Assignment 4 - Neural Network Analysis of Breast Cancer Diagnosis

Tanushree Kumar

DATA 630 - Summer 2024

Professor Ami Gates

University of Maryland Global Campus

Due: July 16, 2024

Introduction

The objective of this analysis is to develop a neural network model to classify breast cancer tumors as either malignant or benign based on diagnostic measurements. Specifically, the aim is to answer the following questions: can a neural network accurately classify the type of breast cancer based on diagnostic features, and which features are most influential in the classification process?

Breast cancer is a prevalent disease, particularly among women. According to Breast Cancer Statistics and Resources (n.d.), about 2.3 million women were diagnosed with breast cancer worldwide and 670,000 died in 2022 alone. Unfortunately, “breast cancer is the 2nd most common cancer worldwide” and the “number one cancer in women” (World Cancer Research Fund International, 2022). As a result, early and accurate diagnosis is critical for effective treatment and improved survival rates. The Breast Cancer Wisconsin (Diagnostic) dataset, provided by the University of Wisconsin Hospitals contains diagnostic features of “cell nuclei present in a digitized image of a fine needle aspirate (FNA) of a breast mass” (Wolberg et al., 1995). By building accurate classification models, preventative screenings and measures can be taken to identify tumors and improve the health of the individual.

Neural networks are powerful tools for classification tasks due to their ability to model complex, non-linear relationships between inputs and outputs. Given the multi-dimensional nature of the diagnostic features in the dataset, a neural network can capture intricate patterns that distinguish malignant tumors from benign ones. This makes neural networks an appropriate choice for the classification of this dataset.

Analysis

The Breast Cancer Wisconsin (Diagnostic) dataset consists of 569 instances with 30 numeric features representing various characteristics of the cell nuclei obtained from the FNA of a breast mass. Each instance is a record of a different breast cancer tumor. The target variable of this dataset is the diagnosis of the tumor and is a binary variable, indicating whether the tumor is malignant or benign.

The dataset used in this analysis is the Breast Cancer Wisconsin (Diagnostic) dataset available from the UCI Machine Learning Repository. The dataset contains an 'ID' variable to identify each tumor alongside the 'diagnosis' variable which contains the result of the tumor diagnosis as either 'M' for malignant or 'B' for benign. The difference between these two types of tumors should be noted and understood. Malignant tumors are cancerous while benign tumors are non-cancerous. When imaged, the difference can be observed as a "benign tumor has distinct, smooth, regular borders" whereas a "malignant tumor has irregular borders and grows faster than a benign tumor" (Cleveland Clinic, 2021).

The numeric features were computed from a digitized image of the FNA of a breast mass. These 30 predictive attributes were compiled from ten features. The "mean value, largest (or 'worst') value, and standard error of each feature were found over the range of isolated cells" (Street et al., 1993). These ten features computed for each cell nucleus are:

- radius: mean of distances from the center to points on the perimeter
- texture: standard deviation of gray-scale values
- perimeter
- area
- smoothness: local variation in radius lengths

- compactness: calculated using perimeter^2 / area - 1.0
- concavity: severity of concave portions of the contour
- concave points: number of concave portions of the contour
- symmetry
- fractal dimension: calculated as "coastline approximation" - 1; (Wolberg et al., 1995).

For reference, the ‘radius’ variable is the mean value; ‘radius2’ is the largest value; and ‘radius3’ is the the standard error. Thus, the variables representing the mean values are ‘radius’, ‘texture’, ‘perimeter’, ‘area’, ‘smoothness’, ‘compactness’, ‘concavity’, ‘concave’, ‘symmetry’, and ‘fractal’. The variables representing the standard error are ‘radius2’, ‘texture2’, ‘perimeter2’, ‘area2’, ‘smoothness2’, ‘compactness2’, ‘concavity2’, ‘concave2’, ‘symmetry2’, and ‘fractal2’. The variables representing the mean of the worst (largest) values are ‘radius3’, ‘texture3’, ‘perimeter3’, ‘area3’, ‘smoothness3’, ‘compactness3’, ‘concavity3’, ‘concave3’, ‘symmetry3’, and ‘fractal3’.

Before building the model, exploratory data analysis (EDA) was performed to understand the distribution and relationships of the variables. Based on the EDA, data preprocessing was performed such as handling missing values, removing unnecessary columns, encoding the target variable, and scaling all the independent variables which made all the input variables have the same mean and the same standard deviation. Scaling the features also ensured that no single feature dominates due to its magnitude, which helps improve the performance of the model. Using functions such as summary(), str(), and sum(is.na()) gave an overview of the dataset and showed that no missing values were present. The dataset description website also stated no missing values for any columns. Although the target variable ‘diagnosis’ was already a factor variable with two levels, the values were not numerical. A command was run to encode and map

the ‘M’ and ‘B’ values to ‘1’ and ‘0’ respectively as machine learning algorithms require numeric input. The summary command was run again after the conversion to validate that the changes had been made correctly. All the variables were numerical, meaning little data preprocessing had to be performed.

The summary function provided a statistical summary for each variable in the dataset. The ‘diagnosis’ variable had a higher proportion of benign cases with the distribution showing 357 benign cases (B) and 212 malignant cases (M). The radius variable ranges from 6.981 to 28.110, with a median value of 13.370. This indicates that the majority of the radius values are approximately between the first quartile of 11.700 and the third quartile of 15.780. The ‘texture’ variable ranges from 9.71 to 39.28, with a median value of 18.84. This indicates that the texture measurements are generally higher compared to the radius values. The ‘perimeter’ variable ranges from 43.79 to 188.50, with a median value of 86.24. This suggests that perimeter measurements show greater variability compared to radius and texture. Since the perimeter is the mean size of the core tumor, the wide range indicates significant variability in tumor size among patients. The ‘area’ variable ranges from 143.5 to 2501.0, with a median value of 551.1. This indicates a significant variation in the area measurements. The ‘smoothness’ variable ranges from 0.05263 to 0.16340, with a median value of 0.09587. The values are relatively close together, indicating less variability compared to other features like area or perimeter.

Other variables, such as compactness, concavity, concave, symmetry, and fractal, along with their respective standard errors and worst values, follow a similar pattern of distribution, indicating the characteristics of cell nuclei in breast tissue samples.

A neural network consists of layers of interconnected nodes (neurons) where each connection has an associated weight. The network learns by adjusting these weights based on the

error in its predictions. The learning process involves forward propagation which calculates the output based on the current weights; and backward propagation which adjusts the weights to minimize the error using optimization techniques like gradient descent.

There are three key parameters. The first is the number of hidden layers and neurons which determines the complexity of the model. The activation functions that introduce non-linearity such as ReLU, sigmoid, etc. (Sharma, 2017). The learning rate controls the step size of the weight updates.

The key steps to fit the neural network model include splitting the data into training and testing sets; defining the model to specify the architecture (number of layers, neurons, activation functions); training the model to training data and using techniques like cross-validation to tune hyperparameters; and evaluating the model to assess the model's performance on the test data using metrics like accuracy, precision, recall, and F1-score.

To fit the model, the data was split into a training set of 70% and a testing set of 30%. The training set was used to build the model while the test set was used to evaluate the accuracy of the model. A seed value was set which ensured the results would be reproducible when the model was run again. The formula was built using the ‘neuralnet’ function from the ‘neuralnet’ library. The dependent variable was the ‘diagnosis’ with all of the other 30 variables being the independent variables. By default, the number of hidden layers was set to ‘2’, meaning the network will have one hidden layer with two nodes. The linear output was also set to ‘FALSE’ for classification. Different output properties were also studied such as the response, covariate, model list, net result, weights, start weights, and the result matrix. The network was graphed

As a final step, confusion matrices will be made to check the classification accuracy for the training data and to evaluate the model based on the test data.

Results

After the model was built and run, the different neural network properties were observed. Using the 'nn$call' command showed the formula that was used to build the network. An image of this can be seen below and in Figure 1 in Appendix A.

```
> nn$call
neuralnet(formula = diagnosis ~ radius + texture + perimeter +
    area + smoothness + compactness + concavity + concave + symmetry +
    fractal + radius2 + texture2 + perimeter2 + area2 + smoothness2 +
    compactness2 + concavity2 + concave2 + symmetry2 + fractal2 +
    radius3 + texture3 + perimeter3 + area3 + smoothness3 + compactness3 +
    concavity3 + concave3 + symmetry3 + fractal3, data = train_data,
    hidden = 2, err.fct = "ce", linear.output = FALSE)
```

Figure 1. *Figure of the formula used to build the neural network.*

Other properties such as the covariate lists the standardized values of the input variables used in the model. These standardized values typically have a mean of 0 and a standard deviation of 1, which helps in training the neural network effectively. Figure 2 in Appendix A shows the covariate.

The net result shows the first 10 predicted probabilities of the neural network for the training data. The figure below shows the values that are all very close to 1, indicating high confidence in the predictions. This figure can also be seen in Figure 3 in Appendix A.

```
> # First 10 predicted probabilities
> nn$net.result[[1]][1:10]
 [1] 0.9999999 0.9999999 0.9999999 0.9999999 0.9999999 0.9999999 0.9944522 0.9999999 0.9999999 0.9999999
```

Figure 3. *Figure of the first 10 predicted probabilities of the neural network for the training data.*

The weights of the neural network after the last training iteration determine the strength of the connections between neurons in the network. This can be seen in Figure 4 in Appendix A.

The start weights show the initial weights of the neural network before any training iterations. These weights are adjusted during the training process to minimize the error. This can be seen in Figure 5 in Appendix A.

The results matrix displays various metrics and weights associated with the neural network after training, including the error, steps taken for convergence, and specific weights for each input variable to hidden layer connections and hidden layer to output connections. A snippet of the matrix can be seen below and the full matrix in Figure 6 in Appendix A.

```
> # Number of trainings steps, the error, and the weights
> nn$result.matrix
                                  [,1]
error                     6.510747284
reached.threshold         0.009896577
steps                   102.000000000
Intercept.to.1layhid1    -0.809570720
radius.to.1layhid1        0.246912783
texture.to.1layhid1       0.948759481
perimeter.to.1layhid1    -1.704345979
```

Figure 6(b). *A snippet of the result matrix of the neural network.*

The neural network was then plotted to visualize the network architecture and to understand the structure and connectivity. The figure of the entire neural network can be seen below and in Figure 9 in Appendix A.

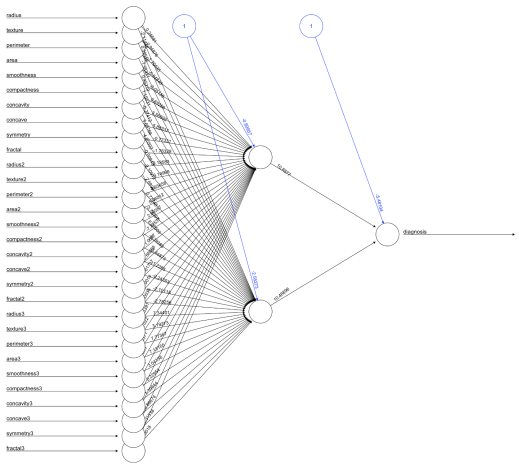

Figure 9. *Figure of the generated neural network.*

The numerical outputs of the neural network model provide insights into its performance, the learned weights, and the significance of various features in predicting breast cancer diagnosis. As seen in Figure 6(b), the final error value is 6.5107. This metric represents the sum of squared errors (SSE) between the predicted and actual values in the training dataset. A lower

SSE indicates a better fit of the model to the data, suggesting that the model effectively minimizes prediction errors. The reached threshold value of 0.0099 indicates the threshold at which the model stopped training. This typically corresponds to a point where the change in error between iterations falls below a predefined value, indicating convergence. The model completed 102 training steps (iterations) to reach the convergence threshold which shows the efficiency of the training process in optimizing the weights.

The weights are critical components in neural networks as they represent the influence of input features on the hidden layers and the final output. Interpreting the weights from input features to the first hidden layer, '1layhid1', it can be seen that the majority of the weights are positive. These significant positive weights, e.g., 'concave.to.1layhid1: 5.8834', indicate that features like concave points of the cell nucleus have a strong positive influence on activating the first hidden neuron, suggesting a correlation with malignant tumors. Other features with positive values are 'radius', 'texture', and 'compactness' but their values are less than 1 which indicates a weaker influence. Another strong positive feature is 'concavity' with a value of 3.5669. Conversely, significant negative weights, e.g., 'perimeter.to.1layhid1: -1.7043, and 'fractal.to.1layhid1: -1.7633', suggest features negatively influencing activation, potentially correlating with benign tumors. Other features such as the 'area', 'smoothness', and 'symmetry' are less than 0, suggesting a weak influence as well.

Similar to the first hidden layer, the large positive weights, e.g., 'concavity.to.1layhid2: 5.6977' and 'concave.to.1layhid2: 4.6030', highlight the importance of features like concavity in predicting malignancy. Negative weights indicate features contributing to benign classifications. A strong negative weight was 'radius.to.1layhid2: -2.1145', indicating a tumor might be benign since benign tumors have smooth shapes.

The weights from the hidden layers should be interpreted. The substantial positive weight values of ‘1layhid1.to.diagnosis: 10.8877’ and ‘1layhid2.to.diagnosis: 10.4690’ from the hidden layers to the diagnosis output indicate a strong influence of the hidden layer activations on the final prediction, showing that the learned representations in the hidden layers are crucial for accurate classification.

The significant weights reveal that features such as ‘concavity’, ‘concave points’, and ‘perimeter’ are most influential in the diagnosis as they play critical roles in the model's decision-making process, aligning with the medical understanding that these metrics often correlate with malignancy. The low error value and high precision in weights demonstrate the model's robustness in distinguishing between malignant and benign tumors. This robustness is essential for clinical applications, ensuring reliable and consistent diagnoses. The model's ability to accurately classify tumors based on cell nucleus characteristics suggests its potential as a powerful diagnostic tool in clinical settings. By highlighting important features, the model can aid pathologists in focusing on critical aspects of cell morphology during assessments.

Using the table command, a confusion matrix was built to check the classification accuracy of the training data. The number of correctly classified instances of benign tumors was 240 and the number of correctly classified instances of malignant tumors was 134, making the total number of correct instances 374. The total number of instances in the training set was 376. The classification accuracy of the training model is (374/376)*100 = 99.47%. Figure 7 in Appendix A shows the confusion matrix for the training data.

The same can be done for the test data. The number of correctly classified instances of benign tumors was 117 and the number of correctly classified instances of malignant tumors was 71, making the total number of correct instances 188. The total number of instances in the test set

was 194. The classification accuracy for the test set is (188/194)*100 = 96.91%. Figure 8 in Appendix A shows the confusion matrix for the test data.

The training data had an accuracy of 99.47% which means the model correctly classified 99.47% of the instances in the training set. Such a high level of accuracy shows that the model has learned to distinguish between benign and malignant cases well and has successfully captured the patterns in the training data. While high accuracy on the training set is desirable, an accuracy this high can be an indication that the model might be overfitting. Overfitting occurs when a model learns not only the underlying patterns but also the noise and specifics of the training data, which can reduce its ability to generalize to new data.

The test data had an accuracy of 96.91% which means the model correctly classified 96.91% of the instances in the test set. The model is performing well on unseen data, indicating it has generalized well from the training data to the test data; and suggests that the model is effective at identifying whether a tumor is benign or malignant in new, unseen cases. The slight drop in accuracy between the training set to the test set is expected as this small difference indicates that the model has not significantly overfitted the training data and is performing well in a real-world context. In the context of diagnosing breast cancer, these results mean that the model can accurately classify tumors based on the features provided. With 96.91% accuracy on the test data, the model has a high probability of correctly identifying whether a tumor is benign or malignant, which can be very valuable in clinical settings for early and accurate diagnosis.

Conclusion

The primary goal of this analysis was to determine if breast cancer tumors could be accurately classified as malignant or benign based on specific diagnostic features. The findings suggest that by using advanced analytical techniques, it is possible to distinguish between the

two types of tumors with high accuracy. The features that stood out as most influential in this classification process were the shapes and textures of the cell nuclei, which are critical in identifying the nature of the tumor.

While the analysis showed promising results, there are a few limitations to consider. The dataset used, although comprehensive, may not fully represent the diversity of breast cancer cases encountered in clinical practice. The features were derived from images of cell nuclei, which means that any variation in image quality or processing could affect the results. The dataset also had an uneven ratio of benign to malignant cases as there were 357 benign cases but only 212 malignant cases which might have caused a skew in the model's results. Additionally, the model's performance was highly accurate on the dataset used, but this does not guarantee similar performance on new or different datasets.

Future improvements could involve using a more diverse dataset that includes a wider variety of cases and imaging conditions. Additionally, incorporating other types of diagnostic information, such as genetic data or patient history, could enhance the accuracy and robustness of the model. Another potential improvement could be the continuous updating and retraining of the model with new data to ensure it remains current and effective.

This analysis demonstrates the potential for advanced data analysis techniques to aid in the early and accurate diagnosis of breast cancer. By identifying the most influential diagnostic features, healthcare professionals can focus on key indicators when assessing tumors. This approach could significantly benefit patients by enabling quicker, more accurate diagnoses, leading to timely and effective life-saving treatments. Ultimately, these advancements can improve patient outcomes and contribute to better breast cancer research, treatment, and care.

References

Breast Cancer Statistics And Resources. (n.d.). Breast Cancer Research Foundation. https://www.bcrf.org/breast-cancer-statistics-and-resources/#:~:text=Breast%20cancer%20is%20the%20most

Cleveland Clinic. (2021, August 10). *Benign Tumor: Definition, Types, Causes & Management*. Cleveland Clinic. https://my.clevelandclinic.org/health/diseases/22121-benign-tumor

Sharma, S. (2017, September 6). *Activation Functions in Neural Networks*. Medium; Towards Data Science. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Street, W. N., Wolberg, W. H., & Mangasarian, O. L. (1993). Nuclear feature extraction for breast tumor diagnosis. *Biomedical Image Processing and Biomedical Visualization*. https://doi.org/10.1117/12.148698

Wolberg, W., Mangasarian, O., & Street, W. N. (1995, October 31). *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

World Cancer Research Fund International. (2022, March 23). *Breast cancer statistics | World Cancer Research Fund International*. WCRF International. https://www.wcrf.org/cancer-trends/breast-cancer-statistics/

Appendix A

All figures and visualizations mentioned in the report can be seen below. The R code is attached as a separate file.

Figure 1

Figure of the formula used to build the neural network.

```
> nn$call
neuralnet(formula = diagnosis ~ radius + texture + perimeter +
    area + smoothness + compactness + concavity + concave + symmetry +
    fractal + radius2 + texture2 + perimeter2 + area2 + smoothness2 +
    compactness2 + concavity2 + concave2 + symmetry2 + fractal2 +
    radius3 + texture3 + perimeter3 + area3 + smoothness3 + compactness3 +
    concavity3 + concave3 + symmetry3 + fractal3, data = train_data,
    hidden = 2, err.fct = "ce", linear.output = FALSE)
```

Figure 2

Figure of the standardized values of the input variables used in the model.

```
> # Values of the input variables that were used to build the model
> nn$covariate [1:12,]
       radius    texture   perimeter        area  smoothness compactness  concavity     concave     symmetry    fractal      radius2    texture2   perimeter2        area2
5   1.7487579 -1.1508038  1.77501133  1.82462380  0.28012535  0.53886631  1.3698061  1.42723695 -0.009552062 -0.5619555  1.269425821 -0.78954898  1.272070124  1.189310289
6  -0.4759559 -0.8346009 -0.38680772 -0.50520593  2.23545452  1.24324156  0.8655400  0.82393067  1.004517928  1.8883435 -0.254846056 -0.59214063 -0.321021720 -0.289003924
7   1.1698783  0.1605082  1.13712450  1.09433201 -0.12302797  0.08821762  0.2998086  0.64636637 -0.064268069 -0.7616620  0.149751305 -0.80423225  0.155273668  0.298364935
8  -0.1184126  0.3581350 -0.07280278 -0.21877241  1.60263890  1.13910006  0.0609721  0.28170239  1.402120910  1.6588935  0.643057179  0.29030552  0.489620171  0.233516951
11  0.5370834  0.9184652  0.44162208  0.40609593 -1.01679116 -0.71291456 -0.7000684 -0.40432978 -1.034565253 -0.8253981 -0.092574387 -0.05411677 -0.197867461  0.003801211
12  0.4689800 -0.3254213  0.47866067  0.35835701  0.05259614  0.47070096  0.1347304  0.44174220  0.110823153 -0.2801003  0.362868097 -0.42047331  0.345198309  0.303860527
14  0.4888435  1.0835417  0.48277607  0.36318774 -0.87814055 -0.07840879  0.1327234  0.12166258  0.129061821 -1.3338705 -0.006750704 -0.25170639  0.018270737 -0.082589493
15 -0.1127373  0.7719888  0.06712078 -0.21763577  1.19024220  2.36607632  1.5554564  0.80743704  0.938858720  1.9860722 -0.696224995 -0.08674625 -0.398178605 -0.464423216
16  0.1171120  1.9182243  0.19593277  0.01111321  1.24712450  1.04442597  0.9420580  0.63708870  1.792428424  1.1291757 -0.126831739 -0.33328009  0.006400447 -0.171178434
17  0.1568390  0.1953835  0.11403633  0.08414239  0.16422766 -0.61237067 -0.1862688  0.09460271 -0.822996694 -0.5067176  0.243508270  0.04195892  0.162692600  0.111294988
18  0.5682975  0.3232597  0.66385362  0.40893753  1.46754343  1.85294273  1.0461727  1.38858000  1.285393429  1.5243395  0.591490849 -0.26077014  0.488630981  0.304300174
19  1.6125509  0.6650378  1.56512598  1.71948452  0.13863062 -0.03107174  0.7413551  1.18704838 -0.837587629 -1.2531381  1.273031858 -0.36228407  1.483262368  1.584113609
    smoothness2 compactness2    concavity2    concave2   symmetry2   fractal2    radius3   texture3  perimeter3        area3 smoothness3 compactness3  concavity3    concave3
5   1.481763364  -0.04847723  0.8277424542  1.14319885 -0.36077483  0.4988892  1.2974336 -1.4654809  1.33736272  1.219651081   0.2203623  -0.31311900  0.61263970  0.72861815
6   0.156209255   0.44515196  0.1598845164 -0.06906279  0.13400090  0.4864178 -0.1653528 -0.3135604 -0.11490835 -0.244105421   2.0467119   1.72010293  1.26213265  0.90509140
7  -0.908230679  -0.65099520 -0.3098687356 -0.22788851 -0.82893670 -0.6106805  1.3677798  0.3225990  1.36712237  1.274098467   0.5181843   0.02119633  0.50910429  1.19566374
8   0.587513921   0.26869628 -0.2323495106  0.43496578 -0.68739939  0.6111311  0.1636190  0.4006953  0.09936115  0.028834057   1.4466882   0.72414833 -0.02103534  0.62364699
11 -1.003151011  -0.90512489 -0.6918331217 -0.68151422 -0.71885213 -0.2845365  0.6043170  1.3345970  0.49218857  0.473194982  -0.6249267  -0.63027366 -0.60533934 -0.22601086
12 -0.422971297   0.84496927 -0.1319720527  0.16593446 -0.05592523  0.1319300  0.8588046  0.2607728  0.87013617  0.734893708   0.3167164   1.94891190  0.59586313  1.01006256
14  0.908577781   0.32286147  0.6167178975  1.31661061  1.12113292 -0.2996533  0.1181009  0.3225990  0.14102467 -0.007171473  -0.8439134  -0.39320214 -0.19167703 -0.04117035
15 -0.203821899   1.89197688  0.7657933300  0.72668650 -0.11278210  1.6243314 -0.2563890  1.0303468  0.04579378 -0.321209945   1.4335490   3.29380018  2.02330963  1.61554837
16 -0.477592120   0.94492359  0.5140214884 -0.14523431 -0.23859305  0.6315387  0.2463792  1.8633740  0.50111647  0.109978225   1.5518018   2.56415380  2.06309407  0.86097309
17 -0.440623149  -0.77384410 -0.3946760928 -0.11444156 -0.77933816 -0.6462048  0.5794890  0.8464951  0.48028471  0.452118574   0.6145385  -0.42688790  0.09208668  0.70427701
18 -0.004988783  -0.02614106 -0.0004543933  0.19024452 -0.44182611  0.1311742  0.9705309  0.9441155  0.87906406  0.762995585   2.0379525   1.07435317  0.98843488  1.41017002
19 -0.182173403  -0.36565073  0.0667951907  0.55327474 -0.84466307 -0.6794617  2.2864179  0.8464951  2.36704672  2.665141363   0.8247658   0.38601952  1.27028127  1.88938618
    symmetry3    fractal3
5  -0.8675896 -0.39675052
6   1.7525273  2.23983079
7   0.2622449 -0.01471753
8   0.4772206  1.72491676
11  0.0763637  0.03179084
12  1.4405702  1.15463563
14 -0.1483101 -1.16690689
15  1.1237639  3.27519557
16  2.1291389  2.77689167
17  0.2072887 -0.09887552
18  1.3015634  1.67508637
19 -0.2145808 -0.43163179
```

Figure 3

Figure of the first 10 predicted probabilities of the neural network for the training data.

```
> # First 10 predicted probabilities
> nn$net.result[[1]][1:10]
 [1] 0.9999999 0.9999999 0.9999999 0.9999999 0.9999999 0.9999999 0.9944522 0.9999999 0.9999999 0.9999999
```

Figure 4

Figure of the weights of the neural network after the last training iteration.

```
> # Network weights after the last method iteration
> nn$weights
[[1]]
[[1]][[1]]
            [,1]        [,2]
 [1,] -0.80957072 -2.55273198
 [2,]  0.24691278 -2.11445796
 [3,]  0.94875948  0.59559340
 [4,] -1.70434598 -1.78330573
 [5,] -0.84848236  0.45219231
 [6,] -0.05134887  1.19331412
 [7,]  0.69268276 -0.77411928
 [8,]  3.56692228  5.69765704
 [9,]  5.88337364  4.60301690
[10,] -0.77331231 -0.00942831
[11,] -1.76328279 -0.10911758
[12,]  0.18582642  2.68959741
[13,] -3.14968747  0.77748109
[14,]  4.93658337  3.75396951
[15,]  8.96012146  6.37609413
[16,]  2.00356543 -0.39382586
[17,] -7.22536776 -5.44873292
[18,]  1.06869559 -1.22850304
[19,]  2.66868252 -0.24353416
[20,]  0.61736742 -2.10117595
[21,] -3.90140236 -2.78256024
[22,]  1.20035545  7.34400829
[23,]  4.25391063  3.19273446
[24,]  7.07022420  7.77367108
[25,]  6.43371253  7.13154740
[26,]  1.54658531  3.04137773
[27,]  1.25735860  0.22954274
[28,]  1.34422928  1.39644458
[29,]  3.69824853  2.86673777
[30,]  0.75365311  0.31537760
[31,]  3.90439721  6.90180199

[[1]][[2]]
          [,1]
[1,] -5.481044
[2,] 10.887705
[3,] 10.468960
```

Figure 5

Figure of the initial weights of the neural network before any training iterations.

```
> # Network weights on the first method iteration
> nn$startweights
[[1]]
[[1]][[1]]
             [,1]        [,2]
 [1,] -0.76694901 -0.20115149
 [2,]  0.32208235 -1.50222129
 [3,] -0.70343114 -0.16926555
 [4,] -0.08094330 -1.21076410
 [5,] -0.72599690 -0.77113133
 [6,]  0.05420224  0.02326499
 [7,]  0.76701532 -0.47906283
 [8,] -0.42141085  0.01543135
 [9,]  0.51899780  1.59300911
[10,]  0.58885757  0.05287188
[11,] -1.63855765 -0.44530161
[12,] -1.20469204  0.29955121
[13,]  0.76292136  1.70847068
[14,]  0.46827920 -0.44078417
[15,]  1.94716722 -1.05798615
[16,]  0.77770453 -1.36012828
[17,] -2.31134268  1.84734420
[18,]  0.61601961 -0.20057548
[19,]  0.91726900 -0.64215120
[20,]  1.06239570  0.20841714
[21,] -0.07431360 -1.95642496
[22,] -0.97878715  2.07155604
[23,] -0.83932764 -0.85208214
[24,] -1.70337580 -0.44038520
[25,]  0.26328117 -1.45573288
[26,] -0.78967820 -1.14518339
[27,]  1.11218770  1.05066752
[28,] -0.74893752  0.44232184
[29,]  1.91248327  1.19816852
[30,]  0.11983564  0.52524216
[31,] -0.38931222  1.34153585

[[1]][[2]]
           [,1]
[1,] -1.5912018
[2,]  1.1193047
[3,]  0.7005598
```

Figure 6

Figure of the result matrix of the neural network.

```
> # Number of trainings steps, the error, and the weights
> nn$result.matrix
                                    [,1]
error                       6.510747284
reached.threshold           0.009896577
steps                     102.000000000
Intercept.to.1layhid1      -0.809570720
radius.to.1layhid1          0.246912783
texture.to.1layhid1         0.948759481
perimeter.to.1layhid1      -1.704345979
area.to.1layhid1           -0.848482360
smoothness.to.1layhid1     -0.051348875
compactness.to.1layhid1     0.692682758
concavity.to.1layhid1       3.566922281
concave.to.1layhid1         5.883373645
symmetry.to.1layhid1       -0.773312307
fractal.to.1layhid1        -1.763282785
radius2.to.1layhid1         0.185826420
texture2.to.1layhid1       -3.149687472
perimeter2.to.1layhid1      4.936583368
area2.to.1layhid1           8.960121462
smoothness2.to.1layhid1     2.003565434
compactness2.to.1layhid1   -7.225367764
concavity2.to.1layhid1      1.068695593
concave2.to.1layhid1        2.668682516
symmetry2.to.1layhid1       0.617367420
fractal2.to.1layhid1       -3.901402360
radius3.to.1layhid1         1.200355454
texture3.to.1layhid1        4.253910625
perimeter3.to.1layhid1      7.070224200
area3.to.1layhid1           6.433712530
smoothness3.to.1layhid1     1.546585310
compactness3.to.1layhid1    1.257358598
concavity3.to.1layhid1      1.344229283
concave3.to.1layhid1        3.698248531
symmetry3.to.1layhid1       0.753653110
fractal3.to.1layhid1        3.904397215
Intercept.to.1layhid2      -2.552731978
radius.to.1layhid2         -2.114457963
texture.to.1layhid2         0.595593396
perimeter.to.1layhid2      -1.783305727
area.to.1layhid2            0.452192311
smoothness.to.1layhid2      1.193314122
compactness.to.1layhid2    -0.774119281
concavity.to.1layhid2       5.697657043
concave.to.1layhid2         4.603016897
symmetry.to.1layhid2       -0.009428310
fractal.to.1layhid2        -0.109117577
radius2.to.1layhid2         2.689597410
texture2.to.1layhid2        0.777481094
perimeter2.to.1layhid2      3.753969505
area2.to.1layhid2           6.376094126
smoothness2.to.1layhid2    -0.393825857
compactness2.to.1layhid2   -5.448732922
concavity2.to.1layhid2     -1.228503042
concave2.to.1layhid2       -0.243534161
symmetry2.to.1layhid2      -2.101175948
fractal2.to.1layhid2       -2.782560243
radius3.to.1layhid2         7.344008290
texture3.to.1layhid2        3.192734462
perimeter3.to.1layhid2      7.773671081
area3.to.1layhid2           7.131547397
smoothness3.to.1layhid2     3.041377733
compactness3.to.1layhid2    0.229542738
concavity3.to.1layhid2      1.396444580
concave3.to.1layhid2        2.866737775
symmetry3.to.1layhid2       0.315377599
fractal3.to.1layhid2        6.901801994
Intercept.to.diagnosis     -5.481044027
1layhid1.to.diagnosis      10.887704729
1layhid2.to.diagnosis      10.468959843
```

Figure 7

Figure of the confusion matrix for the training data.

```
> # Confusion matrix for the training set
> table(mypredict, train_data$diagnosis, dnn = c("Predicted", "Actual"))
         Actual
Predicted   0   1
        0 240   1
        1   0 134
> mean(mypredict == train_data$diagnosis)
[1] 0.9973333
```

Figure 8

Figure of the confusion matrix for the test data.

```
> # Confusion matrix for the test set
> testPred <- compute(nn, test_data[, -1])$net.result  # Excluding the diagnosis column
> testPred <- apply(testPred, c(1), round)
> table(testPred, test_data$diagnosis, dnn = c("Predicted", "Actual"))
         Actual
Predicted   0   1
        0 117   6
        1   0  71
> mean(testPred == test_data$diagnosis)
[1] 0.9690722
```

Figure 9

Figure of the generated neural network.