Comparrison of Backpropogation, Genetic method, and Decision Tree method for Diabetes Diagnosis Prediction

March 28, 2022

1 Introduction

1.1 Project Motivation

The motivation of this project originated from curiosity about the nature of neural networks and the efficiency of the different methods used to train them.

1.2 Aims and Objectives

The objective of the project is to test the methods of backpropogation, genetic algorithms, and decision trees on a diabetes diagnosis training set.

1.3 Report Structure

2 Background and History of the Study

Originally only concentrating on neural networks that utilize decision trees, the scope of the project has grown to include the learning methods of backpropogation and genetic algorithms as time has permitted. Neural networks produce result through the many interactions between its neurons. The neurons are organized in layers; The first layer is the input layer, where data is

first given to the network. The last layer is the output layer, which takes input from other neurons and outputs the result of the network. Between these layers are zero or more hidden layers. They function the same as the output layer, taking input from the layer before it and producing an output.

3 Implementation

The first steps to conducting the experiment were to craft each of the different neural networks.

3.1 Backpropogation

The backpropogation network was first, originally utilizing a graph data structure for its implementation. The network was first ordered by layers, with the nodes of the graph (neurons) being organized by layer in arrays, with each node being connected to every node in the layer after it. Nodes fired after all of the nodes connecting to it had already fired, with the output layer yielding a result right after the input layer was set with a sample to train on.

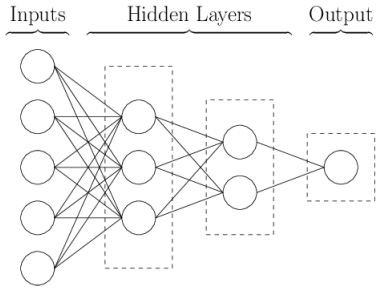

Figure 1: Example of a Neural Network [5]

Output from these neurons is produced by taking the sum of all its inputs (I_i), each multiplied by a weight (W_i), and subtracting the neuron's bias (b). The result of this summation is then passed into the activation function to produce the final output of the neuron. The activation function scales the summation to be between zero and one. The sigmoid (σ)function was used in this experiment.

$$Y = \sigma[\sum_{i=1}^{n}(I_i W_i) - b] \tag{1}$$

One of the most popular methods of training neural networks is through backpropogation. In this method, the network is fed how different each layer's output was from the correct output, starting at the output layer and ending at the first hidden layer.

The error between the expected output and the observed output (e) is calculated by taking their difference. The error gradient of the neuron, used to calculate the change in each weight, is calculated by using a formula similar to 1 but using the derivative of σ instead of σ itself multiplied by the error (e).

$$\delta = \frac{d\sigma}{dx}[\sum_{i=1}^{n}(I_i W_i) - b] * e \tag{2}$$

The change for each weight (ΔW_i) in the output nodes is calculated by multiplying the error gradient (δ), a learning rate constant (α), and the previous output (Y_p) of the node together.

$$\Delta Y = \delta * \alpha * Y_p \tag{3}$$

For the hidden layers, then change in their weights are similar. Instead of the error of the output nodes, however, the summation of each error gradient of the nodes in the previous layer (δ_j for node j in the previous layer) multiplied by their weights for the current node (W_j).

$$\delta = \frac{d\sigma}{dx}[\sum_{i=1}^{n}(I_i W_i) - b] * \sum_{j=1}^{n_p}(\delta_j W_j) \tag{4}$$

When this process is repeated for many epochs over one data set, the output of the network becomes more tuned to the desired result.

3.2 Genetic Algorithm

Genetic algorithms are quite different from backpropogation, utilizing trends within nature as inspiration. The training begins with a population of randomly generated individuals. Each indivudual is composed of a genome which describes every weight and bias of every neuron within a neural network. Each gene within that genome represents the set of all weights (one from each node in layer i) relating to the connections of one node in layer $i-1$.

As the training begins, each member of the polulation has its fitness evaluated. The evaluation function for this project is the reciprocal of the loss

function, $fitness(memberId) = \frac{1}{loss(memberId)}$. Two parents from the population are then selected using the roulette wheel method, where parents are randomly chosen, biased twoard parents with high fittness scores. The genomes of the parents are crossed between two new children. A point on the genome is randomly chosen (p) where child genes are composed of correcponding parent genes for the section up to p then composed of the oppisite parent after p.

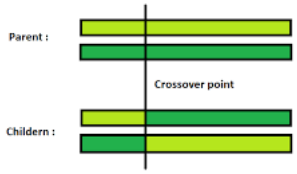

Figure 2: Illustration of crossover [2]

This ensures that there is a good mix between the characteristics of the parents passed down to the children.

Each child may then undergo a mutation, where an individual gene is mutated to a random value. This serves the purpose of guiding genes out of local maximums (genes that evolve to do well for a specific subset of samples, but not the samples as a whole).

This cycle continues until the amount of children produces equals the size of the original population. The population is then replaced with the children. This represents on generation of the neural network. When this process is repeated for many epochs over one data set, the output of the network becomes more tuned to the desired result.

3.3 Decision Tree

4 Experimental Approach

4.1 Data Set and Preparation

The data set used for this project is from the Vanderbilt University Department of Biostatics and contains the testing parameters and results of 390 African American participants from central Virginia [3]. The parameters

of the dataset are comprised of the patient's cholesterol, blood glucode levels, high-density lipoprotein (HDL) or "good" cholesterol, the ratio of HDL cholesterol to total cholesterol, age, gender, height, weight, body mass index (BMI), systolic blood pressure, and diastolic blood pressure. The entries in the dataset are categorized into two groups, those that were diagnosed with diabestes and those who were not. For this experiment, the height and weight parameters were exclused as they were redundant with BMI. The cholesterol ratio parameter was also omitted for redundancy to the other cholesterol parameters.

Before usage in training, the data set was first normalized. When a data set is normalized, all of its values are scaled between zero and one. After normalization, the highest value in the data set is scaled near to 1 and the lowest is scaled near to zero. Normalization allows the neural net to only deal with smaller numbers without removing the scale of the original data. The next step was to then classifiy the data into diebetic and non-diabetic groups and then to split the data into training and testing sets using an 80:20 split.

After pruning unneeded parameters, normalizing the data points, and classifying the data, the data set is ready to be used in the neural network.

4.2 Network Structure and Training

After pruning, redundant parameters, the data set was left with parameters. The input layer of each layer therefore had 8 nodes in the input layer. Hidden layers with 5 nodes or under resulted in low computation times, but no hidden layer at all performed the best across the various methods. The ouptut layer has two nodes, representing the confidence that the given participant does or does not have diabetes.

For training, each one of the three networks was given the same 80% proportion of the original data set and trained for 150 epochs. The loss function for each network is given by the sum of squared errors for every sample.

$$loss = \sum_{i=1}^{n_{samples}} \sum_{j=1}^{n_{outputs}} (output_{ij} - expected_{ij})^2 \tag{5}$$

Each trial was given a score for comparrison. The score is given by the percent of correctly identified diagnoses divided by the time it took for the trial to complete.

For the backpropogation network, a learning rate (α) of 0.6 was used. Across three trials, training over the data set for the 150 epochs, the network trained after 0.75 seconds on average. The average loss for the testing set was 0.125, correctly predicting 84.2% of diagnoses. Its overall score was 1.08.

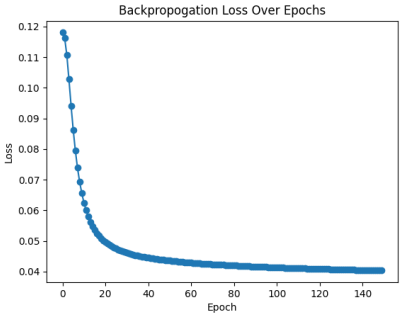

Figure 3: Loss value over successive epochs for backpropogation

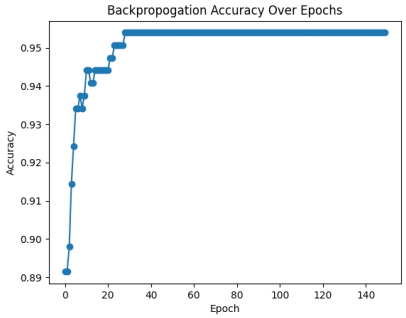

Figure 4: Diagnosis accuracy over successive epochs for backpropogation

The training loss curve for backpropogation follows closely to a curve of $\frac{1}{x}$. This signifies a good loss curve with minimal overfitting [4]. The loss declines sharply and the accuracy rises rapidly over successive epochs of training.

For the genetic algorithm, a population size of 6, a crossover rate of 0.5, and a mutation rate of 0.1. Across three trials, the network took 4.95 seconds. The avarage loss for the testing set was 0.295, correctly identifying 68.4% of diagnoses. Its overall score was 0.138.

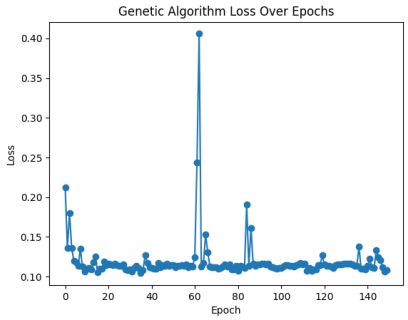

Figure 5: Loss value over successive epochs for the genetic algorithm

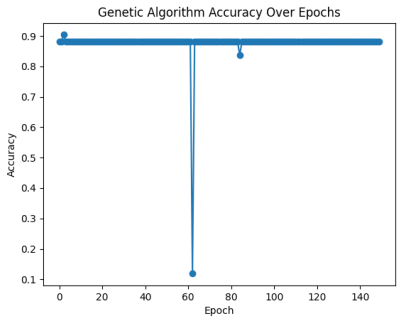

Figure 6: Diagnosis accuracy over successive epochs for the genetic algorithm

The training loss curve for the genetic algorithm shows a less pronounced curve, correcponding well with a high learning rate [4]. The loss of the algorithm dropped sharply after the training started but leveled out higher than backpropogation did, resulting in a high accuracy, but it did not improve significantly over time.

5 Results and Discussion

6 Conclusions

References

[1] MD Benjamin Wedro. Cholesterol management, Jul 2020.

[2] Avik Dutta. Crossover diagram, Jun 2019.

[3] Frank E Harrell. Diabetes data set, Dec 2002.

[4] George V Jose. Useful plots to diagnose your neural network, Oct 2019.

[5] Markus V. S. Lima. Neural network diagram, May 2018.