COS 314 Assignment 3

Priyul Mahabeer - u20421169

04 June 2023

Abstract

This is a comprehensive report comparing the performance of an Artificial Neural Network to a Decision Tree evolved using the Genetic Programming algorithm. The classification was based on the *breast-cancer data set*.

1 Data Pre-Processing

There were a total number of 286 instances in the data set. Each instance contained a total of 10 attributes, with the first attribute being the classifier, and the remaining nine attributes - some of which are linear and some nominal.

1.1 Encoding

The data was encoded using one-hot encoding. For each unique category in the categorical variable, a separate binary column was created. The number of binary columns is equal to the number of unique categories. In total, there were 51 unique categories excluding *class*. Hence, one instance of the data which was one-hot encoded contained a vector of 51 bits, with a value of 1 assigned to the corresponding binary column if the observation belongs to that category; a value of 0 otherwise.

1.2 Data Randomization

The order of instances in the data set were randomly shuffled before the encoding and processing. Since the original data set was listed in order, all the lines were shuffled randomly. The process is as follows:

1. Read all lines from the data set into a vector.
2. Shuffle the indices of the vectors randomly.
3. Write the new shuffled vector back to the data set.

1.3 Incomplete Data

There were a total of 9 instances which were incomplete. Since one-hot encoding was utilized for representing the data, initially all 51 bits are set to 0. If a value at a corresponding index of the instance is true, that value in the encoded data is set to 1. Missing data was represented by a *?* in the data set. During processing this, if an instance with a missing attribute was found, a random index in the range of that attribute was set to 1.

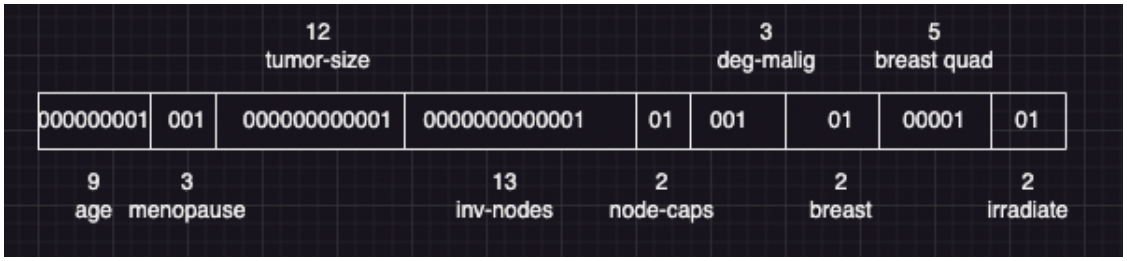
The main motivation for choosing this approach over removing incomplete data was because the data set instances were already small - and removing incomplete instances would make the total data set even smaller, which is detrimental for training.

1.4 Parsing Training and Testing Data

The first 80% of the data set was used for training. Each instance of the data set was separated into *tokens*, with each token identified by the separated comma in an instance. For example, consider a randomly selected instance:

- *recurrence-events,30-39,premeno,15-19,0-2,no,1,right,left-low,no*

The data was split into tokens, where Tokens[1] represented *age* (there were a total of 9 possible age values, so 9 bits in the encoded instance were used), Tokens[9] to Tokens [11] represented *menopause*, and so on. Tokens[0] represented the class which needs to be classified by the network (recurrence-events or no-reccurence-events)



Thus the attribute for *age* in the example instance (30-39) means that instance[2]=1, while instance[0] to instance[8] (excluding instance[2]) were set to 0. This was done for all instances in the training data set.

Similarly, the same was done for the testing data set, but the other 20% of data (which weren't considered in the training data set) was used to assemble the testing data set.

2 Artificial Neural Network

2.1 Initialization

Since the one-hot encoding for an instance was 51 bits long, there were a total of 51 input nodes. Since this is a binary classification problem, naturally there is only 1 output node. The number of hidden layers as well as number of nodes in each hidden layer will be determined during fine-tuning and training.

Weights and biases were initialised randomly generating a number between 0 and 1, and dividing it by the total number of connections that particular node had. Thus one node had a vector of weights and biases, each corresponding to a respective node in the next layer. The random initialization was done using a seed value of *1234*.

2.2 Training

The neural network is initialized with a vector of layers (representing all the layers in the network; with each layer having a number of nodes, input layer = 51 nodes, output layer = 1 node, hidden layer = random(0,51) nodes), an initial learning rate, number of epochs, and a vector of the training data.

2.2.1 Activation function

The ReLU function was used in the hidden layer nodes.

$$ReLU(x) = max(0, x) \tag{1}$$

The activation function used n the output layer was the sigmoid function:

$$f(x) = \frac{1}{1 + e^x} \tag{2}$$

Sigmoid was used for the output layer activation function because the network is performing binary classification, allowing us to interpret the output as a probability of the class. Furthermore, the sigmoid function is differentiable, which means it can be used in conjunction with backpropagation for gradient-based optimization - which is used in this network for optimizing weights and biases.

2.2.2 Learning rate

The initial learning rate selected was to be 0.01. During training, different learning rates were evaluated and the results are presented below.

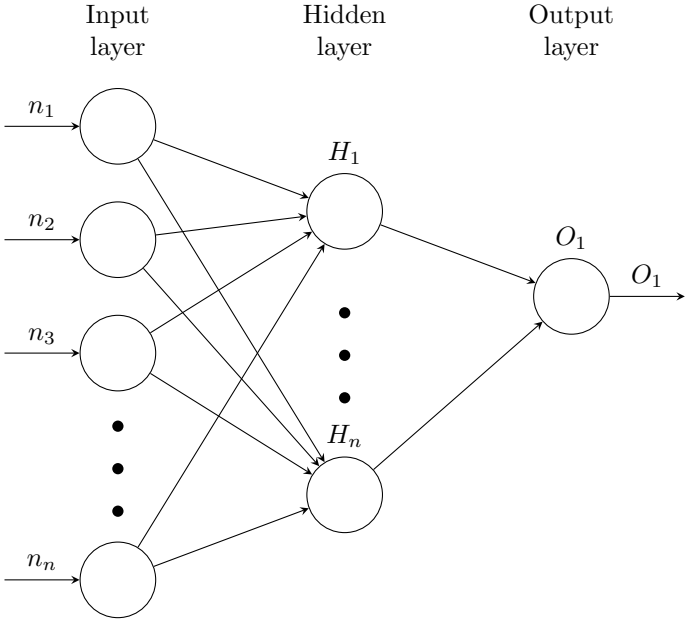
2.2.3 Epochs

The initial number of epochs was set to 100. During training, different epochs were evaluated and the results are presented below.

2.2.4 Hidden layer configuration

The initial number of hidden layers was set to 1, and number of nodes set to half of the input nodes as a starting point (thus 25 nodes). During training, different number of hidden layers and hidden layer nodes were evaluated and the results are presented below.

2.2.5 Architecture



2.2.6 Feed-forward

For each epoch, all instances of the training data were passed through the neural network. The output of a node in the next layer was calculated by:

$$H_n = b_n + \sum_{n=1}^{n} (w_i * n_i) \tag{3}$$

and for the output layer:

$$O_1 = b_O + \sum_{n=1}^{n} (w_i * h_i) \tag{4}$$

2.2.7 Back-propagation

Equations used for the backpropagation of the error were the same as the ones used in the lecture slides (I am lazy to type them out in latex).

2.2.8 Loss function

The loss function used to calculate the total error at the end of each epoch was binary-cross entropy. This was used as this is a binary-classification neural network - and provided accurate results.

$$-(y \log(p) + (1 - y) \log(1 - p)) \tag{5}$$

2.2.9 Results

Different learning rates and different number of epochs:

Results of different permutations			
Learning rate	No. of epochs	No. of nodes in hidden layer	Final output error
0.02	250	37	79,11
0.02	500	37	17,25
0.02	750	37	15,53
0.02	1000	37	13,41
0.04	250	37	9,43
0.04	500	37	7,48
0.04	750	37	6,75
0.04	1000	37	6,31
0.06	250	37	6,26
0.06	500	37	4,90
0.06	750	37	4,84
0.06	1000	37	4,78
0.08	250	37	6,74
0.08	500	37	5,67
0.08	750	37	5,30
0.08	1000	37	5,20
0.1	250	37	6,61
0.1	500	37	5,65
0.1	750	37	5,52
0.1	1000	37	5,49
0.2	250	37	6,62
0.2	500	37	6,26
0.2	750	37	5,51
0.2	1000	37	5,40
0.3	250	37	20,39
0.3	500	37	14,91
0.3	750	37	9,67
0.3	1000	37	8,88

As can be seen here, the lowest output error of the network is when the learning rate is 0.06, number of epochs is 1000, number of hidden layers is 1, and number of nodes in the hidden layer is 37.

2.3 Testing

Using the results formulated above, and training the network with a learning rate of 0.06, 1000 epochs, 1 hidden layer, and 37 nodes in the hidden layer - the network is trained with a loss of 4.78. The remaining 20% of the dataset, which was not used for training the network, is used for testing.
Results:

- Total instances of test data set: 58
- Accuracy: 81.03%

- True positive: 24
- True negative: 24
- False positive: 10
- False negative: 0
- Precision: 0.686
- Recall: 1

2.4 Conclusion

The neural network had an accuracy of 81,03%, with a precision of 0.686 and recall of 1. A factor that can be introduced to improve the accuracy of the neural network is increasing the size of the data set, as 286 instances (221 for training) is small for training a neural network to generalize over some data set.

3 Genetic Programming Classification Algorithm

3.1 Initialization

This algorithm's parameters was initialized with the following:

- Population size: 100
- Number of generations: 50
- Selection method: Tournament selection (5 candidates)
- Genetic operators: Mutation, crossover
- Mutation rate: 0.1

The genetic program tree is initialized with each node containing an attribute from the data set, and each leaf node containing a classifier decision (occurrence-events or no-occurrence-events).

3.2 Fitness function

Mean squared error was used to calculate the fitness after each generation. By squaring the residuals, it penalizes the program more heavily for larger mistakes and less for smaller ones, leading to a better model overall. This characteristic is particularly useful so that the algorithm can be as accurate as possible and not allow for large deviations.

$$\sum_{i=1}^{D}(x_i - y_i)^2 \tag{6}$$

3.3 Training

The genetic program also uses the same pre-processed one-hot encoded data. During training, the algorithm feeds the training data into the genetic program over a number of generations. In each generations, all instances of the training set are used. A population size of 100 ensures that there are 100 possible output candidates for tournament selection to occur on.

3.4 Results

Results of genetic program over generations		
Mutation rate	Tournament Selection Candidates	Best fitness
0.2	5	0,66
0.4	5	0,75
0.6	5	0,80
0.8	5	0,42
1.0	5	0,36

As can be seen here, the lowest best fitness of the network is when the mutation rate is 0.6, tournament selection size is 5 and the other parameters the same as when initialized.

3.5 Testing

Using the results formulated above, and using the remaining 20% of the dataset, which was not used for training the algorithm, is used for testing. Results:

- Total instances of test data set: 58
- Accuracy: 78,91%
- True positive: 27
- True negative: 26
- False positive: 6
- False negative: 0
- Precision: 0.816
- Recall: 1

3.6 Conclusion

The genetic program had an accuracy of 78,91%, with a precision of 0,816 and recall of 1. It does not perform as well as the neural network because genetic programs which evolve decision trees are more suited for data that is discrete - while the breast-cancer data set is continuous.

4 C4.5 Decision Tree

4.1 Introduction

The weka machine learning tool was used for this section (J48).

4.2 Initialization

Default values of the model were used.

4.3 Results

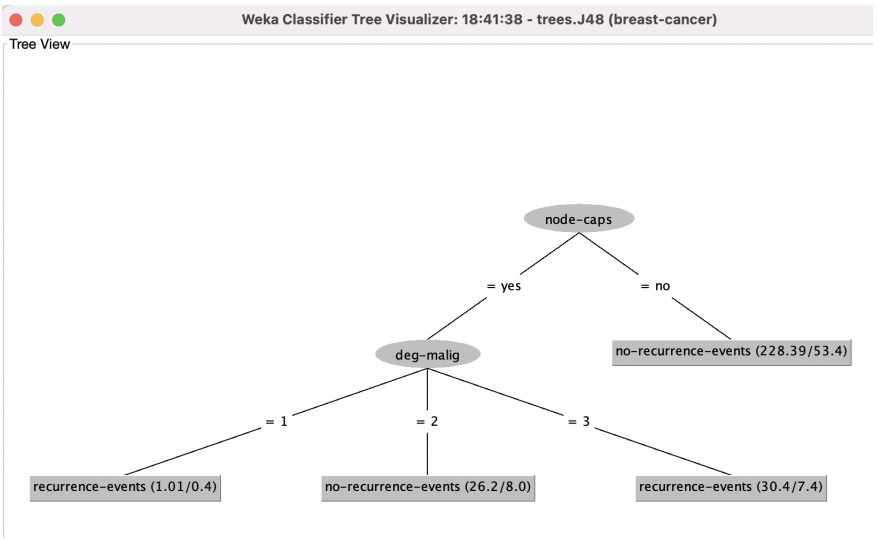


Figure 1: The size of the tree is 6, and number of leaves is 4. The model correctly classified 216 instances (75.52%), and incorrectly classified 70 instances (24.48%).

For no-recurrence-events:

- TP Rate: 0.960
- FP Rate: 0.729
- Precision: 0.757
- Recall: 0.960
- F-Measure: 0.846

For recurrence-events:

- TP Rate: 0.271
- FP Rate: 0.040
- Precision: 0.742
- Recall: 0.271
- F-Measure: 0.397

4.4 Conclusion

The weka machine learning tool produced a similar decision tree as the one produced by the genetic program - thus the results of this tool can be said to be equal to the GP.

5 Comparison of all learning methods used

Comparison of algorithms		
Algorithm	Best accuracy	Lowest Error
ANN	81.03	4.78
GP	78.91	N/A
Weka	75.52	0,88

The best performing algorithm was the Artificial Neural Network.