Breast Cancer Detection using Neural Networks

Neural Networks and Fuzzy Systems Coursework-I

Muhammad Usman

UoB Number: 12036782

Submitted to

Dr. Malik Jahan Khan

13/11/2015

Department of Computer Science

Namal College Mianwali - Pakistan

www.namal.edu.pk

Abstract

Self-directing ability in the software was the stimulus that guided the world to 'Autonomous systems'. Prime objective of these systems is to make the current software self-adaptive so that these systems become independent from human interaction. Breast cancer detection is an open area of study that need to be tackled by these self-governance technologies. To fulfill this, many methods have been used to make such systems that can help in diagnosing this particular disease. Artificial neural network is also one of the intelligent tools that are used for this purpose. Neural networks are exposed to a training data set to be trained in such a way that injects self-adaptiveness in them. This report provides a solution to 'Breast cancer data classification' using an artificial neural network. The results are gathered from different architectures of neural networks and finally a solution is proposed that is able to predict the class of a new case with 97 to 99% accuracy depending upon the distribution of training data.

Contents

List of Figures

List of Tables

1 Introduction

Despite the progress of modern medical technologies, breast cancer is still one of the widely spread diseases of the modern era and it is the second major cause of cancer deaths in women [5]. Therefore, remarkable efforts are required to fight this enemy. To alleviate the harm caused by this severe disease and to cease it from spreading so swiftly, several disciplines have made their contributions. Specifically, in the last few decades artificial intelligence and data mining have been used immensely to develop such tools and expert systems that can assist physicians in the diagnosis of the mentioned illness [2],[9]. Likewise, the use of neural networks to flourish such intelligent systems that can learn by the passage of time, and can make quick decisions for breast cancer diagnosis is being extensively adopted [7],[10],[13], [14]. This report contains the process followed in developing such network that can learn from the dataset and can make intelligent decisions on an unseen case on the basis of learning from training data.

2 Background and related work

2.1 Artificial neuron

Artificial neuron is basic building block of neural networks. The basic model of an artificial neuron is shown in the Figure 1

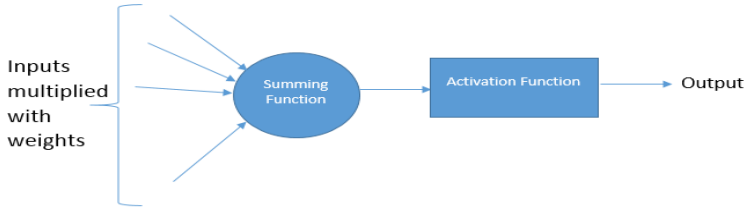
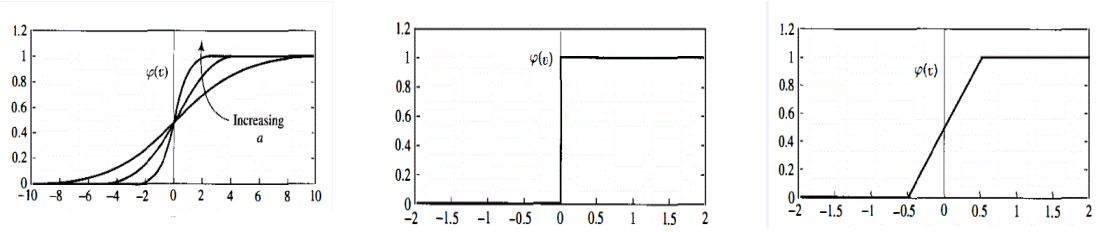


Figure 1: Basic Model of an artificial neuron

Set of inputs (synapsis) are applied to the neuron and each of those inputs have some weight or strength. These inputs, after being multiplied with the corresponding weight (randomly picked initially), are added together by the summing junction and output is forwarded to activation function that limits the output in a certain domain. There are several activation (transfer) function that generate the actual output of the neuron. For example, binary threshold does the binary classification and gives 1 or 0 as output, Figure 2b is showing its graph. Also, piecewise linear function, shown in Figure 2c, maps the summing junctions output to the output of neuron linearly, that is, output remains zero for some particular threshold, then it increases linearly and after reaching at one, it remains constant. Likewise, Sigmoid is also an activation function shown in Figure 2a. Domain of output lies between zero to one with a smooth curve [3].



(a) Sigmoid (b) Binary Threshold (c) Piecewise Linear

Figure 2: Activation functions behaviour

[3]

2.2 Artificial Neural network overview

Network Architectures

An artificial Neural network is actually a combination of neurons connected in a specific order. It has analogy with directed weighted graphs where neurons of the network would become node and edges are the connections between neurons that make output of one neuron as input of other. Based on the connection pattern, there are two possible classes of neural networks: [11]

- Feed forward networks (No loops)
- Recurrent (Feedback) networks (Loops exist)

Feed forward networks are considered to be static. That is, state of the neurons in feed-forward network are only dependent on input-output mapping, and there is no involvement of previous state of neuron. Y.M. Chiang et al.[4] discussed that advantage of using feedforward network is that it is easy to implement. But, it may fail to provide a satisfactory solution for the cases where testing data have major changes than training data. Also, it can easily fall into a local minima and result with a wrong convergence, and in case of large input data, slow convergence[4]. The mentioned work have not discussed the use of 'traingdm', a refined form of 'traingd' that could be helpful in the scenario having different local and global minima.

On the other hand, feedback networks are more dynamic, as the state of a neuron is not only depending on input, but also on the previous states of neuron. Also, feedback link in the network improves the training time. Contrarily, these networks are often not stable in results[4]. Moreover, both of these architectures could be used for classification problems [11].

Further hierarchy of these architectures is explained in Figure 3.

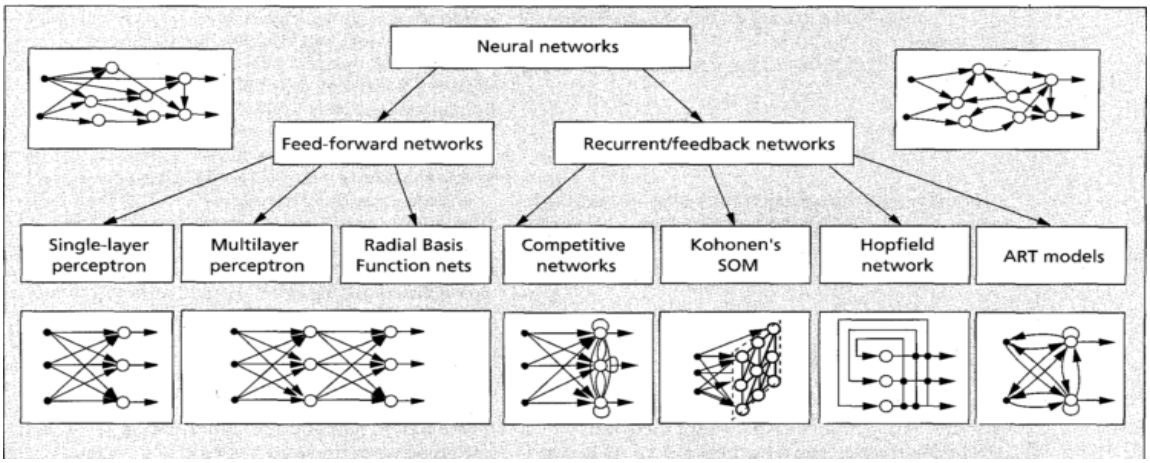


Figure 3: Architecture tree

[11]

Number of hidden layers and hidden layer neurons

After network architecture, another important aspect is the number of hidden layers and hidden neurons in multi-layer perceptron. Also, number of neurons in input and output layers would be equal to number of inputs and outputs respectively. Moreover, theoretically, it is not possible to detect number of neurons in each hidden layer or number of hidden layers[1]. Generally, one or two hidden layers provide better convergence and performance. Also, using more than two hidden layers produce many local minima[4].

2.3 Neural network training

Neural networks were emerged after recognizing the fact that human brain does the computation in a completely different manner than a digital computer[3]. Our brain learns from the environment and takes decisions using prior knowledge. Similarly, artificial neural networks can be trained, and could be useful in dealing with unseen cases.

Neural network training methods

Training method is one of the most important factor that influences the training speed and the performance of the network. [1] The basic purpose of a training algorithm is to update the weights in such a way that can reduce the error and help the system to converge to the target. Particularly, in supervised learning, the desired output 'd' is provided to the network and network itself predict an output 'y' using the initial random weights with inputs. Based on the difference between 'y' and 'd', weights are updated according to Equation 1. [11]

$$w_i(t+1) = w_i(t) + \delta(d-y)x_i \tag{1}$$

Common learning algorithms for feedforward networks include standard back propagation algorithm and conjugate gradient. And, real-time recurrent learning for

dynamic feedback networks.[4]

2.4 Related Work

Classification of breast cancer is a prominent problem in 'Soft Computing'. Specifically, a lot of work had been done in neural networks for this particular problem. Senapati et. al(2013) used local linear wavelet neural network and 'Firefly algorithm' to find the solution using Wisconsin Diagnostic Breast Cancer (WDBC) dataset. The mentioned work resulted with an overall classification accuracy of 98.14% with 99.6% accuracy in Malignant tumors and 97.4% accuracy in Benign tumors[13]. But, low accuracy in Benign cases could be very dangerous in Cancer classification problem. Another approach using the same database was provided by El-Sebakhy et al.(2006) in which 'Functional networks' were proposed as a classifier scheme of breast cancer classification. Using the same dataset (WDBC), they have reported a classification accuracy of 96.8%[6]. Another interesting finding was done by Karabatak et al.(2009), they have used 'Association rules' with neural networks to automate a diagnosis of breast tumor. Specifically, association rules were used as a pre-processing tool to reduce the dimension of data, and after doing the pre-processing step, neural networks were used and 97.40% accuracy was found[8].

3 Main Part

This section will elaborate the sequence of steps followed to solve the mentioned problem. Firstly, data was gathered from WDBC database and preprocessing(data cleaning) was done on the given data. Then, different neural network architectures were tested to get the optimum result.

3.1 Pre-processing

Data gathering and cleaning

Data was collected from WDBC database, and output values were substituted as 4 for malignant and 2 for benign, as output in numbers make it easier for the network to be trained. Also, after analyzing, missing values were detected at 16 instances out of 699. Also, all the missing values were in the same attribute(column). Moreover, it was mentioned in the dataset description that the first attribute is representing ID number, it was removed from the data. It resulted with the nine attributes in the data. Then, the next problem was to handle the missing values. To tackle this issue, literature review was done again. E. Pesonen et al. have suggested following methods for handling the missing values:

- Substituting means
- Substituting random values
- Substituting nearest neighbour

[12]

Moreover, replacing the missing value with the one that is the most similar to that case (nearest neighbour) method was showing good results, so this method was picked for handling missing values[12]. To find the nearest method, Euclidean distance formula was used. The formula is given below:

$$d(x, y) = \sqrt{\sum_{a=1}^{n}(x_a - y_a)^2}. \tag{2}$$

Here, x and y are the two cases and n is number of attributes in them. [15]

Matlab code was written using the mentioned formula and the code detected the nearest neighbour for each missing valued case. The portion of Matlab that used the mentioned formula is:

diff = dataToClean(i,[2 3 4 5 6 8 9 10 11]) - dataToClean(i2,[2 3 4 5 6 8 9 10 11]);
diff = diff.$\hat{2}$; % square each element
diff = sum(diff);
diff = sqrt(diff);

Complete code used for data cleaning is attached in Appendix A, and output of the code is given in Appendix B. Using this output, each missing value was replaced by its nearest neighbour. Also, data was sorted with respect to output.

3.2 Network Architecture

As mentioned in the previous section that classification problems could be solved using both static and dynamic neural networks. So, it was decided to check both of these architectures. Also, it is mentioned that there is no specific rule to detect number of hidden layers, or number of neurons in the hidden layer. Therefore, different combination of hidden layers and hidden neurons was used.

3.3 Solution procedure

After cleaning the data, Matlab code was written to solve the problem. Code is attached in appendix C. Firstly, in the code, training data was loaded and a neural network was created according to number of inputs and desired output. After creating the neural network, it was trained on the training data. When it was trained, testing data was loaded, and the network was simulated on the testing data.

3.4 Post Processing

The usage of 'purelin' as activation function of output layer was giving the optimum results, but it was predicting some values as 3 which was disturbing the binary classification. To deal with this issue, following piece of code was written to force the result in binary values (complete code is attached in appendix C):

```
function result = binaryRound(result)
 [rows,cols] = size(result);
          for i=1:rows
          for j=1:cols
               if result(i,j)<3.000000
                    result(i,j)=2;
               else
                    result(i,j)=4;
               end
             end
          end
          end
end
```

3.5 Results and Analysis

Different combination of training and testing data was used. Also, Accuracy (calculated by 3), Recall (calculated by 4) and Precision (calculated by 5) were used as an indicator of neural network training. Benchmark accuracy, obtained by Senapati et al, was **98.14%** with almost 57% of data as training data using WDBC dataset. But the mentioned work had high false positive detection with accuracy for benign tumor as 97.4%[13]. All of the following results were being performed three times and an average is stated here. Moreover, number of iterations were kept 3000 for every experiment. The experimental results of every iteration are given below:

$$Accuracy = \frac{a+d}{a+b+c+d} \tag{3}$$

$$Recall = \frac{a}{a+c} \tag{4}$$

$$Precision = \frac{a}{a+b} \tag{5}$$

Here,

- a is representing such instances where both actual and predicted values are malignant (True Positive)
- b is representing those cases where actual is benign and predicted is malignant (False Positive)
- c is indicating those cases where actual is malignant and predicted is benign (False negative)
- d is indicating those instances where both actual and predicted values are benign (True Negative)

Effect of Data distribution

Training data	Testing data	Activation functions	Accuracy	Precision	Recall
95 %	5 %	"tansig,purelin"	100%	100%	100%
90 %	10 %	"tansig,purelin"	98.50%	100 %	98%
85 %	15 %	"tansig,purelin"	99 %	100 %	97%
80 %	20 %	"tansig,purelin"	98.5%	97.8%	97.8%
70 %	30 %	"tansig,purelin"	99 %	98.6%	98.6%
50 %	50 %	"tansig,purelin"	98.8%	98.3%	98.3%
60 %	40 %	"tansig,purelin"	98.5%	96.8%	98.9%
40 %	60 %	"tansig,purelin"	98.5%	98%	97%
30 %	70 %	"tansig,purelin"	97%	98.5%	92.5%
20 %	80 %	"tansig,purelin"	96.9%	98.9%	92.6%
10 %	90 %	"tansig,purelin"	95%	94.9%	91.1%

Table 1: Effect of Data distribution: Number of Hidden layer neurons are 7, traingdm as Back-propagation algorithm and activation functions are 'tansig,purelin'

Training data	Testing data	Activation function	Accuracy	Precision	Recall
95%	5%	"tansig, purelin"	100%	100%	100
90%	10%	"tansig, purelin"	98.50%	100	95.8
50%	50%	"tansig, purelin"	98%	98	98
20%	80%	"tansig, purelin"	97%	98	93

Table 2: Effect of Data distribution 2: Number of Hidden layer neurons are 10 and traingdm as Back-propagation algorithm

Table 1 and 2 are indicating that with the increase in quantity training data, neural network is improving its performance most of the time.

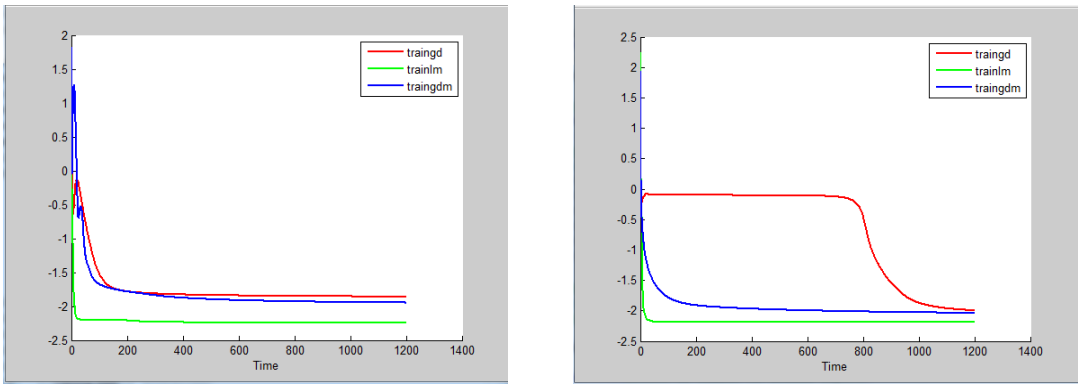
Back Propagation Algorithm

Amongst the Back Propagation algorithm, 'traingdm', 'traingd' and 'trainlm' were used to see how the results are changing for each of them [Table 3].

Training data	Testing data	BP Algo	Accuracy	Precision	Recall
40%	60%	traingdm	98.3%	98.6%	95.5%
40%	60%	traingd	98.3%	98.6%	96.8%
40%	60%	trainlm	97.8%	97.91%	95.9%
60%	40%	traingdm	98.5%	96.8%	98.9%
60%	40%	traingd	98.5%	96%	98.9%
60%	40%	trainlm	95.8%	92.91%	96%

Table 3: Effect of Back Propagation Algorithm: Number of Hidden layer neurons are 4 with Feed Forward neural network and activation function 'tansig','purelin'

The results were interesting, learning rate using 'trainlm' was very high using both of the data distribution (Figure 1). But, using 60% of the data as training data reduced the classification rate of 'trainlm'. The reason for this could be overfitting of the network, i. e. network was overtrained on this amount of training data.



(a) Training data is 40%

(b) Training data is 60%

Figure 4: Effect of learning algorithm on learning speed

Effect of number of Hidden layers

The following section will cover the effect of number of layers in neural network on its training. For this experiment, fixed data distribution had been used with 50% training and 50% testing data, 'traingdm' as back-propagation algorithm, and 'purelin' as activation function for output layer [Table 4]:

Table 4 is indicating that by changing the number of hidden layers, there are no drastic changes in the accuracy, precision, or recall. In the case of 65% accuracy, all of the values were predicted as benign.

Hidden layers	Neuron in Hidden layers	Activation function	Accuracy	Precision	Recall
2	1,1	tansig,purelin	65%	0	0
1	1	tansig	65.8%	0	0
2	2,2	tansig	97.1%	97.4%	96.5%
1	2	tansig	65.8%	0	0
2	3,3	tansig,purelin	65.8%	0	0
2	3,4	tansig,purelin	98.5%	97.4%	98.3%
1	3	tansig	98.8%	98.3%	98.3%
2	4,4	tansig,purelin	98.8%	97.5%	98.3%
1	4	tansig	98.8%	98.3%	98.3%
2	7,7	tansig,tansig	98.2%	97.4%	98.4%
1	7	tansig	97.90%	97.4%	96.6%

Table 4: Effect of Number of layers:

Effect of number of Neurons in Hidden layer

This section will elaborate the the impact of number of neurons on training of our neural network. Number of hidden layer was 1 for the following experiment (Table 6) and data distribution was kept 70 and 30 as training and testing respectively:

Activation function	Number Of Neurons	BP Algo	Accuracy	Precision	Recall
tansig, purelin	1	traingdm	65.8%	0	0
tansig, purelin	2	traingdm	65.8%	0	0
tansig, purelin	3	traingdm	99%	98.5	98.5
tansig, purelin	4	traingdm	99.3%	99	98.5
tansig, purelin	6	traingdm	99%	98.5	98.5
tansig, purelin	7	traingdm	99%	98.5	98.5
tansig, purelin	10	traingdm	99%	98.5	99

Table 5: Effect of Number of neurons in Hidden Layer: Data division is 70 (training) 30 (testing)

Then, the same experiment was done using data distribution as 40 (training) and 60 (testing) to assure the previous results.

This was observed in 6,5 that for 3 to 10 number of neurons in the neural network, there was not much difference in accuracy of the predicted cases. But, with 1 neuron, it was hard for the neural network to train itself on the data. Also, when we decrease the training data, neural network with less neurons took more time to learn. The graph in Figure 5 is showing the fact that for 40% data as training data, the network with 6 neurons was quickly trained as compared to the one having 3 neurons in its hidden layer (1200 epochs).

Number Of Neurons	BP Algo	Accuracy	Precision	Recall
1	traingdm	65.6%	0	0
3	traingdm	98.5%	98.4%	97.2%
4	traingdm	98.2%	97.2%	97.9%
5	traingdm	98.4%	98.2%	97.7%
6	traingdm	98.5%	98.6	96.5
7	traingdm	97.8%	98.6	97.5
8	traingdm	98.3%	98.6	96.5
9	traingdm	98%	98.5	97.3

Table 6: Effect of Number of neurons in Hidden Layer: Data division is 40 (training) 60 (testing), activation functions are 'tansig,purelin' 3000 epochs

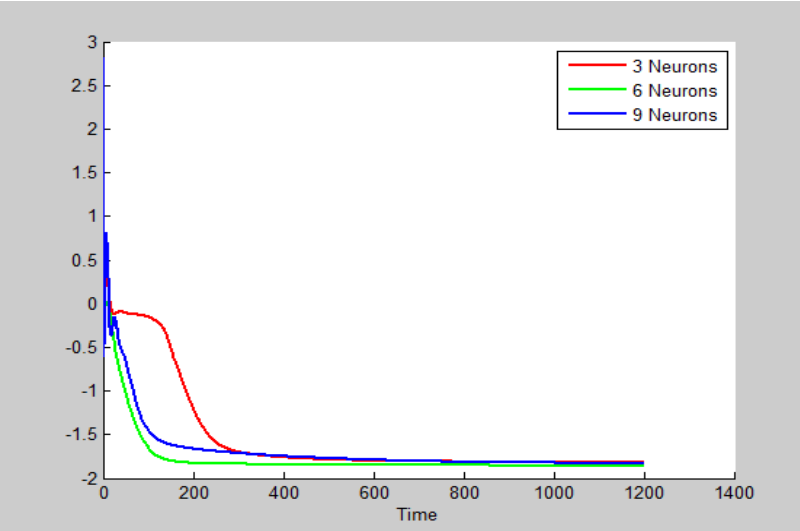


Figure 5: Performance comparison with number of neurons in hidden layer

Effect of Activation/Transfer Functions

For this experiment, data distribution was kept 50 (training) and 50 (testing) and number of neurons in hidden layer were 7. Table 7 covers the results of changing activation functions:

So, the combination of 'tansig' as activation function of hidden layer and 'purelin' as activation function of output layer was giving the best results.

Activation function	BP Algorithm	Accuracy	Precision	Recall
tansig, tansig	traingdm	65.8%	0	0
purelin, purelin	traingdm	34.19%	100%	34.19%
purelin, tansig	traingdm	65.8%	0	0
tansig, purelin	traingdm	98.3%	98.3%	98.3%

Table 7: Effect of Activation functions: Data division is 70 (training) 30 (testing)

4 Conclusion

Neural networks are being widely used as a tool for Breast tumor classification. This work was focused on usage of different approaches of neural networks to achieve the better results. Also, the classification accuracy was dependent on the data distribution and with 90% of data as training data, even 100% accuracy was found. But, having a fair distribution of 60-40 with 60% as training data was able to get an average accuracy of 98%. Also, it was analyzed that different learning algorithm had influenced the results in terms of training speed with 'trainlm' having the quickest learning rate. But, classification rate of 'traingd' and 'traingdm' was better. Therefore, it is proposed that proposed neural network should have one hidden layer with 7 neurons using 'traingdm' as back-propagation algorithm to achieve a better classification accuracy. But, the overall performance would still be influenced by some other factors.

Bibliography

[1] P. Benardos and G.-C. Vosniakos. Optimizing feedforward artificial neural network architecture. *Engineering Applications of Artificial Intelligence*, 20(3):365–382, 2007.

[2] A. ÇakIr and B. Demirel. A software tool for determination of breast cancer treatment methods using data mining approach. *Journal of Medical Systems*, 35(6):1503–1511, 2011.

[3] I. Casas. *Neural networks*. 2009.

[4] Y.-M. Chiang, L.-C. Chang, and F.-J. Chang. Comparison of static-feedforward and dynamic-feedback neural networks for rainfall–runoff modeling. *Journal of hydrology*, 290(3):297–311, 2004.

[5] C. Desantis, R. Siegel, P. Bandi, and A. Jemal. Breast Cancer Statistics , 2011. *Cancer*, 61(6):409–418, 2011.

[6] E. A. El-Sebakhy, K. A. Faisal, T. Helmy, F. Azzedin, A. Al-Suhaim, et al. Evaluation of breast cancer tumor classification with unconstrained functional networks classifier. In *AICCSA*, pages 281–287, 2006.

[7] M. Karabatak and M. C. Ince. An expert system for detection of breast cancer based on association rules and neural network. *Expert Systems with Applications*, 36(2 PART 2):3465–3469, 2009.

[8] M. Karabatak and M. C. Ince. An expert system for detection of breast cancer based on association rules and neural network. *Expert Systems with Applications*, 36(2):3465–3469, 2009.

[9] A. Kele, A. Kele, and U. Yavuz. Expert system based on neuro-fuzzy rules for diagnosis breast cancer. *Expert Systems with Applications*, 38(5):5719–5726, 2011.

[10] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast Cancer Diagnosis and Prognosis Via Linear Programming. *Operations Research*, 43(4):570–577, 1995.

[11] J. Mao. Why artificial neural networks? *Communications*, 29:31–44, 1996.

[12] E. Pesonen, M. Eskelinen, and M. Juhola. Treatment of missing data values in a neural network based decision support system for acute abdominal pain. *Artificial Intelligence In Medicine*, 13:139–146, 1998.

[13] M. R. Senapati and P. K. Dash. Local linear wavelet neural network based breast tumor classification using firefly algorithm. *Neural Computing and Applications*, 22(7-8):1591–1598, 2013.

[14] H. T. Tike Thein and K. M. Mo Tun. An Approach for Breast Cancer Diagnosis Classification Using Neural Network. *Advanced Computing: An International Journal*, 6(1):1–11, 2015.

[15] H. Wang. Nearest neighbors by neighborhood counting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(6):942–53, 2006.

5 Appendix A

```
function x = dataCleaning()

load 'dataToClean.txt';                    % loading the data

data = dataToClean(:,2:11);
[rows,cols] = size(data);

minIndex = 1;
for i=1:rows
        if dataToClean(i,7) == 0           % This row has missing Value
            minDistance = 9999999;         % assuming this is the minimum di
            for i2=1:rows
                    if i2 ~= i
                        diff = dataToClean(i,[2 3 4 5 6 8 9 10 11])
                        - dataToClean(i2,[2 3 4 5 6 8 9 10 11]);
                        diff = diff.^2;        % square each element
                        diff = sum(diff);
                        diff = sqrt(diff);
                        if diff < minDistance && dataToClean(i2,7) ~= 0
% new computed difference is less than previous
                            minDistance = diff;
% Update the minimum distance
                            minIndex = i2;
% Update the index of min for a particular missing value
                        end
                    end
            end
      S=sprintf('Replace the missing value at %d with %d', i, minIndex);
        disp(S)
        end
    %end
end
x = 1;
end
```

6 Appendix B

```
>> dataCleaning
Replace the missing value at 24 with 224
Replace the missing value at 41 with 318
Replace the missing value at 140 with 315
Replace the missing value at 146 with 663
Replace the missing value at 159 with 521
Replace the missing value at 165 with 1
Replace the missing value at 236 with 616
Replace the missing value at 250 with 3
Replace the missing value at 276 with 35
Replace the missing value at 293 with 428
Replace the missing value at 295 with 48
Replace the missing value at 298 with 423
Replace the missing value at 316 with 88
Replace the missing value at 322 with 3
Replace the missing value at 412 with 315
Replace the missing value at 618 with 394
```

Figure 6: Output of cleaning code(Captured after running the code)

7 Appendix C

```
function x = breasrCancopy()

    function [a,b,c,d] = confusionMatrix(Actual,Predicted)
        a = 0;              % Actual is malignant and predicted is also ma
        b = 0;              % Actual is benign and predicted is malignant
        c = 0;              % Actual is malignant and predicted is benign
        d = 0;              % Actual is benign and predicted is also benig
        unclassified = 0;   % Predicted is neither benign nor maligant
        [rows, columns] = size(Actual);
        for i = 1:columns
            if Actual(1,i) == 4 && Predicted(1,i) == 4
                a = a + 1;    % 4 represents malignant, Actual is malignan
            elseif Actual(1,i) == 2 && Predicted(1,i) == 4
                b = b + 1;    % 2 represents benign, Actual is benign and
            elseif Actual(1,i) == 4 && Predicted(1,i) == 2
                c = c + 1;    % Actual is malignant and predicted is benig
            elseif Actual(1,i) == 2 && Predicted(1,i) == 2
                d = d + 1;    % Actual is benign and predicted is also ben
            else
                unclassified = unclassified + 1;
            end
        end

    end

    function result = binaryRound(result)
       [rows,cols] = size(result);
       for i=1:rows
           for j=1:cols
               if result(i,j)<3.000000
                   result(i,j)=2;
               else
                   result(i,j)=4;
               end
           end
       end
%         roundResult = result;
    end

data = load ('breastCancerCleanedData.txt');
numberOfMalignantRows = 241;
numberOfBenignRows = 458;
totalRows = numberOfMalignantRows + numberOfBenignRows;
```

```
trainingDataPartition = 0.80;                    % Training data percentage
trainingBenignRows = round(numberOfBenignRows*trainingDataPartition);
testingBenignRows = numberOfBenignRows - trainingBenignRows;
trainingMalignantRows = round(numberOfMalignantRows*trainingDataPartition)
testingMalignantRows = 241 - trainingMalignantRows;

trainingInput = data(1:trainingBenignRows,1:9);          % pick rows form b
trainingInput = [trainingInput;data(numberOfBenignRows+1:numberOfBenignRow
testingInput = data(trainingBenignRows+1:numberOfBenignRows,1:9);
testingInput = [testingInput; (data(numberOfBenignRows+2+trainingMalignantR
% pick rows form both classes
trainingTarget = data(1:trainingBenignRows,10);          % pick rows form b
trainingTarget = [trainingTarget;data(numberOfBenignRows+1:numberOfBenignR
testingTarget = data(trainingBenignRows+1:numberOfBenignRows,10);
testingTarget = [testingTarget; (data(460+trainingMalignantRows:totalRows,
% pick rows form both classes

trainingInput = trainingInput';
testingInput = testingInput';
trainingTarget = trainingTarget';
testingTarget = testingTarget';

numberOfHiddenLayer1Nueurons = 3;
numberOfHiddenLayer2Nueurons = 3;
net1=newff([0 10;0 10;0 10;0 10;0 10;0 10;0 10;0 10;0 10],[numberOfHiddenL
net1.trainParam.epochs=3000;
[net2,tr]=train(net1, trainingInput,trainingTarget);
%a=sim(net2,p);
a=binaryRound(sim(net2,testingInput));
[a,b,c,d] = confusionMatrix(testingTarget,a);
accuracy = ((a+d)/(a+b+c+d))*100
recall = ((a)/(a+b))*100
precision = ((a)/(a+c))*100
mse(a-testingTarget)
end
```