1.5em 0pt

*Option B: Analysis and Report using WEKA
*Department of Computer Science*
*Brock University*
St.Catharines, Ontario, Canada
mr18kq@brocku.ca

# Fall 2021 COSC 3P71 Artificial Intelligence: Assignment 3

Micah Rose-Mighty

***Abstract*—This report is a detailed comparison and analysis of a neural network implemented in WEKA for performing data classification on two popular machine learning classification problems. The problems that will be analyzed in this report involve the Breast Cancer Wisconsin Data Set and the Iris Data Set, both found in the UCI datasets archive. These datasets are excellent test subjects when analyzing the performance of a neural network due to the nature of their respective data. The job of the neural network is to use these datasets to classify the data and the performance of the neural network will be analyzed using WEKA. These datasets contain quantitative data each pertaining to different attributes of the elements in each respective dataset. These attributes act as an input in our neural network and the classification of each element is our output. WEKA will be used to help analyze the performance capability of the neural network when classifying this data and the results of those experiments will be included in this report.**

***Index Terms*—neural network, machine learning, supervised learning, learning rate, hidden nodes, momentum factor, epochs, cross validation styling, insert**

## I. Introduction Problem Definition

This report analyzes the performance of a neural network using two popular machine learning classification problems. These neural networks will be implemented and analyzed using WEKA for increased accuracy and efficiency. Our problems are presented in the form of two distinct datasets. These datasets each present a distinct classification problem, in the case of the Iris Data Set we are given data regarding three different classifications of the Iris plant. These classifications are the Iris-setosa, Iris-versicolor and Iris-virginica. Within the dataset, there are four attributes used to describe each of the 150 Iris plants in the dataset. These attributes pertain to different measurements of an Iris plant such as petal length or sepal length. The neural network must use these measurements to attempt to accurately classify these plants, the performance of this classification will be thoroughly analyzed in this report. Within the Breast Cancer Wisconsin Data Set we are given ten attributes pertaining to different cell nuclei and breast cancer clusters. The neural network must use these attributes to classify whether a tumor is malignant or benign. In this case, the input to the neural network will be the attributes of each case and the output would be the classification of the tumor. The performance of the neural network in these two distinct classification problems will be used to analyze the behaviour of a neural networks when various different parameters are modified. More simply, this report will analyze neural networks and the impact that parameters can have on the performance of a neural network. The parameters that will be varied in this report to aid analysis will be the learning rate, number of hidden nodes, momentum factor, number of epochs, number of folds used in cross validation and percentage used in a percentage split approach. This report will summarize the results of these numerous experiments involving neural networks to help further understand their behaviour and how the variation of parameters within the neural network can impact overall performance.

## II. Background

### A. Feed Forward Neural Networks

The specific form of a neural network that will be analyzed in this report is known as a feed forward network. The feed forward network is the simplest of all neural networks and the concept can be very clearly explained and visualized. The first thing to know about feed forward networks is that there are two different classes of feed forward network architectures. One class of feed forward neural networks is the single-layer feed forward network. The architecture of the single-layer feed forward neural network involves an input layer of source nodes each connected to an output layer of neurons by various weighted links. In a neural network the input is any high-dimensional discrete or real value, the input layer does no computation it solely objectifies any input data. In this specific experiment, the input is the various numerical attributes in each dataset. The output layer of neurons in a neural network each contain a transfer function and once the correct input is received to activate the neuron, a signal is sent to the outputs. The other class of feed forward neural networks, and the main focus of this report, is the multi-layer feed forward network. The architecture of the multi-layer feed forward network is similar to that of the single-layer feed forward network except for one major difference. Within the multi-layer feed forward network there is one or more hidden layers of nodes that operate similarly to the output nodes in the single-layer neural network without actually sending a signal to the external environment but to the next layer of nodes. The architecture of a neural network is directly linked with the learning algorithm used to train neurons in a neural network and facilitate neural network learning. The encompassing objective of neural network learning is to find parameter settings that minimize error given a set of training examples, basically to obtain a neural network that behaves correctly on new problem instances of the same learning task. This is done through the actual architecture of the neural network and the learning algorithm of the neural network. The learning algorithm is used for training the network by modifying the weighted links to solve the learning task problem accurately on the set of

training examples. Depending on the architecture of the neural network there are many different learning algorithms that can be applied. The subsequent sections will thoroughly describe these learning algorithms and their relevance to the problems in this report and neural network learning as a whole.

### *B. Error Correcting Learning*

The idea of Error Correcting Learning strongly correlates to the perceptron concept. The initial definition of a perceptron is a machine that learns, using examples, to assign input samples to different classes using linear functions of the inputs. The perceptron was later described as a stochastic gradient-descent algorithm that tries to linearly separate a set of multi-dimensional training data. More simply, perceptrons can be described as single-layer feed forward networks with each output neuron independent of the others. The output neuron of the perceptron has an activation function that is contingent on the input and the weighted links. The theory behind percepton learning is that given enough training examples, there is a learning algorithm that will learn any linearly separable function.

Linearly Separable: Data is linearly separable if it is possible to draw a line on a graph that will separate all data of one type of one side of the line and the data of another type on the other side of the line

This theory is based upon the perceptron rule that converges to weights that accurately classify all training examples, given the dataset represents a function that is linearly seperable. The algorithm behind the perceptron rule can be easily described in 2 steps: Step 1. Assign random weights to the initial network Step 2. Repeat for several epochs until convergence

Epoch: A single pass of the entire training dataset by a learning algorithm

Given this perceptron rule and formula it is evident that this rule is guaranteed success given the training examples are linearly separable and the learning rate is sufficiently small. This will be insufficient for our given classification problems in this report which is why the main focus will be multi-layer feed forward networks, as stated in the previous section. Given the definition of linear separability it is clear that neither the Iris or Wisconsin Breast Cancer Data Set fit that description. The architecture of a multi-layer feed forward network is better suited for the back-propagation learning algorithm and this idea will be further discussed in the following section.

### *C. Back-Propogation Learning*

Back-Propagation learning takes the network topology(all units and their connections), termination criteria, learning rate(constant of proportionality of gradient-descent search, initial parameter values and a set of classified training data as input and gives updated parameter values as the output. Back-propagation learning is described as a gradient-descent search through the parameter space to gradually minimize error. Back-propagation learning can be used to train multi-layer feed forward networks if the activation function of the output is differentiable. Back-propagation learning is very similar to perceptron learning and has a few key principles:

- If the neural network computes an output that matches the desired output then nothing is done.
- If there is a difference between the actual output and desired output then the weights are adjusted to minimize the error.
- Assess the fault of the error and divide it among the corresponding weights

The exact methodology behind Back-propagation is easily described in the pseudo-algorithm seen below:

> Randomly choose initial weights termination criterion not met each example in training set randomly selectedApply the inputs to the neural network. Calculate the output for each neuron starting from the input layer, through the hidden layer(s) to the output layer. Calculate the error at all the outputs, by finding the difference between the desired output and actual output. Using the previously calculated output error, compute the error for the nodes of pre-output layers. Using the calculated error for the nodes of the pre-output layers, compute the necessary weight adjustments apply the weight adjustments.

Although Back-propagation learning is very useful in the classification problems of this report among other problems, there are a few possible errors that can occur in a gradient-descent search algorithm such as this. Some of the possible errors seen in back-propagation learning algorithms among other gradient-descent algorithms are:

- Convergence toward the wrong paramaters(detecting bad local minima when global minimun is present)
- Lack of convergence to desired parameters due to small gradient factor(Quasi-standstill)
- Oscillation between incorrect parameters endlessly
-

Within a Back-propagation learning algorithm specifically, a low learning rate can cause your algorithm to learn too slow but a high learning rate can lead to non-convergence. This makes it so important to find the correct parameters to ensure optimal performance of the neural network. The method to find the optimal parameters of a neural network relies on the strength of the weight updates of the layers. In a good Back-propagation learning algorithm, the weight updates closer to the input layer change slower and the weight updates closer to the output layer change faster. The reason for this is that a smaller and smaller portion of the overall error between the desired output and actual output is attributed to the nodes further and further back from the output layer. To ensure optimality of a neural network the learning rate for each weight must be updated based on time. Another important concept of Back-propagation learning algorithms is momentum functions. momentum functions have been used to improve learning time and avoid local minima when completing a gradient-descent search. The basic concept of momentum functions is to add a portion of the previous weight change to the current

weight change. There are many different forms of momentum functions that all follow the same principles and serve the same purpose. In Back-propagation learning algorithms the idea of generalization is employed. The goal of generalization is to train the neural network to a reasonably low global error. Neural networks with too many hidden layers allow for too many solutions and take too many weight changes to reach a global minima and therefore do not generalize well. Cross validation is also employed when working with Back-propagation learning algorithms. Cross validation breaks the data into 3 random groups:

- Training
- Testing Validation

Error curves are plotted for both training and validation, validation data is used to ensure that the training set is correct and accurately represents the attributes of the dataset. When cross validating the training and validation data should have the same error and similar error curves. This entire idea of data classification using neural networks is contingent on the idea of Supervised Learning which will be thoroughly described in the next section.

### *D. Supervised Learning*

Machine learning is concerned with programs that can learn from experience, but there are different types of learning experiences each of which present a different type of problem. There is a Supervised Learning experience in which the correct answers for each problem are known and the neural network must then learn a function that best approximates the relationship between the input and output based upon the given correct answers and desired outputs. This Supervised Learning experience is present in many problems including classification, regression and pattern recognition problems. Since the problems analyzed in this report are classification problems we will focus on Supervised Learning as it directly correlates with our neural networks being analyzed. Supervised Learning facilitates all of the training and learning algorithms previously mentioned in this report as it is used to provide the desired output and calculate the error within a neural network. Without Supervised Learning, the classification learning task would be impossible to solve using a neural network as there would be no real desired output and no way to calculate the error and make the necessary adjustments to the neural network. Given that our problems within this report are classification problems, knowledge of Supervised Learning will be detrimental to the operation and analysis of these neural networks.

## III. Methods of Analysis

In this report we will analyze the behaviour of a neural network given two different data classification problems and different parameters. The neural networks will be implemented and analyzed using WEKA. The main focus of this report is to analyze the efficiency of accuracy of a neural network given the nature of different data input and parameters. The different data classification problems that will be analyzed in this report are the Iris Data Set and Breast Cancer Wisconsin Data Set(see Introduction and Problem Definition section for detailed description of these datasets.). The different parameters that will be varied and monitored will be the learning rate, the number of hidden nodes, the momentum factor, the number of epochs used in training, the number of folds used in cross validation(when applicable) and the percentage used in the percentage split approach(when applicable). The performance of the neural network given the different parameters and datasets will be thoroughly analyzed and explained in this report. Some of the performance attributes of the neural network that will be analyzed using WEKA are:

- Time taken to build the neural network model.
- Time taken to test the neural network model on the test split
- Correctly Classified Instances
- Incorrectly Classified Instances
- Relative absolute error
- Total Number of Instances
- Confusion Matrix

When analyzing these neural networks the nature of the datasets will also be taken into account to give a more accurate interpretation of the results. The main purpose of this study is to analyze the performance behaviour of neural networks given different parameters and datasets in order to further understand neural networks and machine learning capability under different circumstances and restraints.

The neural networks in this report will be analyzed rigorously in order to ensure the accuracy and validity of this report. Each neural network will begin with a set of default parameters. In the case of the percentage split approach, the default parameters for each neural network run will be:

- 70% percentage split
- a nodes in the hidden layer (a = (total number of attributes + total number of output classes)/2
- 0.3 or 30% learning rate
- 0.2 or 20% momentum factor
- random number seed to ensure statistical accuracy
- 500 epochs of training

In the case of the cross-validation approach the default parameters for each neural network run will be:

- 10 folds used in cross-validation
- a nodes in the hidden layer (a = (total number of attributes + total number of output classes)/2
- 0.3 or 30% learning rate
- 0.2 or 20% momentum factor
- random number seed to ensure statistical accuracy
- 500 epochs of training

Each different parameter set will be run 15 times for each dataset with the results averaged out to promote accuracy even further. To actually analyze the behaviour of the neural networks in these classification problems, with emphasis on parameter impact, one of the above mentioned parameters will be altered while the others remain the same. The neural network will then be run 15 times and the results will be averaged out and relayed in this report. This process will be

repeated until all parameters have been altered individually and the results of these numerous experiments will be in this report. The actual results of these experiments are seen in the next section of the report.

## IV. Results and Discussion

This report begins with the percentage split approach default parameter set applied to the neural network given the Iris Data Set. The details of the percentage split approach default parameter set can be seen in the Methods of analysis section. This default parameter set is being tested to give a basis for the performance of a neural network given this specific classification problem and the results will be used as a gauge on whether certain parameters improve or impair the given neural network. To help aid comprehension in this report the parameters and results for each neural network run will be given in point form accompanied by captions.

### *A. Percentage Split Approach with default parameters on Iris Data set:*

- 0.05 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 5.5698%
- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

This being our default set does not tell us much about he behaviour of a neural network but once the variation of the parameters begins we will start to see the impact that parameters can have on a neural networks performance. The first parameter that will be altered will be the learning rate. First, the learning rate will be doubled from the default 0.3 to 0.6 and the results will be below.

### *B. Percentage Split Approach with 0.6 learning rate on Iris Data set:*

- 0.05 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 4.9907%
- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

We can see that by doubling the learning rate, the only difference is a small decrease in the Relative Absolute Error which means that an increased learning rate leads to a small increase in the performance of our neural network model. Next, I will divide the default learning rate by a factor of 2, making 0.15 our new learning rate and then analyze the results.

### *C. Percentage Split Approach with 0.15 learning rate on Iris Data set:*

- 0.06 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 7.2147%
- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

In this run, with a decreased learning rate, it takes slightly longer to build the model than in the default run. Another difference is the increase in Relative Absolute Error in this run in comparison with the default run. It is clear to see that a low learning rate can negatively affect the performance of a model.

Next, we will analyze hidden nodes and their impact on this given model. Of course, the default run has already been taken and has given a good standpoint to judge our further findings on. To analyze the impact of the number of hidden nodes on a neural network, a similar approach to that of the learning rate analysis will be taken. I will first double the number of hidden nodes in the single hidden layer and monitor the impact.

### *D. Percentage Split Approach with a*2= 6 hidden nodes on Iris Data set:*

- 0.09 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 5.2295%
- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

Doubling the amount of hidden nodes in the single hidden layer increases the time taken to built the model. This makes sense because extra nodes are being added to the neural network. Another strange change was a slight decrease in the Relative Absolute Error in comparison to that of the RAE in the default experiment. According to these experiments, adding extra nodes to a single hidden layer can slightly increase the performance of a neural network model. To verify this idea of the impact of hidden nodes on the performance of neural networks, a similar experiment has been done but instead the a value has been divided by a factor of 2. The results are below.

### *E. Percentage Split Approach with a/2= 1.5 = 2 hidden nodes on Iris Data set:*

- 0.04 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 6.0579%

- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

The decrease in hidden nodes caused the model to be built quicker than in the default run and the RAE increased. That verifies the idea mentioned in the prior paragraph that states that the number of hidden nodes impacts the overall performance of the neural network. When hidden nodes increase, the neural network performs slightly better. When hidden nodes decrease, the neural network performs slightly worse.

The momentum will be analyzed in n identical fashion to those parameters analyzed in prior paragraphs. The momentum factor will be first doubled from the default value and then analyzed.

*F. Percentage Split Approach with 0.4 momentum factor on Iris Data set:*

- 0.04 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 5.6671%
- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

In this case, practically nothing has changed except for a slight decrease in the build time. Since that is the case, an impactful increase in the momentum factor will be applied to see if it will have a greater impact on the performance of the neural network.

*G. Percentage Split Approach with 0.9 momentum factor on Iris Data set:*

- 0.14 seconds to build the model
- 0 seconds to test the model on the test split
- 45 total instances taken from the dataset
- 44 instances were classified correctly
- 1 instance was classified incorrectly
- Relative Absolute Error is 4.8326%
- the incorrectly classified instance was an Iris-virginica flower classified as an Iris-versicolor

We can see here that given a higher momentum factor, the time to build the model increases while the RAE decreases. That tells us that a higher momentum leads an improved neural network model but this model will take longer to build.

Unfortunately, as seen in my email sent to the Professor and Teaching Assistant, I was unable to complete this report to the best of my abilities within given deadline. Please be considerate of all my efforts when grading my submission. Thank you in advance