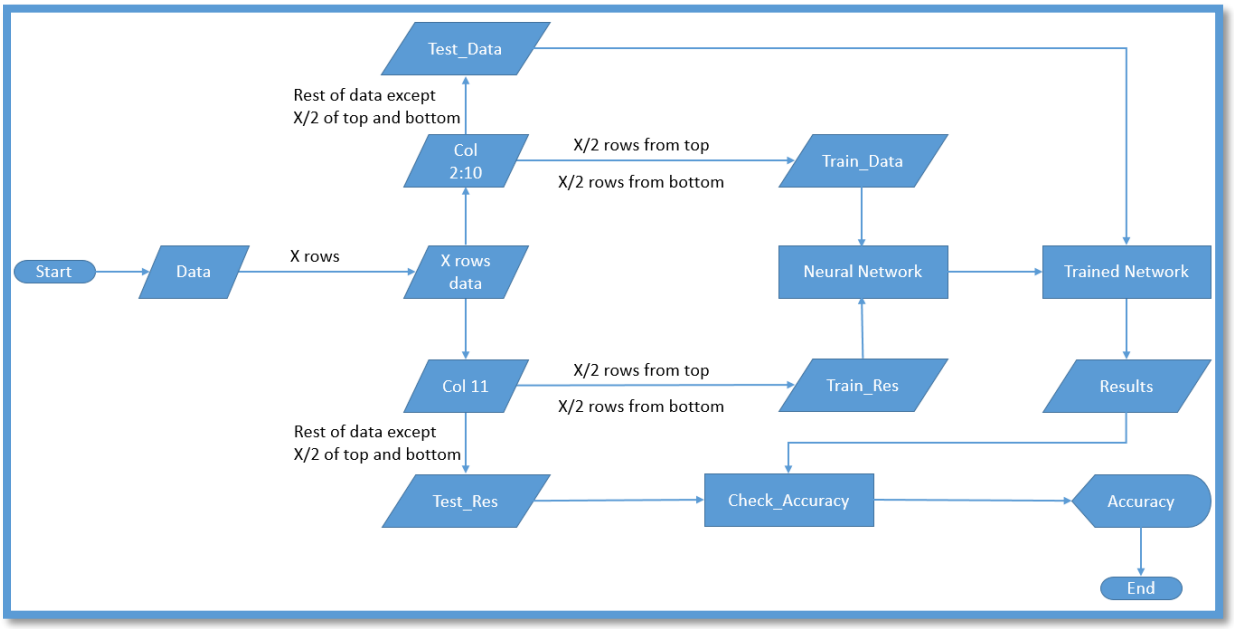
# Methodology Description

Following subsection covers the preprocessing steps on data and how do those steps help to minimize the efforts and simplify the problem.

## Preprocessing steps on provided data

1. There are 16 lines which have '?' as one of their values. '?' can be replaced with any non-negative integer but it would make data irregular and uneven. So removing all of those lines which contain '?' would prevent our data from being irregular and uneven.
2. You may observe two unique values in the last column of each row, that either be 2 (Benign) or 4 (Malignant). Last row would be the targeted data for corresponding input/training data. In the experiments to be conducted, you would be picking some amount of rows for training data, containing any percentage of both values, preferably 50% of the size of training data to train our network for both cases with an equal amount of data.
3. How to pick 50% of each values for our training data from main data? There are many ways and simple way is to iterate through the main data and pick values and maintain counters. The interesting way is to sort the data out on last column and pick desired amount of values from top and bottom. This sorting method will increase your code efficiency and also simplify your problem.
4. There are 11 column values in each rows, first column is ID that can be used or not, eleventh column contains targeted values of each row and in between is the data to train or test our neural network on.

## Flow Diagram




## Neural Network Model

To solve the problem, we are using a Multi-Layer Perceptron - a Feedforwarding Neural Network which uses Backpropagation method. You must know following two definitions to understand this neural network model.

### Feedforward Neural Network

This is an Artificial Neural Network which has individual units (neurons/nodes) connected together in such a fashion that they do not form a cycle [13].

### Multi-Layer Perceptron

It is a neural network which consists of multiple layers of units which applies computational processes (especially Sigmoid function – activation function) on any input and those layers are usually interconnected in a feedforwarding fashion.

### Backpropagation Method

This method helps in calculating the error contribution of each neuron in the neural network after a batch of data is processed [8].

## Experimental Results and Analysis

This section will cover the experiments conducted to train neural network.

**0**

| Run Configurations | | | | |
|---|---|---|---|---|
| **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| 100 | 100 | 100 | 0.1 | 10 |

**Hypothesis**

N/A

**Results**

| Accuracy | Time | Epochs Completed | Goad Met |
|---|---|---|---|
| 90.595% | 7 secs | 22 | 0.097 |

**Conclusion**

This is a test experiment to make some configuration our base of experiments.

**1**

| Run Configurations | | | | |
|---|---|---|---|---|
| **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| 400 | 100 | 100 | 0.1 | 10 |

**Hypothesis**

Previously we had a small amount of training data. With an increased amount of training data, it is for sure that the accuracy will improve. Rest of the configurations are kept the same.

**Results**

| Accuracy | Time | Epochs Completed | Goal Met |
|---|---|---|---|
| 96.466% | 21 secs | 16 | 0.095 |

**Conclusion**

The hypothesis results to be true and accuracy has improved. This is because our network has been trained on a reasonably good amount of data. It can be seen that this configuration takes more time to train the network and we need to come up with a trained network which is time efficient with high accuracy rate.

**2**

| Run Configurations | | | | |
|---|---|---|---|---|
| **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| 400 | 10 | 10 | 0.1 | 10 |

**Hypothesis**

Our previous network gives us a better accuracy rate but does not seem to be time efficient, so to make it time efficient previous configurations have been modified. Epochs (Iterations) amount has been decreased as it can be seen that epochs were set to 100 and the training completed in 16. Max_Fail is changed to smaller value but it will not affect overall performance of the network as it was seen in the experiment that Max_Fail (Validation Checks) were not going above 10. It is assumed that with lesser iterations, network will be more time efficient and maybe with higher accuracy rate.

**Results**

| Accuracy | Time | Epochs Completed | Goal Met |
|---|---|---|---|
| 96.8198% | 12 secs | 10 | 0.105 |

**Conclusion**

| | |
|---|---|
| | Hypothesis results to be true with slight improvement in the accuracy and a really good improvement in time but the goal is slightly away from expected goal. With current configurations, the goal must be a little higher. |

| | | | | | |
|---|---|---|---|---|---|
| 3 | **Run Configurations** | | | | |
| | **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| | 400 | 10 | 5 | 0.11 | 10 |
| | **Hypothesis** | | | | |
| | Keeping the results in the view from previous experiment, it has been concluded to set goal a little higher. It will not only meet all the expected configurations but also improve the time efficiency and accuracy. Max_Fail value is decreased keeping in view the previous experiment value of it in results and it will not affect the performance as it did not earlier. | | | | |
| | **Results** | | | | |
| | **Accuracy** | **Time** | **Epochs Completed** | **Goad Met** | |
| | 94.347% | 6 secs | 7 | 0.109 | |
| | **Conclusion** | | | | |
| | The accuracy has decreased whereas time efficiency has improved. This is a great improvement but accuracy needs to improve too. Accuracy could not improve because of a low data size for the set Goal. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 4 | **Run Configurations** | | | | |
| | **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| | 500 | 10 | 5 | 0.11 | 10 |
| | **Hypothesis** | | | | |
| | As already seen, increasing the data increases the performance. With the same previous configuration but with an increase in the training data size, the accuracy rate will increase. | | | | |
| | **Results** | | | | |
| | **Accuracy** | **Time** | **Epochs Completed** | **Goad Met** | |
| | 98.907% | 6 secs | 6 | 0.104 | |
| | **Conclusion** | | | | |
| | The accuracy has increased at a good rate and time delay is same. Epochs and Goal Met have improved too because of increased data. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 5 | **Run Configurations** | | | | |
| | **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| | 600 | 10 | 5 | 0.11 | 10 |
| | **Hypothesis** | | | | |
| | As previously noted that with increase in the amount of data, our accuracy increases. This is for sure that in this experiment with an increased amount of data, our performance will increase at a better rate again. | | | | |
| | **Results** | | | | |
| | **Accuracy** | **Time** | **Epochs Completed** | **Goad Met** | |
| | 98.795% | 10 secs | 6 | 0.104 | |
| | | | | | |

| | Conclusion |
|---|---|
| | Accuracy has decreased at a very small rate but time delay, because of increased amount of data, has increased while all of the other expectations have met. |

| | | | | | |
|---|---|---|---|---|---|
| 6 | **Run Configurations** | | | | |
| | **Training Data Size** | **Epochs** | **Max_Fail** | **Goal** | **Hidden Layers** |
| | 600 | 10 | 5 | 0.11 | 20 |
| | **Hypothesis** | | | | |
| | Accuracy was decreased in last experience. Let's try increasing the hidden layer neurons and with increased neurons in the layer, the accuracy of the network will increase as there would be more activation functions applied. | | | | |
| | **Results** | | | | |
| | **Accuracy** | **Time** | **Epochs Completed** | **Goad Met** | |
| | 98.795% | 5 secs | 3 | 0.089 | |
| | **Conclusion** | | | | |
| | The accuracy has neither increased nor decreased but time efficiency has increased and training took less Epochs and Goal met has improved too. So, hidden layer neurons increment increases other factors but accuracy. | | | | |

Later few experiments were conducted and increment in hidden layer neurons decreased the performance and with increase in data, the accuracy was increased but so far, best performance configurations are in $4^{th}$ and $6^{th}$ experiments. Anyone of them can be used to get better results.

## Conclusion

The attempt has been made to develop an accurate and time efficient neural network for the classification of the breast cancer. This coursework was very useful and helpful in understanding the concepts of neural networks in details. I look forward to learn more about neural networks and come up with at least a solution to any existing unsolved or partially solved problem.

# References


[1] Brechon, S. (2012). A Brief History of Breast Cancer. [online] Maurer Foundation. Available at: https://www.maurerfoundation.org/a-brief-history-of-breast-cancer/ [Accessed 27 Nov. 2017].

[2] Bollinger, Ty. (2016). Benign and Malignant: What is the difference? . [online] The Truth About Cancer. Available at: https://thetruthaboutcancer.com/benign-malignant-tumors-difference/ [Accessed 27 Nov. 2017].

[3] Raad, A., Kalakech, A. and Ayache, M. (2013). BREAST CANCER CLASSIFICATION USING NEURAL NETWORK APPROACH: MLP AND RBF. [ebook] The 13th International Arab Conference on Information Technology. Available at: http://www.acit2k.org/ACIT/2012Proceedings/13233.pdf [Accessed 1 Dec. 2017].

[4] Agatonovic-Kustrin, S. and Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. Volume 22(Issue 5), pp.717-727.

[5] Open-Access Textbooks (n.d.). Linear Neural Network. [image] Available at: https://www.otexts.org/sites/default/files/fpp/images/nnet1.png [Accessed 1 Dec. 2017].

[6] Open-Access Textbooks (n.d.). Linear Neural Network. [image] Available at: https://www.otexts.org/sites/default/files/fpp/images/nnet2.png [Accessed 1 Dec. 2017].

[7] Open-Access Textbooks (2017). 9.3 Neural network models. [online] Available at: https://www.otexts.org/fpp/9/3 [Accessed 3 Dec. 2017].

[8] Nielsen, M. (2017). Neural Networks and Deep Learning. [online] Neuralnetworksanddeeplearning .com. Available at: http://neuralnetworksanddeeplearning.com/chap2.html [Accessed 2 Dec. 2017].

[9] O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

[10] Radio.feld.cvut.cz. (2017). newff (Neural Network Toolbox). [online] Available at: http://radio.feld.cvut.cz/matlab/toolbox/nnet/newff.html [Accessed 1 Dec. 2017].

[11] Mathworks.com. (2017). Levenberg-Marquardt backpropagation - MATLAB trainlm - MathWorks United Kingdom. [online] Available at: https://www.mathworks.com/help/nnet/ref/trainlm.html [Accessed 1 Dec. 2017].

[12] Cheng, H., Shan, J., Ju, W., Guo, Y. and Zhang, L. (n.p.). Automated breast cancer detection and classification using ultrasound images: A survey.

[13] Research Gate. (2011). Application of a Modular Feedforward Neural Network for Grade Estimation. [online] Available at: https://www.researchgate.net/publication/225535280_ Application_of_a_Modular_Feedforward_Neural_Network_for_Grade_Estimation [Accessed 1 Dec. 2017].