"An incremental framework based on cross validation for estimating the architecture of a multilayer perceptron, implementation of the MOST framework."

PROJECT REPORT

Submitted in partial fulfilment for
the award of the degree of

BACHELOR OF TECHNOLOGY
In
Computer Engineering

By

Pratik Varshney
11-PEB-002

Abhay Mittal
11-PEB-005

Under the guidance of
Mr. Nadeem Akhtar

Department of Computer Engineering
Zakir Husain College of Engineering and Technology
Aligarh Muslim University
Aligarh (U.P.) – 202002

Z. H. College of Engineering & Technology

Aligarh Muslim University

CERTIFICATE

This is to certify that the project entitled "An incremental framework based on cross validation for estimating the architecture of a multilayer perceptron, implementation of the MOST framework" being submitted by Abhay Mittal and Pratik Varshney as their minor project for the 5th semester in partial fulfilment of the degree of Bachelor of Technology in Computer Engineering is a record of their own work carried out under my supervision and guidance.

(Mr. Nadeem Akhtar)

Department of Computer Engineering
Zakir Husain College of Engineering and Technology
Aligarh Muslim University
Aligarh - 202002

ACKNOWLEDGEMENT

We would sincerely like to thank our supervisor Mr. Nadeem Akhtar who guided and encouraged us throughout the project and provided us with valuable suggestions whenever we faced any problems. He regularly checked our progress and provided his valuable time and effort throughout the project.

We would also like to sincerely thank our senior Mr. Sandeep Chauhan, PhD Scholar, IIT Bombay who helped us in selecting this project and removing our doubts. He also helped us in becoming familiar with the topic of MLPs and cross–validation.

We also thank the library staff of our department. Finally, we would also like to thank our family members and friends for supporting is and encouraging us throughout our study.

ABSTRACT

For any particular application it is generally very difficult to estimate the number of hidden units or layers for a MLP network even for an expert user. Some Algorithms have been proposed to find a suitable architecture. The MOST (Multiple Operators using Statistical Tests) framework is one of them. It begins with an initial network and continuously generates new models and evaluates them till it finds the best network. The best Architecture may not always be the optimal architecture. The Multi Test algorithm performs an exhaustive search over all the possible architectures and returns the optimal architecture.

Table of Contents

1. Introduction

In this project we have implemented the MOST framework that incrementally modifies the structure and check for improvement using cross validation. We have considered MultiFwd variant of this framework to determine the best architecture because it uses all the operators of MOST and its results are generally very close to optimal architecture. We have also implemented the MultiTest algorithm to determine the optimal architecture and sort the architectures in the order of their preference. This ordering is then used to compute the distance (number of architectures) between the results of MultiTest and MOST algorithm. This project has been motivated from the paper [1] mentioned in the bibliography.

a. Artificial Neural Networks

Artificial Neural Networks are computational models of biological networks of neurons inside a brain. They are usually presented as a system of interconnected neurons that can compute values from inputs by feeding through the network.

Neural Networks are very helpful in areas where we can't formulate an algorithmic solution and have lots of examples of the behaviour we require.

In short, Neural Networks are a form of multi processor computer system with

- Simple processing elements
- A high degree of interconnection
- Simple scalar messages
- Adaptive interconnection between elements

The application areas of ANNs include quantum chemistry, game playing and decision making, pattern and sequence recognition, data mining, etc. ANNs are also used to diagnose several cancers.

b. Perceptrons

Perceptrons are one of the simplest types of neural networks. The basic concept of a simple perceptron was introduced by Frank Rosenblatt in 1957 at the Cornell Aeronautical University.

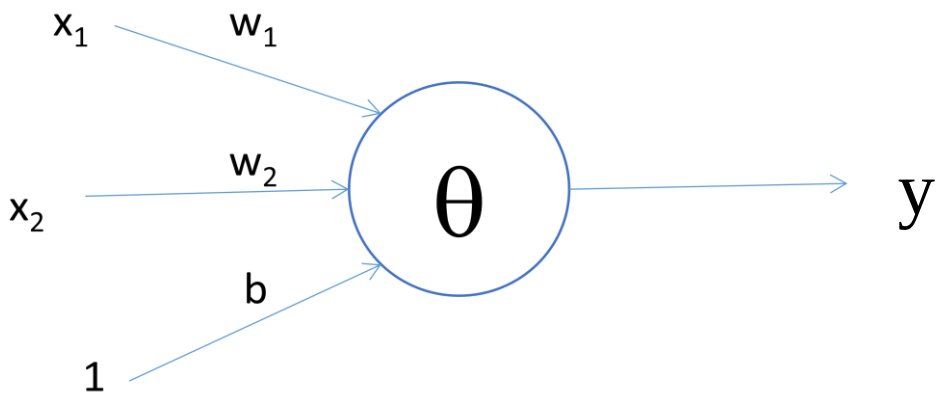


Figure 1 shows the graphical representation of a perceptron.

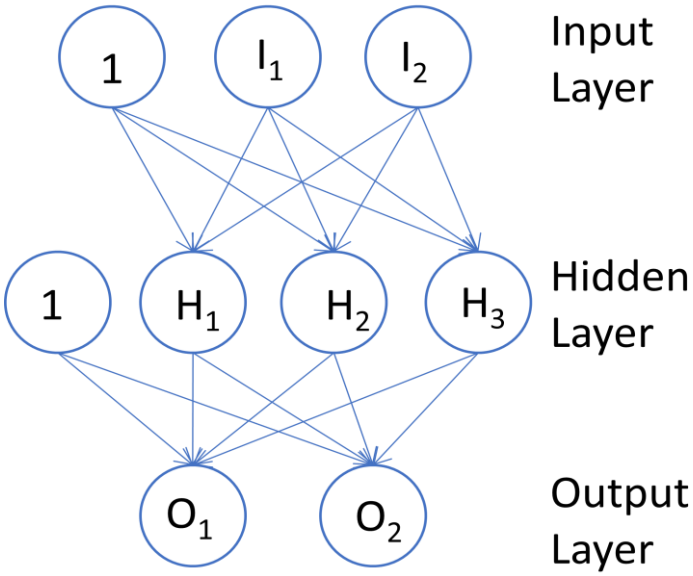
A perceptron has a number of external inputs, one internal input (called as bias), a threshold and one output. The output of a perceptron is always Boolean. All inputs have weights attached to them. The threshold determines whether the output of the perceptron will be 1 or not. It basically take all the weighted input value and adds them if the sum exceeds the threshold value, the output is 1, otherwise it is zero.

$$y = \begin{cases} 1, \text{ if } \sum w_i x_i + b > \theta \\ 0, \text{ otherwise} \end{cases}$$

An important feature of a perceptron learning rule is that if there exist a set of weights that solve the problem, perceptron will find the weights.

A perceptron can be trained to solve any 2 class classification problem where the classes are binary separable. For example, perceptrons can be used to solve the AND, OR, NAND, NOR problems. The perceptron is an elegantly simple way to model human neuron's behaviour.

c. Multi layer Perceptron (MLP)



A Multilayer Perceptron (MLP) is a feed-forward Artificial Neural Network model that maps a set of input data onto a set of appropriate outputs. It is formed by combining multiple layers of perceptrons. It consists of one or more hidden layer. Except for the input nodes (perceptrons), each node is a neuron with non-linear activation function. Each node in one layer connects with certain weight to every node in the following layer.

MLPs can act as non-linear classifier i.e. they can distinguish data that are not linearly separable and their output is not restricted to Boolean. It utilizes the back-propagation technique to train the network. Learning occurs in the perceptron by modifying the connection weights as the data processes, based on the amount of error in the output compared with the expected result.

Multilayer perceptrons using a back-propagation algorithm are the standard algorithm for any supervised learning pattern recognition process and the subject of outgoing research in computational neuroscience and parallel distributed processing. Since they are able to solve problems stochastically, they are widely used in researches to get approximate solutions for extremely complex problems like fitness approximation. They are widely used in speech recognition, image recognition and machine learning.

d. Back-propagation

The back-propagation algorithm is a common method of training artificial neural networks. It is a supervised learning method and is a generalization of the delta rule. To train the neural network we need training dataset which consists of input values and their corresponding output values.

The only requirement for the back-propagation algorithm is that the activation function used by the algorithm neurons is differentiable.

The back-propagation algorithm consists of two phases:

Phase 1:

1. The outputs of the first layer are calculated from the inputs provided.
2. Then the output of the next layer is calculated treating the outputs of the previous layer as inputs. This continues till all the output layers are determined.
3. After calculating the output, error in the output layer is determined and is back-propagated through all the layers.

Phase 2:

1. The gradient of the weight is determined by multiplying the output delta of each neuron with its input activation.
2. Then the gradient is multiplied by the learning rate and the result is subtracted from the weight. The greater the value of learning rate the faster the neuron trains, but lower value of learning rate results in more accuracy.

The back-propagation algorithm enables this neural network to learn and represent any mapping of input to output.

We used back-propagation based on stochastic gradient descent with adaptive learning rate and momentum.

e. Organization of the report

Chapter 2 covers the Tools and Resources used for the project. Then the next chapter explains the MOST and MultiTest Frameworks. The 4th chapter provides the results and the next chapter describes the conclusions that we have drawn.

2. Tools and Resources Used

a. MATLAB

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java. You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

Key Features

- High-level language for numerical computation, visualization, and application development
- Interactive environment for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- Built-in graphics for visualizing data and tools for creating custom plots
- Development tools for improving code quality and maintainability and maximizing performance
- Tools for building applications with custom graphical interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft Excel

b. Datasets

Datasets are retrieved from following sources:

- UCI Machine Learning Repository (*http://archive.ics.uci.edu/ml/datasets.html*)
- Delve (*http://www.cs.toronto.edu/~delve/data/datasets.html*)
- Statlib (*http://lib.stat.cmu.edu/*)
- Statlog (*http://www.liacc.up.pt/*)

3. The MOST algorithm for architecture selection

The MOST algorithm applies some pre-defined operators on the BEST network and generates candidate models. These models are then compared with the BEST in a breadth first manner to find the new BEST. The process is repeated till the algorithm is unable to find a better network than the BEST network.

a. Getting Started

i.Cross Validation

If we train a neural network with the complete dataset, we may obtain 100% accuracy but the neural network may predict very poorly when presented with new input data. This is called overfitting.

To avoid overfitting we divide the dataset into training data (to determine weights and bias) and test data (to evaluate the quality of the resultant neural network). This is known as Cross Validation.

1. K-fold Cross Validation

In K-fold cross validation, The dataset is divided into K-folds of roughly the same size. Then one of the folds is selected as the validation set and the remaining K-1 folds are chosen as the training set. The network is then trained and validated. Then another fold is chosen as the validation set and network is trained on the remaining folds. This process is continued till validation is performed on the folds. Here we have used 10 fold cross validation.

2. 5 x 2 Cross validation

In 5 x 2 cross validation, the dataset is divided into two folds. Then the network is trained on the first set and validated on the second. After this the network is trained on the second set and validated on the first. This process is repeated five times.

ii.Ordering of the networks on the basis of complexity

The networks are ordered on the basis of the number of hidden layers and nodes. The number of hidden layers is the first criterion to decide the complexity and the complexity increases as the number of hidden layers increase. If the number of hidden layers is same, the number of hidden nodes is considered as the deciding factor and the complexity increases as the number of hidden nodes increase.

iii.Determination of learning rate and number of epochs

Before running the MOST or the MultiTest algorithm, an initialization step is performed to determine the learning rate and epoch pairs for linear perceptron and multi layer perceptron networks for each cross validation technique used. Thus we

obtain four such pairs. For the initialization process we have used a multi layer perceptron network with a single hidden layer consisting of 10 hidden nodes.

The number of epochs is determined from the following set –

(50 – 100 – 150 – 200 – 250)

The learning rate is determined from the following set –

(0.001 – 0.005 – 0.01 – 0.05 – 0.1)

b. The MOST Operators

There are five operators that are used in the MOST framework.

i.REMOVE – n

This operator removes n hidden units from the last hidden layer of the network. The value of n can be specified as a percentage of the hidden units in that layer.

ii.REMOVE – 1

This operator removes a single hidden unit from the last hidden layer of the network.

iii.ADD – 1

This operator adds a single hidden unit to the last hidden layer of the network.

iv.ADD – n

This operator adds n hidden units to the last hidden layer of the network. The value of n can be specified as a percentage of the hidden units in that layer.

v.ADD – L

This operator adds a new hidden layer before the output layer and set the number of units in this new layer as:

H1: Number of nodes in the upper layer.

H2: Number of nodes in the upper layer x 2.

H3: Average number of nodes in the upper and lower layers.

H4: Average number of nodes in the upper and lower layers / 2.

c. The MOST Algorithm

The following is the algorithm for the MOST framework for architecture selection:

1) Select linear perceptron as the initial BEST network and set flagBestChanged=1.
2) Repeat steps 3 to 7 until flagBestChanged=0
   a) Generate candidate models(5) by applying each operator on BEST.
   b) Sort the generated models on the basis of complexity.
   c) Repeat for i = 1 to number of candidate models generated
      i) Train and validate candidate C(i) on K folds.
      ii) If C(i) is more complex than BEST, then
          If the expected error of C(i) is more complex than the expected error of best, then
              Set C(i) as the best network
              Set flagBestChanged = 1
              Break
          Else
              Set flagBestChanged = 0.
          End
      Else
          If Expected error of C(i) is less than or equal to the expected error of BEST, then
              Set C(i) as the BEST network
              Set flagBestChanged = 1
              Break
          Else
              Set flagBestChanged = 0
          End
      End
      iii) Go to step c
3) Return BEST

The flow chart for the algorithm is provided on the next page.

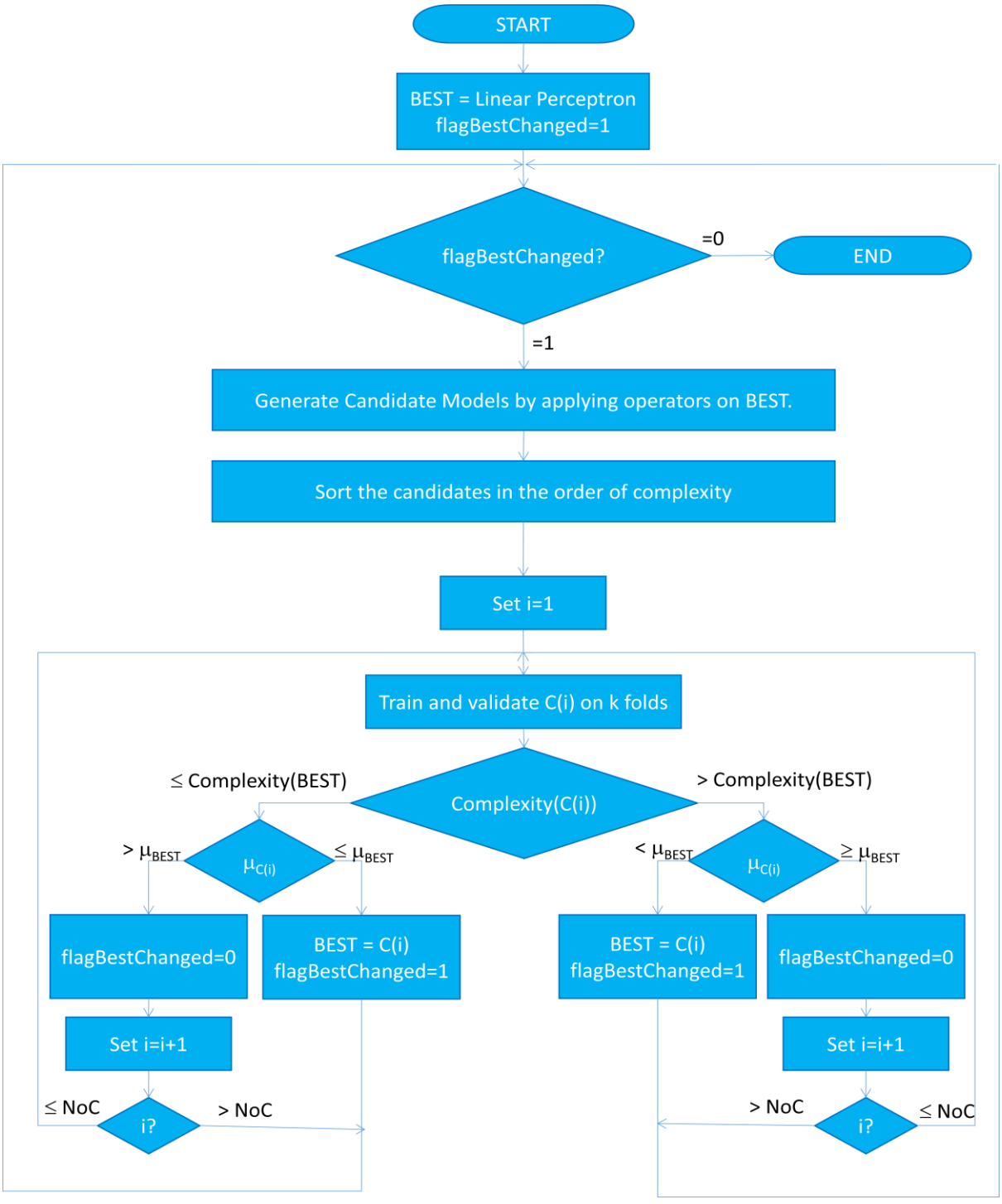


Figure 3: The MOST Algorithm

d. MultiTest

Given a dataset and a number of supervised learning algorithms, we would like to find the algorithm with smallest expected error. This can be achieved by using MultiTest Algorithm. It performs an exhaustive search on a set of architectures and returns the best architecture depending on error and prior preference. Thus, it is able to find a single best architecture which is not possible with most other algorithms.

It requires a list of trained and sorted architectures as input along with their errors.

MultiTest Algorithm:

1. Set k = number of architectures.
2. Initialize graph E with k nodes and no edges.
3. Set i = 1 and repeat steps 4 and 7 while i < k
    4. Set j = i + 1 and repeat steps 5 and 6 while j ≤ k
        5. If error in i^{th} architecture is more than the error in j^{th} architecture, create a directed edge in graph E from node i to node j.
        6. Set j = j + 1
    5. Set i = i + 1
8. Find all nodes with no outgoing edges to produce a set S.
9. Set l = lowest index in set S.
10. Return l^{th} architecture.

If we iterate steps 8 to 10, removing l^{th} node and edges incident to it at the end of each iteration, we get an ordering in terms of "goodness". This ordering is used to compute the distance between the architectures returned by MultiTest and MOST.

The flow chart for the algorithm is provided on the next page.

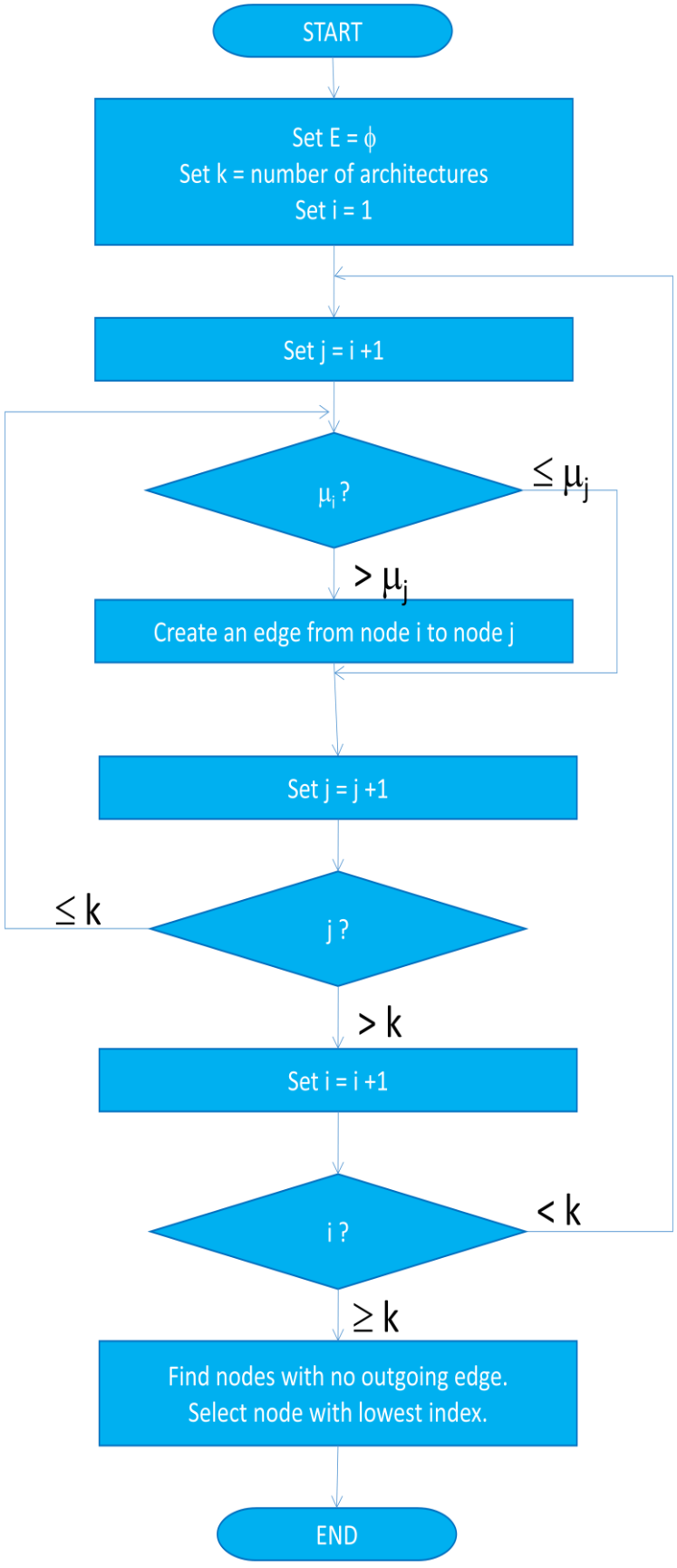


Figure 4: Flowchart for the MultiTest Algorithm

4. Results

The test was carried out on 4 datasets. The max number of nodes in any hidden layer was 10 and max number of hidden layers was 2.

a. Glass Dataset

Architecture	Learning Rate	Number of Epochs
Linear Perceptron	0.001	200
Multi Layer Perceptron	0.01	200

Algorithm	Error	Architecture
MOST	0.056186	Linear Perceptron
MultiTest	0.0436	Multilayer perceptron with two hidden layers containing 9 and 10 nodes respectively.

b. Thyroid Dataset

Architecture	Learning Rate	Number of Epochs
Linear Perceptron	0.05	150
Multi Layer Perceptron	0.001	200

Algorithm	Error	Architecture
MOST	0.02682	Linear Perceptron
MultiTest	0.0250	Multilayer perceptron with single hidden layer containing 7 nodes.

c. Wine Dataset

Architecture	Learning Rate	Number of Epochs
Linear Perceptron	0.05	200
Multi Layer Perceptron	0.1	200

Algorithm	Error	Architecture
MOST	0.010734	Linear Perceptron
MultiTest	0.0044	Multilayer perceptron with two hidden layers containing 3 and 9 nodes respectively.

d. Post-Operative Dataset

Architecture	Learning Rate	Number of Epochs
Linear Perceptron	0.05	250
Multi Layer Perceptron	0.005	150

Algorithm	Error	Architecture
MOST	0.14166	Multilayer perceptron with a single hidden layer containing 5 nodes.
MultiTest	0.1443	Multilayer perceptron with two hidden layers containing 2 and 4 nodes respectively.

Note: The results may vary with the experiments.

5. Conclusion and limitations

Our experiments in this project led us to the following conclusions:

1. The MOST architecture is highly efficient than the method of exhaustive search to determine the most suitable architecture. For e.g. - For the wine dataset, we were able to find the best architecture using MOST framework in less than 5 minutes while it took us more than 15 hours to find the best architecture using exhaustive search (MultiTest) where the maximum number of hidden nodes per layer was 35 and there were a maximum of 2 hidden layers resulting in 1261 possible architectures using 10 fold cross validation.

2. The architecture found by MOST may not be the optimal one.

6. Proposed Future Work

1. Further improving the efficiency of the forward variants of the MOST algorithm. Currently the framework starts with a linear perceptron network in the forward variants. We are trying to establish a way to determine an initial guess based on the characteristics of the dataset in order to reduce the time taken to achieve the best architecture.

2. We are also studying the effects of adding and removing random number of nodes as operators.

3. Using alternative methods of sorting the architectures in order to reduce the time complexity.

7. Bibliography

1. *Oya Aran, Olcay Taner Yildiz and Ethem Alpaydin, An incremental framework based on cross-validation for estimating the architecture of a multilayer perceptron, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 23, No. 2 (2009) 159–190.*

2. *O. T. Yıldız and E. Alpaydın, Ordering and finding the best of K > 2 supervised learning algorithms, IEEE Trans. Patt. Anal. 28(3) (2006) 392–402.*

3. Artificial Neural Network (*http://en.wikipedia.org/wiki/Artificial_neural_network*)

4. Perceptron (*http://en.wikipedia.org/wiki/Perceptron*)

5. Multilayer Perceptron (*http://en.wikipedia.org/wiki/Multilayer_perceptron*)

6. Backpropagation (*http://en.wikipedia.org/wiki/Backpropagation*)

7. Principles of training multilayer neural network using back propagation (*http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html)*