## Neural Networks and Fuzzy Systems

## Breast cancer classification using Neural Network

## Abstract

Classification is one of the interested areas in the field of Neural Networks. Artificial Neural Networks has been extensively used in pattern recognition, medical diagnosis and there are many other applications of Neural Networks. Breast cancer diagnosis is one of the interesting and important application in Artificial Neural Networks [1]. In this document numerous experiments, on different hypothesis, have been discoursed in order to diagnose breast cancer.

## Background

Breast cancer, is a type of cancer that affects women. The struggle to fight with breast cancer is an important area in medical. Neural Network can be the powerful tool in robust and reliable recognition of breast cancer [1]. In this document we brought different hypothesis into experiments to check their validity and also discussed the environmental setup to make up an efficient Neural Network to identify breast cancer.

## Environment Setup

Wisconsin data set is used in training and testing of Neural Networks. In this data set first column indicates id of each row, next nine columns indicate actual data and the last column is the output against each row. Therefore, we need to separate ids, actual data and output data from given data set before we train Neural Network. There are some unknown values in the given data set which are replaced by 5 in the following experiments.

Significant key parameters which are used to set architecture in training of Neural Networks are:

* Activation Functions
* Hidden Layers
* Learning Functions
* Mean Square Error
* Epochs

## 1st Hypothesis

Training of Neural Network on half (50%) of provided data, in which 32.5% of selected data indicates Benign and 17.5% indicates Malignant, would be much more effectual as compare to randomly chosen data from given data set.

Experiments which have been done over the above given hypothesis are explained in **Table 1**. The key parameters which remained constant are:   
MSE = 0.01  
Epochs = 100

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Expr | Hidden Layers | Activation  Functions | Output Layer Function | Learning Function | Training Function | Efficiency | | Efficiency of Randomly Chosen Data | |
| Accuracy % | Time (secs) | Accuracy % | Time (secs) |
| 1 | 20 | {1xtansig} | tansig | learngd | trainr | 97.7143 | 11 | 73.4286 | 03 |
| 2 | {2x20} | {2xtansig} | tansig | learngd | trainr | 98 | 10 | 70.8571 | 04 |
| 3 | {3x20} | {3xtansig} | tansig | learngd | trainr | 99.1429 | 08 | 74.2857 | 11 |
| 4 | {4x20} | {4xtansig} | tansig | learngd | trainr | 98.2857 | 08 | 74.5714 | 06 |
| 5 | {5x20} | {5xtansig} | tansig | learngd | trainr | 97.1429 | 12 | 74.5714 | 13 |
| 6 | {6x20} | {6xtansig} | tansig | learngd | trainr | 96.5714 | 12 | 75.7143 | 08 |
| 6 | {6x20} | {6xtansig} | softmax | learngd | trainr | 34.5714 | 11 | 0 | 13 |
| 7 | {6x20} | {6xtansig} | purelin | learngd | trainr | 98.5714 | 13 | 74 | 12 |
| 8 | 20 | {1xtansig} | purelin | learngd | trainr | 97.4286 | 7 | 72.8571 | 10 |
| 9 | {2x20} | {2xtansig} | purelin | learngd | trainr | 95.4286 | 6 | 74.8571 | 08 |
| 10 | {3x20} | {3xtansig} | purelin | learngd | trainr | 97.1429 | 11 | 74 | 09 |
| 11 | {3x20} | {2xlogsig} | purelin | learngd | trainr | 97.7143 | 19 | 69.4286 | 05 |
| 12 | {3x20} | {3xlogsig} | logsig | learngd | trainr | 34.5714 | 43 | 0 | 11 |
| 13 | 5 | {1xtansig} | tansig | learngd | trainr | 98 | 07 | 74 | 06 |
| 14 | 10 | {1xtansig} | tansig | learngd | trainr | 98.5714 | 05 | 09 | 76 |
| 15 | 15 | {1xtansig} | tansig | learngd | trainr | 98.5714 | 13 | 72.8571 | 07 |
| 16 | {2x5} | {2xtansig} | Tansig | learngd | trainr | 98.2857 | 10 | 74.8571 | 09 |
| 17 | {3x5} | {3xtansig} | tansig | learngd | trainr | 98.2857 | 10 | 75.7143 | 05 |
| 18 | {3x10} | {3xtansig} | tansig | learngd | trainr | 97.1429 | 09 | 74.8571 | 11 |
| 19 | {1x15, 2x10} | {3xtansig} | tansig | learngd | trainr | 97.7143 | 09 | 74.2857 | 06 |
| 20 | {3x20} | {3xtansig} | softmax | learngd | trainr | 34.5714 | 23 | 0 | 06 |
| 21 | {3x20} | {3xtansig} | purelin | learngd | trainr | 97.4286 | 15 | 68.8571 | 07 |
| 22 | {3x0} | {3xtansig} | tansig | learngdm | trainr | 96 | 08 | 75.4286 | 09 |
| 23 | {3x20} | {3xtansig} | purelin | learngdm | trainr | 97.4286 | 11 | 74 | 06 |
| 24 | {3x20} | {3xtansig} | tansig | learngdm | traingdx | 97.1429 | 00 | 73.4286 | 0 |
| 25 | {3x15} | {3xtansig} | tansig | learngd | trainr | 98 | 10 | 74.2857 | 09 |
| 26 | {1x20,  1x15,  1x10} | {3xtansig} | tansig | learngd | trainr | 98.5714 | 10 | 75.1429 | 08 |
| 27 | {1x20,  2x15} | {3xtansig} | tansig | learngd | trainr | 98 | 07 | 74.2857 | 10 |

Table 1

**Analysis: -** We can see a great difference in the performance of Neural Networks if we compare accuracy of experiments being done under given hypothesis and under randomly chosen data. Nearly all of the experiments gave more than 96% of correctness. The appropriate environmental setup for training of Neural Network, for above given hypothesis, has been conversed below.

1. **Hidden Layers: -** According to the accuracy rate of above experiments it has been observed that the number of hidden layers may vary between 1 to 3 and the size of hidden layers can be set between 5 to 20. More than 3 hidden layers would be of no use because we see that the performance decreases, a little bit or stays constant, as we set number of hidden layers more than 3.
2. **Activation Functions: -** **logsig** and **tansig** activation functions have been used for hidden layers in above experiments and **tansig**, **purelin** and **softmax** have been used for output layer. The finest activation function for hidden layer remained **tansig** while for output layer **purelin** and **tansig** functions gave a much better performance than **softmax**. Therefore, we could use **tansig** or **purelin** as activation functions for output layer.
3. **Learning and Training Functions: -** Recommended learning and training functions are **learngd** and **trainr**. **learngdm** and **traingdx** also gave similar performance to **learngd** and **trainr**.

## 2nd Hypothesis

Performance of Neural Network rises as we increase number of hidden layers but after a perimeter performance reduces and if we frequently increase hidden layers performance surges again. This rise and fall in the performance of neural network would persist if we stay increasing hidden layers.

To test this hypothesis training of Neural Network is done according to the first hypothesis. The key parameters which remained constant are:

MSE = 0.01  
Epochs = 100  
Training Function = learngd  
Training Function = trainr  
Output Layer Functions = tansig t, purelin p

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Expr | Hidden Layers | Activation  Functions | Efficiency | |
| Accuracy % | Time (secs) |
| 1 | 20 | {1xtansig} | t=96.8571  p=98.5714 | t=07  p=19 |
| 2 | {2x20} | {2xtansig} | t=98.5714  p=97.7143 | t=05  p=04 |
| 3 | {3x20} | {3xtansig} | t=97.4286  p=97.4268 | t=11  p=08 |
| 4 | {4x20} | {4xtansig} | t=98.5714  p=96.8571 | t=14  p=06 |
| 5 | {5x20} | {5xtansig} | t=98.2857  p=98 | t=11  p=10 |
| 6 | {6x20} | {6xtansig} | t=97.1429  p=97.1429 | t=08  p=24 |
| 7 | {7x20} | {7xtansig} | t=97.4286  p=96.8571 | t=12  p=12 |
| 8 | {8x20} | {8xtansig} | t=98  p=97.7143 | t=18  p=17 |
| 9 | {9x20} | {9xtansig} | t=98.2857  p=97.4286 | t=10  p=10 |
| 10 | {10x20} | {10xtansig} | t=97.1429  p=98.2857 | t=26  p=16 |
| 11 | {1x5} | {1xtansig} | t= 97.7143  p= 98.2857 | t=04  p=16 |
| 12 | {2x5} | {2xtansig} | t=94.5714  p=97.7143 | t=13  p=37 |
| 13 | {3x5} | {3xtansig} | t=97.4286  p=98.5714 | t=29  p=30 |
| 14 | {4x5} | {4xtansig} | t=98.2857  p=97.4286 | t=12  p=15 |
| 15 | {5x5} | {5xtansig} | t=97.1429  p=97.1429 | t=16  p=23 |
| 16 | {1x10,  4x5} | {5xtansig} | t=97.4286  p=98.2857 | t=19  p=33 |
| 17 | {1x20,  1x10  3x5} | {5xtansig} | t=96.5714  p=98.8571 | t=35  p=30 |

Table 2

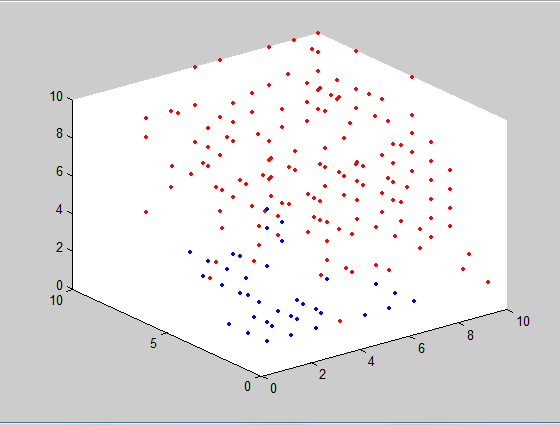
**Analysis: -** We have experienced from above experiments that accuracy remains more than 95% in, almost all experiments. Increasing number of hidden layers did not affect accuracy. Our Neural Network gave almost same accuracy with single hidden layer and with five neurons as with ten hidden layers with twenty neurons. This verifies that the hypothesis that we made does not match with actual results because there is no fluctuation in the accuracy rate against different experiments. So, the above hypothesis could be well stated as follows.

**Evolved Hypothesis: -** Performance of Neural Network is not affected with continual increase of hidden layers. Therefore, increasing number of hidden layers, after a maximum, would be of no use.

## 3rd Hypothesis

Performance of Neural Network increases if we train Neural Network on fewer-dimensional data instead of higher- dimensional data.

**Description: -** We know that every attribute (Symptoms of a Cell) is mapped to real number in the domain of 1-10. If an attribute is closer to 1 it indicates benign and if it is nearer to 10 it indicates malignant [2]. If we take a look at the given data we perceive that in the most of the cases if the values of first three columns are greater than or equal to 5 it is malignant and if the values of last three columns are less than 5 it is benign. For training purpose we separate the first three columns for malignant and last three columns for benign. In three dimension data complexity can be seen in the following figure. Red dots indicate Malignant and blue dots indicates Benign.



Data Complexity of Benign and Malignant

All of the environmental setup for training of Neural Network remains same as of above experiments; data dimension changes. . The key parameters which remained constant are:

MSE = 0.01  
Epochs = 100  
Training Function = learngd  
Training Function = trainr  
Output Layer Functions = tansig t, purelin p

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Expr | Hidden Layers | Activation  Functions | Efficiency | |
| Accuracy % | Time (secs) |
| 1 | {1x5} | {1xtansig} | t=98.5714  p=98.5714 | t=28  p=29 |
| 2 | {2x5} | {2xtansig} | t=98.5714  p=98.8571 | t=35  p=25 |
| 3 | {3x5} | {3xtansig} | t=99.1429  p=98.5714 | t=70  p=44 |
| 4 | {1x10,  2x5} | {3xtansig} | t=98  p=98 | t=26  p=50 |
| 5 | {2x10,  1x5} | {3xtansig} | t=98.5714  p=97.7143 | t=22  p=18 |
| 6 | {3x10} | {3xtansig} | t=99.4286  p=98.8571 | t=31  p=11 |
| 10 | {1x20,  2x10} | {3xtansig} | t=99.4286  p=99.4286 | t=14  p=31 |
| 11 | {2x20,  1x10} | {3xtansig} | t=98.2857  p=99.4286 | t=13  p=06 |
| 12 | {3x20} | {3xtansig} | t=98.8571  p=98.8571 | t=08  p=24 |

Table 3

**Analysis: -** It has been observed from above experiments that if we bring data down to fewer dimensions we get more good accuracy as compared to higher dimensional data because all the experiments done above gave more than 98% of accuracy which is a more good precision as compared to experiments done in the first hypothesis, on 9 dimensional data and some of the experiments gave almost 100% accuracy.

## 4th Hypothesis

With higher number of hidden layers Neural Network takes less time to be trained as compared to the time with less number of hidden layers.

The key parameters which remained constant in the experiments of the above given hypothesis are:

MSE = 0.01  
Epochs = 100  
Training Function = learngd  
Training Function = trainr  
Output Layer Functions = tansig t, purelin p

|  |  |  |  |
| --- | --- | --- | --- |
| Expr | Hidden Layers | Activation  Functions | Training Time  (secs) |
|
| 1 | {1x5} | {1xtansig} | t=21  p=19 |
| 2 | {2x5} | {2xtansig} | t=24  p=20 |
| 3 | {3x5} | {3xtansig} | t=12  p=11 |
| 4 | {4x5} | {4xtansig} | t=18  p=15 |
| 5 | {5x5} | {5xtansig} | t=24  p=12 |
| 6 | {1x10} | {1xtansig} | t=10  p=23 |
| 7 | {2x10} | {2xtansig} | t=19  p=9 |
| 8 | {3x10} | {3xtansig} | t=18  p=17 |
| 9 | {1x20,  2x10} | {3xtansig} | t=13  p=22 |
| 10 | {2x20,  1x10} | {3xtansig} | t=14  p=17 |
| 11 | {3x20} | {3xtansig} | t=18  p=21 |

Table 4

**Analysis: -** The above stated hypothesis is incorrect because sometimes Neural Network takes more time in training on higher number of hidden layers and sometimes it takes less time in training with lesser number of hidden layers. So, time is not affected by number of hidden layers and its size. The given hypothesis could be stated as:

**Evolved Hypothesis: -** Time taken by Neural Network during training is independent of number and size of hidden layers.

## Bibliography

(1). Tike Thein, H. T. and Mo Tun, K. M. (2015) ‘An Approach for Breast Cancer Diagnosis Classification Using Neural Network’, Advanced Computing: An International Journal, 6(1), pp. 1–11. doi: 10.5121/acij.2015.6101.

(2). Wolberg, W. H. and Mangasarian, O. L. (1990) ‘Multisurface method of pattern separation for medical diagnosis applied to breast cytology.’, Proceedings of the National Academy of Sciences of the United States of America, 87(23), pp. 9193–6. doi: 10.1073/pnas.87.23.9193.