

Contributions to the Determination of Neural Network Architectures

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

in the field of Artificial Intelligence and Machine Learning
in “Politehnica” University of Timisoara

Drd.Ing. Mohamed Lafif Tej

Scientific Coordinator: Prof. Dr. Ing. Stefan Holban

Abstract

This thesis defines the procedure of identifying optimum neural network design by incorporating data mining. Identify patterns in the training dataset and establish relationships in the training dataset used to train the neural network. Then the information obtained will be used to determine the architecture of the artificial neural network.

There is no evidence supporting any method to determine the optimum ANN architecture. Contemporary approaches are restricted and consume a lot of time. Proven successful approaches offer a solution to given problems but under a given environment. No verifiable theory exists, explaining the design of an ANN.

This scientific research related to artificial intelligence seeks to utilize pattern recognition methods to define the structure of an ANN. It involves clustering methods to group training dataset to identify some common features, that can be grouped depending on given conditions. The results obtained through clustering as far as multilayer ANN structure is concerned.

A regression model is adopted to increase how predictable the projected is, by adopting results from grouping to define the structure of ANN. Depending on hypothesis testing through F-test for regression, the conclusion is arrived at; the neurons in hidden layers and the quantity of hidden layers themselves depend on the factors that are considered through the projected clustering method.

The proposed method reduces the time allocated for network design. This method can make the design simple and easy and possible for a Non-specialist designer. The focus was to develop a fast solution (in terms of learning iterations) maintaining also an acceptable efficiency.

Keywords: Artificial intelligence, machine learning, neural networks architecture, data mining, clustering methods, multi-layer neural network, pattern recognition, regression analysis.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. Ing. Stefan Holban for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my thesis committee for their insightful comments and encouragement, but also for the hard question, which incited me to widen my research from various perspectives.

I wish to thank “Politehnica”, Timisoara University for the opportunity given me to follow the PhD courses.

Last but not the least, I would like to thank my family: my parents and to my brothers and sister for supporting me spiritually throughout writing this thesis and my life in general.

Contents

1.	INTRODUCTION	1
1.1.	PROBLEM STATEMENT	1
1.2.	RESEARCH APPROACH.....	2
1.3.	LIST OF PUBLICATIONS	2
1.4.	OUTLINE OF THE THESIS.....	3
2.	STRUCTURE OF NEURAL NETWORKS.....	5
2.1.	INTRODUCTION.....	5
2.2.	NEURAL NETWORKS	6
2.2.1.	<i>General aspects</i>	6
2.2.2.	<i>Artificial neuron</i>	7
2.2.2.1.	Activation function	8
2.2.2.2.	Cost Function	10
2.3.	MULTI-LAYER PERCEPTRON (MLP)	11
2.3.1.	<i>Network architecture</i>	12
2.3.2.	<i>Performance of number of layers to achieve solution</i>	13
2.3.3.	<i>Performance of number of neurons to achieve solution</i>	14
2.4.	CONTRIBUTIONS TO IDENTIFY THE OPTIMUM MLP ARCHITECTURE	15
2.4.1.	<i>Methods currently used to identify the MLP architecture</i>	15
2.4.1.1.	Pruning algorithms	17
2.4.1.2.	Growing algorithms	19
2.4.1.3.	Evolutionary algorithms.....	21
2.4.1.4.	Reinforcement learning	22
2.4.1.5.	Convolutional Neural Fabrics.....	22
2.4.1.6.	Optimization of MLP architecture Using Heuristic techniques	22
2.4.2.	<i>Ability learning neural network architectures depending on number of hidden layers</i>	23
2.5.	CONCLUSION	24
3.	DETERMINATION OF OPTIMAL NEURAL NETWORK ARCHITECTURE USING CLUSTERING TECHNIQUES	25
3.1.	INTRODUCTION.....	25
3.2.	OPTIMIZATION PROBLEM.....	26
3.3.	THE PROJECTED METHOD FOR OPTIMIZATION	26
3.4.	PROJECTED METHOD STAGES.....	28
3.4.1.	<i>Establishing the learning data</i>	28
3.4.2.	<i>Establishing the quantity of input units</i>	28
3.4.3.	<i>Identifying the quantity of output units</i>	28
3.4.4.	<i>Identifying the quantity of hidden layers</i>	28
3.5.	DATA MINING TECHNIQUES USED	29
3.5.1.	<i>Clustering Algorithm Used</i>	29
3.6.	CREATING AND TRAINING NEURAL NETWORKS USING DATA MINING TECHNIQUES	32
3.6.1.	<i>Presentation of the case studies used to validate theoretical outcomes</i>	33
3.6.1.1.	The analysis of an ECG signal	33
3.6.1.2.	The analysis of a signal from a Sonar	33
3.6.1.3.	Classification of Statlog (Landsat Satellite) Data Set.....	34
3.6.1.4.	Glass Identification Dataset	34
3.6.1.5.	Description of case studies	35
		iii

3.6.2.	<i>Determination of optimum MLP Architecture using clustering algorithms</i>	35
3.6.2.1.	Determination of the optimum quantity of hidden layers for an MLP to classify Statlog (Landsat Satellite) Dataset.....	35
3.6.2.2.	Identifying the optimum quantity of hidden layers for an MLP used to analyze the components of an ECG signal	37
3.6.2.3.	Identifying the optimum quantity of hidden layers for an MLP used in the diagnosis of ECG signal ...	42
3.6.2.4.	Identifying of the optimum quantity of hidden layers for an MLP used for the analysis of a signal from a Sonar	44
3.6.2.5.	Conclusions.....	45
3.7.	EXPERIMENTAL RESULTS.....	46
3.7.1.	<i>Experimental results for clustering techniques</i>	47
3.7.1.1.	Experimental results for Landsat Satellite	47
3.7.1.2.	The experimental results for P-wave	48
3.7.1.3.	Experimental results for QRS complex	50
3.7.1.4.	Experimental results for the T wave signal	51
3.7.1.5.	Experimental results for a diagnostic ECG signal	52
3.7.1.6.	Experimental results for the analysis of a signal from a sonar	53
3.8.	COMPARATIVE STUDY OF CLUSTERING DISTANCE MEASURES TO IDENTIFY THE MLP STRUCTURE.....	54
3.8.1.	<i>Distance Measures</i>	55
3.8.1.1.	Influence of distance measure on the AHC Algorithm results	55
3.8.1.2.	Clustering Distance Measure Techniques Used.....	55
3.8.2.	<i>Trial results for choosing the best distance measure to the clustering method</i>	58
3.8.2.1.	First case study	59
3.8.2.2.	Second Case Study.....	62
3.8.2.3.	Third Case Study	63
3.8.2.4.	Fourth Case Study.....	65
3.8.2.5.	Discussion of Obtained Results.....	67
3.9.	EXPERIMENTAL RESULTS USING “ <i>DETERMINE THE NN ARCHITECTURE</i> ” PLATFORM	68
3.9.1.	<i>Experimental results for Landsat Satellite</i>	69
3.9.2.	<i>The experimental results for P-wave</i>	69
3.9.3.	<i>Experimental results for QRS complex</i>	70
3.9.4.	<i>Experimental results for the T wave signal</i>	70
3.9.5.	<i>Experimental results for a diagnostic ECG signal</i>	71
3.9.6.	<i>Experimental results for the analysis of a signal from a sonar</i>	71
3.10.	DOCUMENTARY RELATED TECHNIQUE USED FOR ESTABLISHING THE ARCHITECTURE OF NEURAL NETWORKS.....	72
3.10.1.	<i>Experimental result for percentage of classification accuracy of datasets</i>	72
3.10.2.	<i>Comparison of experimental result of classification accuracy of datasets</i>	77
3.10.3.	<i>Documentary related technique conclusion</i>	82
3.11.	CONCLUSION.....	82
4.	DETERMINATION OF OPTIMAL NEURAL NETWORK ARCHITECTURE USING REGRESSION TECHNIQUES	83
4.1.	INTRODUCTION.....	83
4.1.1.	<i>Statistical Modeling</i>	83
4.1.2.	<i>The Confirmatory Data Analysis</i>	84
4.1.3.	<i>Designation of Factors</i>	84
4.2.	REGRESSION METHOD USED	85
4.2.1.	<i>Stages of the Method Used</i>	86
4.2.2.	<i>Factors to identify the quantity of hidden layers of an ANN</i>	88
4.2.3.	<i>Factors to identify the quantity of hidden units of ANN</i>	89
4.3.	EXPERIMENTAL RESULTS USING REGRESSION TECHNIQUES	91

4.3.1.	<i>Experimental result for various quantities of hidden layers</i>	92
4.3.2.	<i>Experimental result for various quantities of hidden units</i>	92
4.4.	CONCLUSION	102
5.	IMPORTANCE OF TRAINING DATA ANALYSIS TO IMPROVE GENERALIZATION CAPABILITIES OF NEURAL NETWORK ARCHITECTURES	104
5.1.	INTRODUCTION	104
5.2.	ARTIFICIAL NEURAL NETWORK STRUCTURES AND GENERALIZATION	105
5.3.	NEURAL NETWORK ARCHITECTURE THROUGH GROUPING OF TRAINING DATA	105
5.4.	COMPARISON STUDY TO IDENTIFY THE OPTIMUM ANN STRUCTURE	107
5.5.	STUDIES FOR THE VALIDATION OF HYPOTHETICAL OUTCOMES	108
5.6.	EXPERIMENTAL RESULTS	109
5.6.1.	<i>Comparative Study of Clustering Distance Measures</i>	109
5.6.2.	<i>First Case Study</i>	109
5.6.3.	<i>Second Case Study</i>	113
5.6.4.	<i>Third Case Study</i>	116
5.6.5.	<i>Fourth Case Study</i>	119
5.6.6.	<i>Analysis of Results</i>	122
5.6.7.	<i>Prove of Generalization Capabilities of the projected Method</i>	123
5.7.	CONCLUSION	126
6.	COMPARISON OF THE ANTICIPATED METHOD VS. TRADITIONAL TECHNIQUE	128
6.1.	TRADITIONAL AND PROPOSED METHOD PERCENTAGE OF ACCURACY COMPARISON	130
6.2.	ERROR/EPOCH INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS	131
6.3.	TRADITIONAL AND PROPOSED METHOD TRAINING PERIOD COMPARISON	132
6.4.	CONCLUSION IN THE COMPARISONS	133
7.	CONCLUSION	135
7.1.	SUMMARY OF CONTRIBUTIONS	135
7.1.1.	<i>Clustering techniques to identify the architecture of ANN</i>	135
7.1.2.	<i>Linear regression to identify the quantity of hidden units of an ANN</i>	136
7.1.3.	<i>Generalization Capabilities of ANN Architectures using the projected method</i>	136
7.1.4.	<i>Contribution of the proposed method</i>	137
7.2.	FUTURE WORKS	137
A.	LEARNING NEURAL NETWORK ARCHITECTURES	139
8.	REFERENCES	158

List of Figures:

FIGURE 2-1: THE ARCHITECTURE OF MULTI-LAYERED PERCEPTRON (MLP) WITH NETWORK N:3:3:2	7
FIGURE 2-2: A SIMPLE ARTIFICIAL NEURON.....	8
FIGURE 2-3: RESOLVE THE PORTS LOGIC AND, OR, XOR WITH ONE LAYER	11
FIGURE 2-4: PATTERN SPACE - LINEARLY SEPARABLE	11
FIGURE 2-5: PATTERN SPACE - NONLINEARLY SEPARABLE	12
FIGURE 2-6: GRAPH FOR THE ARCHITECTURE OF MULTI-LAYERED PERCEPTRON (MLP) WITH NETWORK 3:4:2	12
FIGURE 2-7: CONCEPTUALLY: FORWARD ACTIVITY - BACKWARD ERROR.....	13
FIGURE 2-8: PERFORMANCE OF NUMBER OF LAYERS TO ACHIEVE SOLUTION	14
FIGURE 2-9 NEURAL NETWORK PRUNING	18
FIGURE 2-10 THE TOPOLOGIES GENERATED BY TOWER, PYRAMID AND UPSTART ALGORITHMS	20
FIGURE 3-1 THE PROPOSED FRAMEWORK FOR CLUSTERING METHOD.....	30
FIGURE 3-2 NUMBER OF CLUSTERS OBTAINED FOR VARIOUS VALUES OF RD [92].	32
FIGURE 3-3: THE QUANTITY OF CLUSTERS OBTAINED FOR LANDSAT SATELLITE IMAGES WITH A REFERENCE DISTANCE D = 75, 2 GROUPS WERE OBTAINED.	36
FIGURE 3-4: THE RELATIONSHIP BETWEEN THE NUMBER OF GROUPS AND THE VALUE OF RD FOR LANDSAT DATASET	37
FIGURE 3-5: THE QUANTITY OF CLUSTERS OBTAINED FOR P WAVE WITH A REFERENCE DISTANCE D = 11.4, 2 GROUPS OBTAINED.	38
FIGURE 3-6: THE RELATIONSHIP BETWEEN THE NUMBER OF GROUPS AND THE VALUE OF RD FOR THE P-WAVE DATASET	39
FIGURE 3-7: THE QUANTITY OF CLUSTERS OBTAINED FOR QRS COMPLEX WITH A REFERENCE DISTANCE D = 22, WE HAVE ONE GROUP AND 2 ELEMENTS DISPERSED.....	39
FIGURE 3-8: THE RELATIONSHIP BETWEEN THE NUMBER OF GROUPS AND THE VALUE OF RD FOR THE QRS DATASET.	40
FIGURE 3-9: THE QUANTITY OF CLUSTERS OBTAINED FOR T WAVE WITH A REFERENCE DISTANCE D = 11.58, WAS OBTAINED THREE GROUPS.....	41
FIGURE 3-10: THE RELATIONSHIP BETWEEN THE NUMBER OF GROUPS AND THE VALUE OF RD FOR THE T WAVE DATASET	42
FIGURE 3-11: THE QUANTITY OF CLUSTERS OBTAINED FOR ECG SIGNAL WITH A REFERENCE DISTANCE D = 23, WAS OBTAINED 1 GROUP AND 2 OF ELEMENTS THAT COULD NOT BE CLASSIFIED	42
FIGURE 3-12: THE RELATIONSHIP BETWEEN THE NUMBER OF GROUPS AND THE VALUE OF RD FOR ECG DATASET	43
FIGURE 3-13: THE QUANTITY OF CLUSTERS OBTAINED FOR SONAR WITH A REFERENCE DISTANCE D = 2.1, OBTAIN TWO GROUP	44
FIGURE 3-14: THE RELATIONSHIP BETWEEN THE NUMBER OF GROUPS AND THE VALUE OF RD FOR A SONAR DATASET	45
FIGURE 3-15 MLP ARCHITECTURES	46
FIGURE 3-16 THE RELATIVE ERROR FOR THE LANDSAT DATA SET USING VARIOUS QUANTITY OF HIDDEN LAYERS	48
FIGURE 3-17 THE RELATIVE ERROR FOR THE P-WAVE USING VARIOUS QUANTITY OF HIDDEN LAYERS	49
FIGURE 3-18 THE RELATIVE ERROR FOR THE QRS SIGNAL USING VARIOUS QUANTITY OF HIDDEN LAYERS	50
FIGURE 3-19 THE RELATIVE ERROR FOR THE T-WAVE SIGNAL USING VARIOUS QUANTITY OF HIDDEN LAYERS.....	51
FIGURE 3-20 THE RELATIVE ERROR FOR THE ECG SIGNAL USING VARIOUS QUANTITY OF HIDDEN LAYERS.....	52
FIGURE 3-21 THE RELATIVE ERROR FOR THE SONAR SIGNAL USING VARIOUS QUANTITY OF HIDDEN LAYERS.....	53
FIGURE 3-22: EXAMPLE OF SINGLE LINKAGE CLUSTERING	56
FIGURE 3-23: EXAMPLE OF COMPLETE LINKAGE CLUSTERING.....	57
FIGURE 3-24: EXAMPLE OF AVERAGE LINKAGE CLUSTERING	57
FIGURE 3-25: COMPLETE LINKAGE CLUSTERING (EUCLIDEAN) [111]	59
FIGURE 3-26: ERRORS FOR VARIOUS QUANTITY OF HIDDEN LAYERS AND HIDDEN UNITS FOR THE FIRST CASE STUDY [111].....	61
FIGURE 3-27: THE ERROR INDICATOR FOR VARIOUS QUANTITY HIDDEN UNITS [111].....	61
FIGURE 3-28: ERRORS FOR VARIOUS QUANTITY OF HIDDEN LAYERS AND HIDDEN UNITS FOR THE SECOND CASE STUDY	63
FIGURE 3-29: AVERAGE LINKAGE CLUSTERING (MANHATTAN)	64

FIGURE 3-30: ERRORS FOR VARIOUS QUANTITY OF HIDDEN LAYERS AND HIDDEN UNITS FOR THE THIRD CASE STUDY [111].....	64
FIGURE 3-31: SINGLE LINKAGE CLUSTERING (MANHATTAN)	65
FIGURE 3-32: ERRORS FOR VARIOUS QUANTITY OF HIDDEN LAYERS AND HIDDEN UNITS FOR THE FOURTH CASE STUDY [111].....	66
FIGURE 3-33: THE ERROR INDICATOR FOR VARIOUS QUANTITY HIDDEN UNITS [111].....	67
FIGURE 3-34 NEURAL NETWORK ARCHITECTURE.....	68
FIGURE 3-35 CHART OF EPOCH NUMBER AND THE CORRESPONDING PERCENTAGE OF CLASSIFICATION ACCURACY	73
FIGURE 3-36 ACCURACY FOR THE TRAINING OF SONAR DATASET WITH VARIOUS QUANTITY OF HIDDEN LAYERS/UNITS	74
FIGURE 3-37 CHART OF EPOCH NUMBER AND THE CORRESPONDING PERCENTAGE OF CLASSIFICATION ACCURACY	76
FIGURE 3-38 ACCURACY FOR THE TRAINING OF GLASS IDENTIFICATION DATASET WITH VARIOUS QUANTITY OF HIDDEN LAYERS/UNITS	77
FIGURE 3-39 ACCURACY FOR TRAINING OF DATASETS WITH ONE HIDDEN LAYER.....	78
FIGURE 3-40 ACCURACY FOR TRAINING OF DATASETS WITH TWO HIDDEN LAYERS	79
FIGURE 3-41 ACCURACY FOR TRAINING OF DATASETS WITH THREE HIDDEN LAYERS.....	80
FIGURE 3-42 ACCURACY FOR TRAINING OF DATASETS WITH FOUR HIDDEN LAYERS.....	81
FIGURE 4-1 PROPOSED FRAMEWORK USING REGRESSION METHODS	87
FIGURE 4-2 COEFFICIENTS DIAGRAM [123].....	89
FIGURE 4-3: COEFFICIENTS DIAGRAM [123].....	91
FIGURE 4-4: ERRORS INDICATOR FOR THE LANDSAT TRAINING DATA	94
FIGURE 4-5: ERRORS INDICATOR FOR THE P-WAVE SIGNAL TRAINING DATA	95
FIGURE 4-6: ERRORS INDICATOR FOR THE QRS SIGNAL TRAINING DATA	97
FIGURE 4-7: ERRORS INDICATOR FOR THE T WAVE SIGNAL TRAINING DATA	98
FIGURE 4-8: ERRORS INDICATOR FOR THE ECG SIGNAL TRAINING DATA	100
FIGURE 4-9: ERRORS INDICATOR FOR THE SONAR SIGNAL TRAINING DATA.....	101
FIGURE 5-1 FRAMEWORK FOR THE PROPOSED METHOD	106
FIGURE 5-2 ORANGE CANVAS OPEN SOURCE MACHINE LEARNING AND DATA VISUALIZATION.	109
FIGURE 5-3 DENDROGRAM FOR CLUSTERING IONOSPHERE DATASET, TWO CLUSTERS APPEARED	110
FIGURE 5-4 PERCENTAGE OF CLASSIFICATION ACCURACY FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH TWO HIDDEN LAYERS.	111
FIGURE 5-5 ERROR INDICATOR FOR VARIOUS QUANTITY OF HIDDEN UNITS WITH TWO HIDDEN LAYERS.....	112
FIGURE 5-6 TRAINING TIME FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH TWO HIDDEN LAYERS.	112
FIGURE 5-7 DENDROGRAM FOR CLUSTERING IONOSPHERE DATASET, ONE CLUSTER APPEARED.....	113
FIGURE 5-8 PERCENTAGE OF CLASSIFICATION ACCURACY FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH ONE HIDDEN LAYER.....	114
FIGURE 5-9 ERROR INDICATOR FOR VARIOUS QUANTITY OF HIDDEN UNITS WITH ONE HIDDEN LAYER	115
FIGURE 5-10 TRAINING TIME FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH ONE HIDDEN LAYER.	115
FIGURE 5-11 AVERAGE LINKAGE (MANHATTAN) THREE CLUSTERS APPEARED.....	116
FIGURE 5-12 PERCENTAGE OF CLASSIFICATION ACCURACY FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH THREE HIDDEN LAYERS.	117
FIGURE 5-13 ERROR INDICATOR FOR VARIOUS QUANTITY OF HIDDEN UNITS WITH THREE HIDDEN LAYERS.....	118
FIGURE 5-14 TRAINING TIME FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH THREE HIDDEN LAYERS.	118
FIGURE 5-15 AVERAGE LINKAGE (EUCLIDEAN).	119
FIGURE 5-16 PERCENTAGE OF CLASSIFICATION ACCURACY FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH FOUR HIDDEN LAYERS.	120
FIGURE 5-17 ERROR INDICATOR FOR VARIOUS QUANTITY OF HIDDEN UNITS WITH FOUR HIDDEN LAYERS	121
FIGURE 5-18 TRAINING TIME FOR DIFFERENT NUMBERS OF HIDDEN NEURONS WITH FOUR HIDDEN LAYERS.....	121
FIGURE 5-19 PERCENTAGE OF CLASSIFICATION ACCURACY FOR IONOSPHERE DATA SET.	122
FIGURE 5-20 DENDROGRAM FOR DATASETS.....	124
FIGURE 6-1 PERCENTAGE OF ACCURACY INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS	131
FIGURE 6-2 ERROR/EPOCH INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS	132

FIGURE 6-3 TRAINING PERIOD INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS.....	133
FIGURE 7-1 THE DIFFERENCE IN THE INFORMATION FLOW BETWEEN AN RNN AND A FEED-FORWARD NEURAL NETWORK	141
FIGURE 7-2 THE PLOT OF THE COST FUNCTION USING MSE FUNCTION	146
FIGURE 7-3 THE PLOT OF THE COST FUNCTION USING MAE FUNCTION.....	147
FIGURE 7-4 THE PLOT OF THE COST FUNCTION USING HUBER LOSS FUNCTION WITH THREE VALUES OF δ (DELTA) EQUAL TO 0.1, 1 AND 10	149
FIGURE 7-5 THE PLOT OF THE LOG HYPERBOLIC COSINE (LOG-COSH) LOSS FUNCTION	150
FIGURE 7-6 THE PLOT OF THE COST FUNCTION USING QUANTILE LOSS WITH VALUES OF QUANTILE γ EQUAL TO 0.25, 0.5 AND 0.75.....	151

List of Tables:

TABLE 2-1: EXAMPLES OF NEURON ACTIVATION FUNCTIONS	9
TABLE 2-2: ABILITY LEARNING NEURAL NETWORK ARCHITECTURES DEPENDING ON NUMBER OF HIDDEN LAYERS	24
TABLE 3-1: DECISION CLASSES, THE NUMBER IS A CODE FOR THE FOLLOWING CLASSES	34
TABLE 3-2: DATA OBTAINED WITH THE PROGRAM OF GROUPING FOR THE LANDSAT SATELLITE IMAGES DATASET	36
TABLE 3-3: DATA OBTAINED WITH THE PROGRAM OF GROUPING FOR THE P WAVE OF THE ECG SIGNAL	38
TABLE 3-4: THE OBTAINED WITH THE PROGRAM OF CLUSTERING THE QRS COMPLEX, OF THE ECG SIGNAL	40
TABLE 3-5: DATA OBTAINED WITH THE PROGRAM OF GROUPING FOR THE T WAVE, OF THE ECG SIGNAL	41
TABLE 3-6: DATA FOR THE DIAGNOSIS OF AN ECG SIGNAL	43
TABLE 3-7: DATA OBTAINED WITH THE PROGRAM OF GROUPING FOR THE ANALYSIS SIGNAL FROM A SONAR	44
TABLE 3-8: COMPARISON BETWEEN THE BEST QUANTITY OF HIDDEN LAYERS OBTAINED BY DIFFERENT MLP ARCHITECTURES AND THE NUMBER OF CLUSTERS OBTAINED BY CLUSTERING METHOD FOR ALL DATASETS	54
TABLE 3-9 FIRST CASE STUDY	59
TABLE 3-10: THE ACCURACY AND ERROR VALUES [111]	60
TABLE 3-11 SECOND CASE STUDY	62
TABLE 3-12: THE ACCURACY AND ERROR VALUES [111]	62
TABLE 3-13 THIRD CASE STUDY	63
TABLE 3-14 FOURTH CASE STUDY	65
TABLE 3-15: THE ACCURACY AND ERROR VALUES [111]	66
TABLE 3-16 ACCURACY AND ERROR/EPOCH VALUES BASED ON FORMULAS (2.4)-(2.9) FOR LANDSAT SATELLITE DATASET	69
TABLE 3-17 ACCURACY AND ERROR/EPOCH VALUES BASED ON FORMULAS (2.4)-(2.9) FOR P-WAVE DATASET	69
TABLE 3-18 ACCURACY AND ERROR/EPOCH VALUES BASED ON FORMULAS (2.4)-(2.9) FOR QRS DATASET	70
TABLE 3-19 ACCURACY AND ERROR/EPOCH VALUES BASED ON FORMULAS (2.4)-(2.9) FOR THE T-WAVE DATASET	70
TABLE 3-20 ACCURACY AND ERROR/EPOCH VALUES BASED ON FORMULAS (2.4)-(2.9) FOR ECG DATASET	71
TABLE 3-21 ACCURACY AND ERROR/EPOCH VALUES BASED ON FORMULAS (2.4)-(2.9) FOR SONAR DATASET	71
TABLE 3-22 CLASSIFICATION ACCURACY FOR THE TRAINING OF SONAR DATASET WITH A DIFFERENT NUMBER OF EPOCHS	72
TABLE 3-23 ACCURACY FOR THE TRAINING OF SONAR DATASET WITH A VARIOUS QUANTITY OF HIDDEN LAYERS/UNITS	73
TABLE 3-24 CLASSIFICATION ACCURACY FOR THE TRAINING OF GLASS IDENTIFICATION DATASET WITH DIFFERENT NUMBER OF EPOCHS	75
TABLE 3-25 ACCURACY FOR THE TRAINING OF GLASS IDENTIFICATION DATASET WITH VARIOUS QUANTITY OF HIDDEN LAYERS/UNITS	76
TABLE 3-26 THE PERCENTAGE OF CLASSIFICATION ACCURACY FOR TRAINING OF DATASETS WITH ONE HIDDEN LAYER	78
TABLE 3-27 THE PERCENTAGE OF CLASSIFICATION ACCURACY FOR TRAINING OF DATASETS WITH TWO HIDDEN LAYERS	79
TABLE 3-28 THE PERCENTAGE OF CLASSIFICATION ACCURACY FOR TRAINING OF DATASETS WITH THREE HIDDEN LAYERS	80
TABLE 3-29 THE PERCENTAGE OF CLASSIFICATION ACCURACY FOR TRAINING OF DATASETS WITH FOUR HIDDEN LAYERS	81
TABLE 4-1 SELECTED PARAMETERS	85
TABLE 4-2 SELECTED FACTORS TO IDENTIFY THE HIDDEN LAYERS QUANTITY	88
TABLE 4-3: THE INDEPENDENT VARIABLES [123]	88
TABLE 4-4 SELECTED FACTORS TO IDENTIFY THE HIDDEN LAYERS QUANTITY	90
TABLE 4-5: THE INDEPENDENT VARIABLES [123]	90
TABLE 4-6: THE NUMBER OF LAYERS OF NEURAL NETWORKS USING REGRESSION TECHNIQUES FOR ALL DATASETS	92
TABLE 5-1 THE VALUE OF CLASSIFICATION ACCURACY AND ERRORS FOR FIRST CASE STUDY (TWO HIDDEN LAYERS)	111

TABLE 5-2 THE VALUE OF CLASSIFICATION ACCURACY AND ERRORS FOR SECOND CASE STUDY (ONE HIDDEN LAYER)	114
TABLE 5-3 THE VALUE OF CLASSIFICATION ACCURACY AND ERRORS FOR THIRD CASE STUDY (THREE HIDDEN LAYERS)	117
TABLE 5-4 THE VALUE OF CLASSIFICATION ACCURACY AND ERRORS FOR FOURTH CASE STUDY (FOUR HIDDEN LAYERS)	120
TABLE 5-5 QUANTITY OF HIDDEN LAYERS BASED ON THE PROJECTED METHOD	125
TABLE 5-6 THE VALUES OF CLASSIFICATION ACCURACY	125
TABLE 5-7 ERRORS/EPOCH	126
TABLE 6-1 CONFIGURATION OF THE ANN	129
TABLE 6-2 QUANTITY OF HIDDEN NODES THROUGH THE TRADITIONAL METHODS	129
TABLE 6-3 QUANTITY OF HIDDEN NODES THROUGH THE PROJECTED METHOD	130
TABLE 6-4 PERCENTAGE OF ACCURACY INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS	130
TABLE 6-5 ERROR/EPOCH INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS	132
TABLE 6-6 TRAINING PERIOD INDICATOR IN BOTH PROJECTED AND TRADITIONAL METHODS (SECONDS)	133

Chapter I

1. Introduction

Researchers concentrated on the concept that computers can perform different tasks without programming. [1], AI researchers [2] wanted to confirm if machines can learn from a given set of instructions. Computer science has a major evolving section called machine learning- determined to ensure computers perform tasks without programming. Machine learning has its roots in computer learning theory [3] and pattern recognition.

Some areas of study encompassed under the canon of biologically inspired computing [4], and their biological counterparts. Take for example genetic algorithms inspired from evolution. Biodegradability prediction inspired by biodegradation. Cellular automata inspired by life. Emergent systems inspired by ants, termites, bees, and wasps. Neural networks inspired by the brain. Artificial life inspired by life. Artificial immune systems inspired by the immune system. Rendering (computer graphics) inspired by patterning and rendering of animal skins, bird feathers, mollusk shells and bacterial colonies.

Artificial Neural Networks is Biologically Inspired [5]. A biologically inspired programming paradigm enables a computer to learn from observational data. Current researchers focused on the biological knowledge of how the brain learns, recently included in majority prevailing computing, followed by improvements in hardware and modulation in the software models. More than two decades ago, neural networks were widely seen as the next generation of computing, one that would finally allow computers to think for themselves.

Achievement of optimal neural network architecture is necessary due to neural networks applications in medical imaging (localization of tumors and other pathologies, measuring tissue volume, Computer-assisted surgery, diagnosis, treatment planning), the location of objects in satellite images, facial recognition, iris recognition, fingerprint recognition, traffic control systems and more [6]. These applications recommended as the current one in which have invested the effort to satisfactory results as close to the ideal.

1.1.Problem Statement

Here, we will show that there is no analytical method to determine the numbers of hidden layers [7] together with hidden node numbers inside layers [8] to multiple layer ANN [9]. There are series of Neural Network architectures available but only offers solutions in given situations,

but also successfully solves a few applications under specified conditions. The neural network design is a major area of concern for researchers and lacks theoretical viewpoints.

Identifying neurons and hidden layers number is a multifaceted task and key point in the formation of the ANN - that has no supporting theories. To date, there is still no verifiable theory that can be adopted to obtain the extent of the architecture of the ANN considering the intricacy of the subject under study [10] [11] [12].

1.2. Research Approach

In this thesis, we present research under a subdomain of current scientific research in artificial intelligence namely the determination of optimal neural network architecture. This subdomain related to Pattern Recognition and Data Mining. All the current research directions highly specialized in neural network, forming part of the broader subdomain Artificial Intelligence.

Notably, the importance that neural networks currently represent in the data classification. I also noted that this is the only technique that allows generalizations, based on a set of data designed for analysis [13] [14] [15] [16] [17]. In other words, the results provided by the neural network is closest to the concept of inference in which we can get conclusions from the analysis of a given context.

I intend in this respect to use Data Mining techniques to analyze the data to be processed by the ANN. This method will work unsupervised to analyze the training data through clustering techniques and to correlate the quantity of groups obtained with the optimum quantity of hidden layers for an MLP. In determining the quantity of hidden units we will use the results by clustering training forms from the analyzed database on the basis of a reference distance (RD).

To come up with a neuron quantity in hidden layers, a linear regression method incorporating the parameters obtained from grouping given data used to train the neural network is developed. The approach decreases the build time of the optical ANN since it is not being supervised.

1.3. List of Publications

Tej, Mohamed Lafif, and Stefan Holban. "Determining Multi-layer Perceptron Structure Using Clustering Techniques." *International Journal of Artificial Intelligence* 17, no. 1 (2019): pp. 139-166.

Tej, Mohamed Lafif, and Stefan Holban. "Determining neural network architecture using data mining techniques." In *2018 International Conference on Development and Application Systems (DAS)*, pp. 156-163. IEEE, 2018.

Tej, Mohamed Lafif, and Stefan Holban. "Comparative Study of Clustering Distance Measures to Determine Neural Network Architectures." In 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 000189-000194. IEEE, 2018.

Tej, Mohamed Lafif, and Stefan Holban. "Determining optimal neural network architecture using regression methods." In 2018 International Conference on Development and Application Systems (DAS), pp. 180-189. IEEE, 2018.

Tej, Mohamed Lafif, and Stefan Holban. "Determining Optimal Multi-Layer Perceptron Structure Using Linear Regression." In International Conference on Business Information Systems, pp. 232-246. Springer, Cham, 2019.

1.4. Outline of the thesis

The rest of this thesis is organized as follows:

- **Chapter 2** contains an analysis of the structures of Neural Networks and explanation of the effect of the architecture of neural network on the ability of learning of network. In addition, a presentation of the methods currently used to determine the architecture of the neural network.
- **Chapter 3** describes the method used to determine the optimal Neural Network Architecture using clustering techniques by presenting the stages, which we pass through to determine the architecture of the neural network. In addition, a discussion about clustering techniques used, the role of clustering distance measures in determining the optimal neural network architecture and a comparative study of clustering distance measures used to Determine Neural Network Architectures.
- **Chapter 4** describes another data mining technique used to determine the optimal Neural Network Architecture by means of regression methods. The results obtained from clustering the training data are found to be useful to develop a Multiple Linear Regression model in accordance with the purpose of determining the number of hidden layers and the number of neurons on each layer for a multilayer neural network. A discussion on factors selected and the influence of each factor on the number of hidden layers and the number of the hidden neuron will be presented in this chapter.
- **Chapter 5** describes the importance of training data analysis to improve generalization capabilities of the Neural Network Architectures. In this chapter, we discuss the generalization capabilities of neural network architectures through the analysis of training data. There is a solid relationship between the generalization performance of artificial neural networks and their structure. It is shown that the generalization performance of neural networks is affected by the structure of the network.

- **Chapter 6** describes a comparison of the proposed method with the most currently used methods based on the percentage of accuracy, Error/epoch and training time. This chapter shows how the proposed method performs well for the different type of datasets and how it's adapted to the complexity of the training data to provide the best results regardless the size and type of dataset in contrast to other methods.
- **Chapter 7** presents a summary of the work presented in this thesis and concludes this work.

Chapter II

2. Structure of Neural Networks

The most important theoretical problem presented in this thesis is the correlation between the structure of the neural network and the learning ability of the network. To solve a complex problem using the neural network it needs a complex network structure represented by the quantity of hidden layers/units. The size of the network affects the learning capability of the neural network, which makes adding more hidden layers and units in these layers essential when the complexity of the problem to be solved increases.

Theoretically, it can be proved that a neural network with one advantage of using multiple hidden layers is improving the predictive ability of the network. The problem is how many hidden layers we need based on the level of complexity of the problem to be solved by the neural network without exceeding the required number of hidden layers to avoid the decrease of the accuracy in the test set. Increasing the number of hidden layers much more than required will cause the network to overfit to the training set that means it will learn the training data, but it won't be able to generalize to new unseen data.

Quantity of hidden neurons signifies a key area of concern in neural network completion. Prominent methods include; "Trial and Error, evolutionary algorithms, exhaustive search, and Growing and Pruning algorithms". they consume a lot of time and have some limitations.

2.1.Introduction

In this chapter, the structures of Neural Networks will be analyzed and an explanation of the effect of the ANN architecture on the ability of learning of network. In addition, the structure of the ANN depends on many parameters represented by the Cost function, Activation function, and Hyper-parameters. The selection of the suitable cost function and activation function appropriate to the considered problem is still the aim of the current research [18] [19] [20].

The cost function and activation function affect the ability to learn for a neural network. It is not possible to define a general cost function or a general activation function, which work with all type problems. A comparative study to determine the optimal activation function for certain neural networks is necessary to determine the appropriate activation function for the learning algorithm of the neural network. As well as the cost function, also need a comparative study to determine the appropriate function for the learning algorithm of the neural network. The cost function is used in machine learning to improve the performance of the model.

Neural networks depend on many Hyper-parameters, which are related to the structure of the neural network and to the learning algorithm of the network. Hyper-parameters related to the

structure of neural network are the variables, which determines the network structure such as the number of hidden layers and neurons, network weight initialization and activation function. Hyper-parameters related to the learning algorithm of the network are the variables, which determine how the network is trained such as Learning Rate, Momentum, Number of epochs, Batch size and so on. It is necessary to tune the hyper-parameters during training see Appendix A.

A large number of Neural Network Architectures was created for different issues each one was designed based on certain specifications to achieve different goals. These specifications are imposed by the problem to be solved by the neural network. In this chapter, we will mention the most efficient architectures see Appendix A.1.

We currently lack a practicable principle to define the size of the ANN, putting into consideration the problem to be solved. Current methods are not consistent and may only perform in some settings.

2.2. Neural networks

ANN is an integral topic in Artificial intelligence [21]. The ANN works like a human brain. The human brain is composed of approximately 100 billion neurons. Until now, artificial neural networks were unable to reach the computational power of the brain.

In (1943) Warren McCulloch and Walter Pitts create a computational model for neural networks based on mathematics and algorithms called threshold logic. This model opens the door for more neural network research.

The idea of multilayer neural networks has begun since the sixties. The emergence of the Back Propagation (BP) algorithm in the mid-1980s, make training of multilayer perceptron (MLP) possible, where it has become widely used.

Since the mid-80s, there have been contributions made to the theory and applications of neural networks, most of them are concentrated in multilayer perceptron MLP.

2.2.1. General aspects

Neural networks typically consist of multiple layers, and the signal path traverses from the first input layer to the output layer, which is the last layer of neural units. A multilayer neural network consists of an input layer of source neurons, at least one middle or hidden layer of computational neurons, and an output layer of computational neurons. The input signals propagated in a forward direction on a layer-by-layer basis. Multilayer perceptron training involves both forward and backward information flow. An example of a simple Neural Network illustrated in the Figure 2-1.

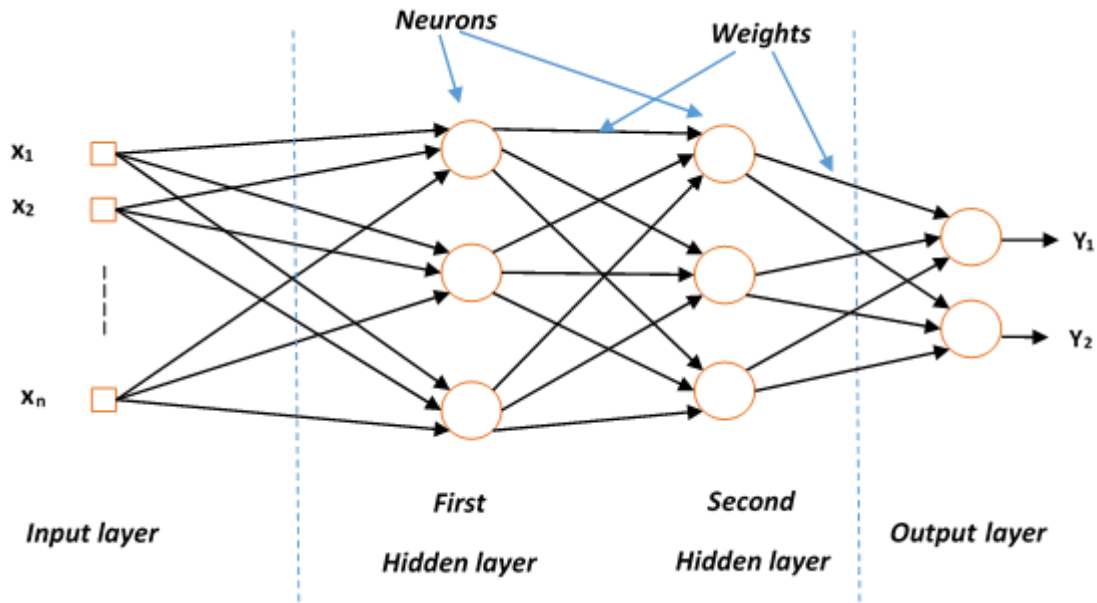


Figure 2-1: The architecture of multi-layered perceptron (MLP) with network $n:3:3:2$

All neurons in one of the layers linked with all neurons in the following layer, for a feedforward network, all data in the input layer streams to the output layer. The structure of the network presented in Figure 2-1 is $n:3:3:2$. A bias connection assigned to all units in hidden layers and to all units in the output layer. Every unit is a Perceptron.

All features of the training set assigned to the input layer; the quantity of input units equal to the quantity of features of the training dataset. Input layer neurons connected to the neurons of the first layer of hidden layers.

A neural network can contain one or more hidden layers, one hidden layer can solve many problems and theoretically, there is no reason to use more than two hidden layers. It is not necessary for the quantity of hidden units to be equal to the quantity of units in the input layer. The quantity of hidden units is defined using experimental methods. Hidden layer treats the inputs and communicates with all output neurons.

The final outputs generated by the output layer. The quantity of output units equal to the quantity of required outputs. The output could be linear or sigmoid [22].

2.2.2. Artificial neuron

Neural Networks consists of many artificial neurons. Each input into the neuron has its own weight $w_1, w_2, w_3... w_n$. Weight is simply a floating-point number and these we adjust these weights when we eventually come to train the network.

A neuron can have any number of inputs from 1 to n , where n is the total number of inputs. The inputs represented therefore as $x_1, x_2, x_3 \dots x_n$, and the corresponding weights for the inputs as $w_1, w_2, w_3 \dots w_n$. Output $y = x_1w_1 + x_2w_2 + x_3w_3 \dots + x_nw_n$.

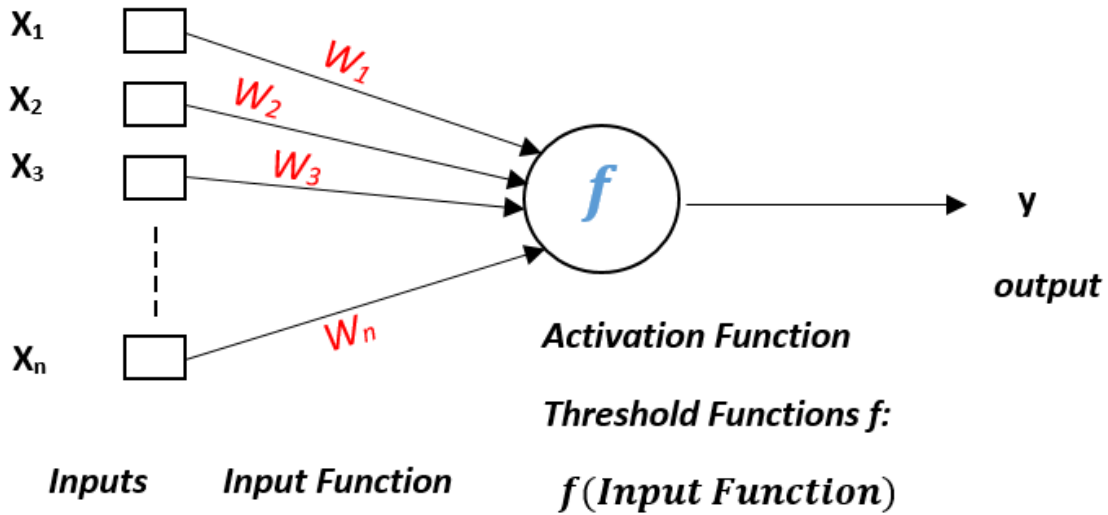


Figure 2-2: A simple artificial neuron

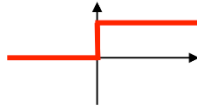
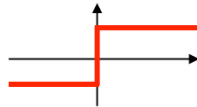

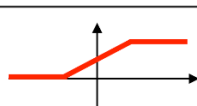
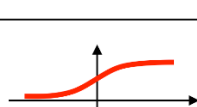

Where X_i input, Y_i output, n is the number of inputs to the Perceptron. w_i weight: where w is a vector of real-valued weights. Input function is:

$$\sum_{i=1}^n W_i * X_i \tag{2.1}$$

2.2.2.1. Activation function

Activation function use threshold Θ to evaluate and validate the weight. Transforms neuron's input into output [22]. Table 2-1 present the most common activation functions [23]:

Table 2-1: Examples of neuron activation functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer NN	

Predicting the best activation function is usually impossible. The selection depends on trial and error to define the suitable activation function for the considered problem. Training the network with a kind of activation function then evaluating its performance on a validation set until we can compare it with other activation functions.

However, it is possible to approach a better choice depending upon the properties of the problem. Some activation functions characteristics can be relied upon but are not perfect such as sigmoid function generally work better in the case of classification, for the vanishing gradient problem it is better to avoid sigmoid and Tanh functions.

The activation function ReLU work good with the majority of cases and can be taken as a general activation function and represents the best choice for the case of dead neurons, the ReLU function can represent a best starting point and then move over the other function.

2.2.2.2. Cost Function

There are numerous Machine Learning algorithms used to optimize the architecture of the ANN. The purpose of optimization is to minimize the cost function [24] in the training data. Cost function or loss function used to calculate the error obtained from the difference between the actual output and the predicted output.

$$J(w) = P - \hat{P} \quad (2.2)$$

Neural networks use back propagation to update neurons with new weights calculated using cost function in such a way the error is minimized. The most commonly used method to define the minimum point of the cost function is Gradient Descent. A cost function is a measure of the performance of a neural network architecture to predict the expected outcome.

To optimize the cost function using Gradient Descent:

$$W^{(k+t)} = W^{(k)} - \frac{\partial}{\partial W^{(k)}} J(W) \quad (2.3)$$

The cost function like activation function has an important role to obtain accurate results when training model. To obtain the optimum results for different problems we have to compare different type of cost function. We cannot define a general cost function, which works with all type of data. A number of factors affect the selection of the suitable cost function for a considered problem.

The cost function falls under several categories such as classification cost functions, regression cost function and so on.

Classification cost function algorithms:

- Binary Cross Entropy
- Margin Classifier
- Soft Margin Classifier
- Negative Log Likelihood

Regressive cost function algorithms:

- Mean Absolute Error
- Huber Loss, Smooth Absolute Error

It is recommended to determine the suitable combinations of the cost function with the activation function. Depending on the type of the considered problem such as Binary classification, Multiclass classification, and Regression. Some experimental test prove that cross entropy is suitable with probabilistic output, it is preferable to use Mean squared error for regression and the combination of sigmoid function and Mean squared error is not recommended.

2.3. Multi-Layer Perceptron (MLP)

Multilayer perceptron (MLP) are multilayer feedforward networks with one or more hidden layers [25] and continuously differentiable nonlinear activation functions.

Single layer perceptron can only deal with linear problems, Multi-Layer Perceptron MLP used to describe any general feedforward (no recurrent connections) network. We need extra layer until we can resolve the problem, which cannot be resolved with one single layer, for example, we cannot resolve the port logic XOR with one layer for this we need another extra layer since it is not linearly separable.

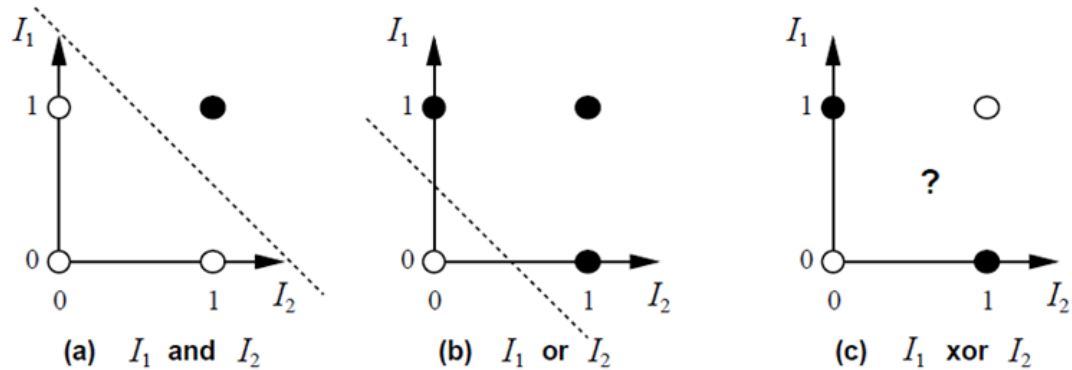


Figure 2-3: resolve the ports logic AND, OR, XOR with one layer

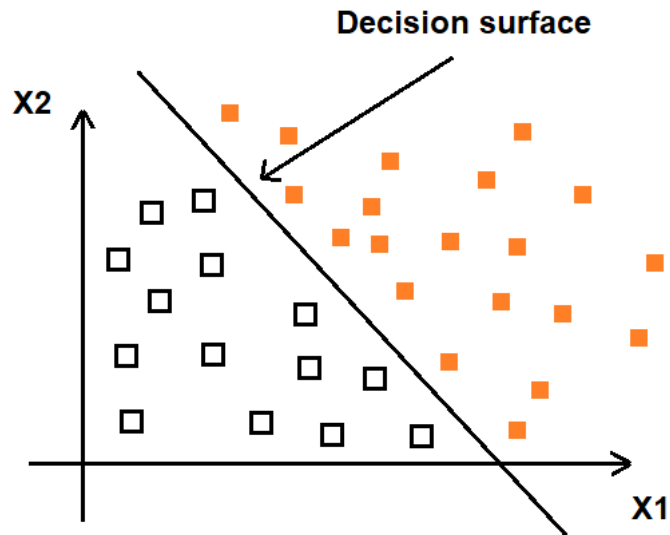


Figure 2-4: Pattern space - linearly separable

If a problem is linearly inseparable pattern, Multi-layer perceptron found as a solution to represent nonlinearly separable functions.

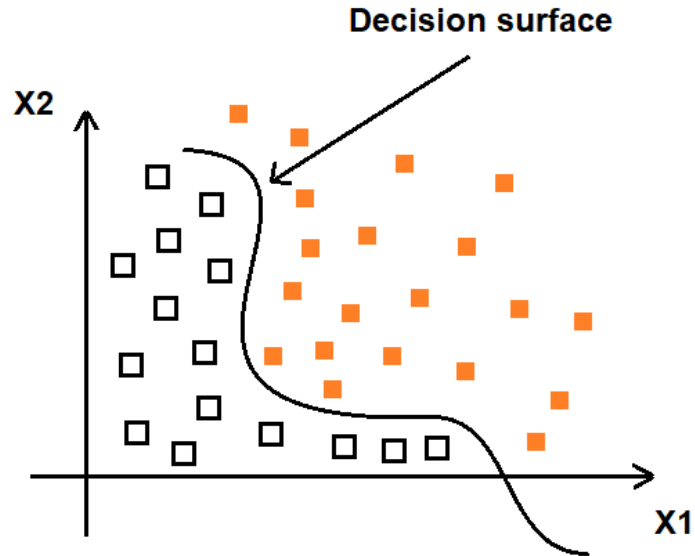
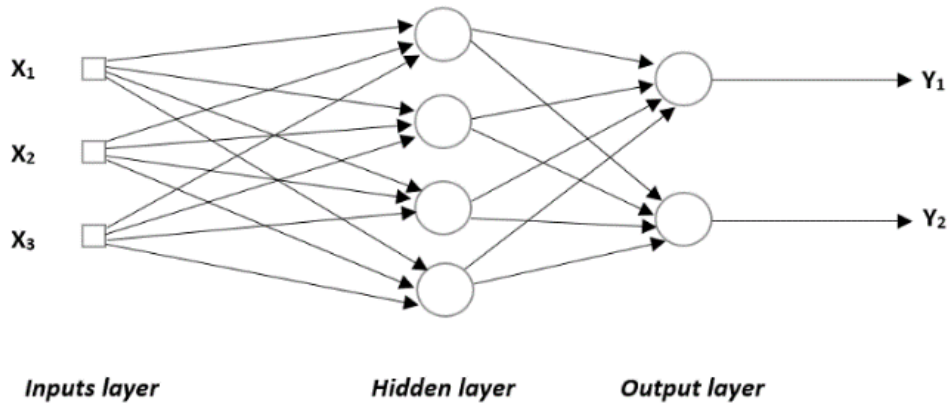


Figure 2-5: Pattern space - nonlinearly separable

2.3.1. Network architecture



$$y_i = f \left(\sum_{j=1}^m w_{ij} * X_j + b_i \right)$$

Figure 2-6: Graph for The architecture of multi-layered perceptron (MLP) with network 3:4:2

The Backpropagation learning algorithm is a solution to the credit assignment problem in MLP. Rumelhart, Hinton, and Williams (1986) (though actually invented earlier in a Ph.D. thesis relating to economics).

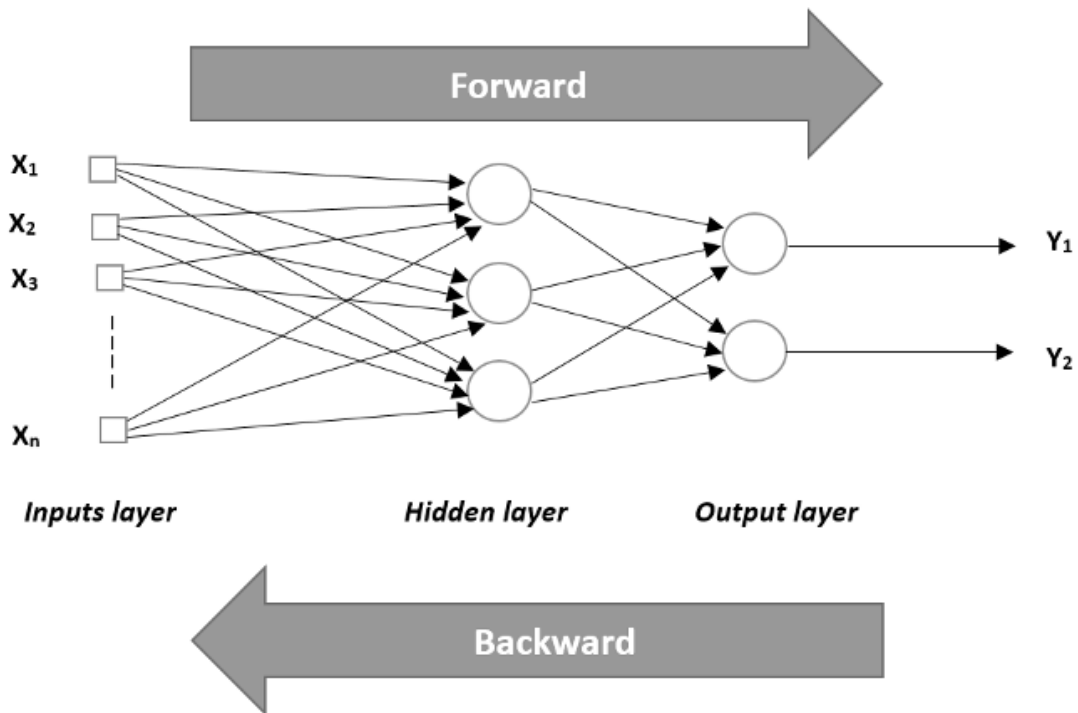


Figure 2-7: Conceptually: Forward Activity - Backward Error

2.3.2. Performance of number of layers to achieve solution

The multi-layered perceptron (MLP) [26] can represent any function with a single layer. Therefore, an MLP with 3 hidden layers can solve any problem. However, there is no available analytical method for determining the optimal number of hidden neurons for a multi-layer neural network.

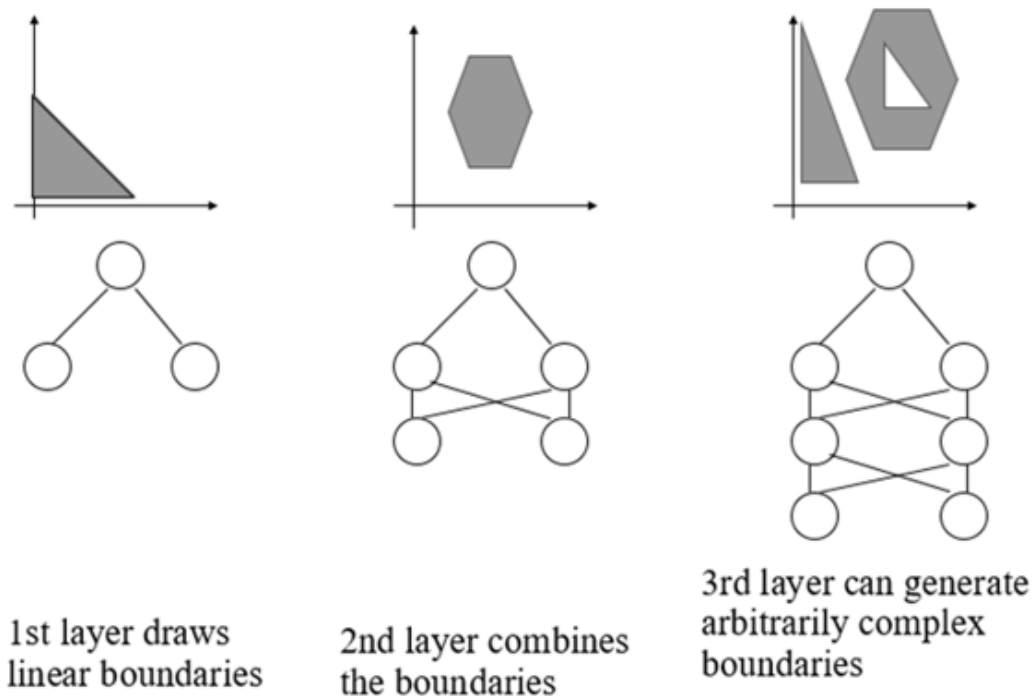


Figure 2-8: Performance of number of layers to achieve solution

If the data is linearly separable, the network does not need a hidden layer [27]. An MLP composed of two hidden layers can represent function sufficient to solve any problem [28]. For MLPs with any continuous nonlinear hidden-layer activation functions, need one hidden layer with an arbitrarily large number of hidden neurons suffices for the "universal approximation" property [29] [30] [31] [32]. Until now, there is no theory to identify the required quantity of hidden units to approximate any given function.

2.3.3. Performance of number of neurons to achieve solution

The ANN can be affected by hidden neurons. By incorporating more hidden neurons to handle a problem. This can be solved by minimizing hidden neurons. When it is minimized, the connection will not be that powerful to satisfy the desired necessities. The network can be optimized by lowering the percentage of hidden neurons. Lowering hidden neurons comes with a reduction in the capability of the neural networks.

Several factors affect the quantity of hidden units. Among them the quantity of input and output units, the quantity of training forms, the MLP architecture that takes into account the configuration and interconnection layers, the complexity of the function or classification to be learned, the activation function of hidden neurons, the training algorithm, the cost function used and the

amount of noise in the targets. In most cases, it is not possible to identify the optimum quantity of hidden layers without training several networks and evaluating the generalization error for each network. For a few hidden neurons, the training errors and the generalization errors become too high due to under-fitting. For a large number of hidden neurons, the training errors decrease but the network keep the high value of generalization errors due to overfitting [33].

Many researchers use Rules of Thumb to identify the structure of an MLP. Among them

- The rule, which takes the number of hidden layers between the number of input neurons and the number of output neurons [34].
- For an MLP with a single hidden layer, do not need a quantity of hidden units more than twice of the quantity of input units [35].
- Another rule of thumb impose that the hidden layer cannot take a quantity of hidden units more than twice of the quantity of units of input layers [36].
- Another rule of thumb impose that impose to specify as many hidden neurons needed to capture 70-90% of the variance of input data forms [37].

The rules of thumb presented above do not have a good performance because they ignore the number of training cases, the noise, and the complexity of the function.

2.4. Contributions to Identify the Optimum MLP Architecture

Valid general-purpose theories to identify the structure of an MLP do not exist. A viable theory to determine the structure of the network based on the complexity of the considered problem to be solved is not reached. Given these circumstances, the design of the neural network is determined empirically and based on exhaustive simulations. The designer of a neural network must have the appropriate qualifications and good experience. The neural network designs that are achieved are specially designed for specific issues and achieves success only for certain conditions.

Now, there are no well-developed formal methods for a priori determination of the optimal architecture of neural networks, the choice being made on the basis of experience designer and exhaustive simulations.

2.4.1. Methods currently used to identify the MLP architecture

In this respect, I should mention the currently used methods to identify the architecture of an MLP include the method, which based on the design of multiple architectures of neural networks [38] [39]. The selected neural network architectures will be trained on a common data set until we obtain the best classification accuracy or the acceptable values of error/epoch. After evaluating the results of all ANN architectures, the selection of the best architecture will be based on the lowest quantity of hidden layers and units, optionally they will take the training time into consideration [40]. This method has several disadvantages such as it is time-consuming, and the results depend on the initially selected architecture.

Sometimes it is more convenient in terms of design an MLP using numerous MLP with simpler and interconnected architectures, instead of just one with more design (analysis of problems by decomposing them into simpler problems, if possible).

Simplistically to the quantity of input units will be equivalent to the quantity of features that compose the entry forms, and for the quantity of output units, the quantity of units must allow fair representation (unambiguous) which response to the neural network [41]. Overall, the number of outputs is directly imposed on the application, in case of problems of classification are required a quantity of output units equivalent to the quantity of distinct classes they need to recognize the MLP, or equivalent to the size of the output vector.

For the quantity of hidden units, there is no generally agreed method to find out this quantity is usually determined by experimental or heuristic rules [9]. Hidden units quantity must be enough to be able to generate enough complex configuration of decision regions in order to solve the problem taken into account. A large quantity of hidden units causes an increase in the quantity of connections between units resulting in a possible risk failure by the wrong estimation of the perfect values of weights using the training forms available, especially if we have few training forms and so the network can generate noise. A few numbers of hidden units can cause neural network failure to learn all information in the learning data also decreases fault tolerance by decreasing network redundancy. Many methods to calculate the quantity of hidden units are defined in research papers [42] among them:

$$\text{Number of neurons in the hidden layers} = 1/2(\text{input} + \text{output}) \quad (2.4)$$

$$\text{Number of neurons in the hidden layers} = \text{SQRT}(1/2(\text{input} + \text{output})) \quad (2.5)$$

In other studies [43] [44], this formula is proposed for calculating the quantity of hidden layers formula:

$$\text{Number of neurons in the hidden layers} = (\text{input} + \text{output}) * 2/3 \quad (2.6)$$

$$\text{Number of neurons of the first hidden layer} = \text{training forms} / 10(\text{input} + \text{out}) \quad (2.7)$$

$$\text{The number of neurons in the hidden layer} = (\text{training forms} - \text{output}) / \text{input} + \text{output} + 1 \quad (2.8)$$

If we have more than one hidden layer, they usually (two or three layers,) have half the quantity of units from the precedent hidden layer. Usually, the optimum MLP architecture cannot be identified by taking into account only the quantity of inputs and outputs, architecture is critically

dependent on the quantity of data in the learning data set and the complexity of the function that it needs to learn. NeuroShell program [45] for simulation and training MLP calculate the quantity of hidden units using this formula:

$$\text{Number of neurons in the hidden layers} = 1/2(\text{input} + \text{out}) + \text{SQRT}(\text{training forms}) \quad (2.9)$$

If we have more than one hidden layer, the quantity of hidden units calculated using the formula above will be evenly distributed on the quantity of hidden layers.

The most widely used methods are evolutionary algorithms, exhaustive search, and Growing and Pruning algorithms, but they have many problems such as time-consuming and do not respond to the complexity of the problem to be solved.

Structure of neural network still determined using trial-and-error. A few general approaches to automatic topology learning among them Growing algorithms, Pruning algorithms, evolutionary algorithms, and reinforcement learning approaches.

2.4.1.1. Pruning algorithms

The pruning approaches start with a large network and then prune it. The aim of the pruning the network is to save storage in the memory and decreases the training time that allowed for the cases in which the pruning of network do not affect the accuracy. This method reduces the number of operations performed inside the network and therefore reduce the compute cost of the operations. This method increases the level of sparsity in the weights by setting weights to zero or eliminating connections. The best pruning algorithm is based on the function that eliminates weights that do not improve the accuracy. There are two methods to achieve that. The first method is automatically set the weight to zero if it is below a certain threshold. The second is to set a small percentage of weights to zero.

A large number of connections helps a neural network to have more capacity for learning training set error. Depending on pruning, it is possible to reduce testing errors. The generalization factor GF is close to one, which means that pruning is effective.

$$GF = \frac{SSE_{TEST}}{SSE_{TRAIN}} \quad (2.10)$$

SSE is the sum of the square of the errors.

The pruning algorithm steps:

1. Choose a reasonable network
2. Train the network until we obtain a reasonable solution.
3. Prune weights
4. Train the obtained network

Repeat the tree last steps until we obtain the optimal neural network structure.

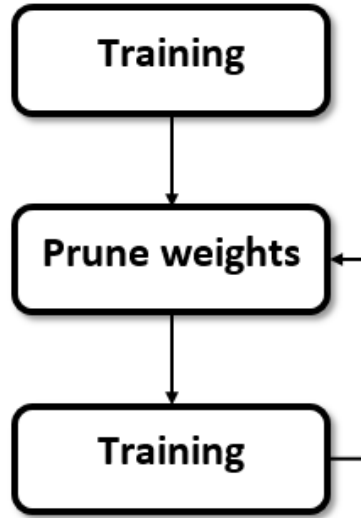


Figure 2-9 Neural network pruning

There are many methods based on pruning algorithm. Among them the method that uses Hessian matrix [46], this method called Optimal Brain Damage, which deletes the parameters with the lowest importance. The deletion obtained by setting this parameter to zero and they are not updated any more. Another method called Optimal Brain Surgeon [47], which tries to choose the weights in a much better way, this method try to calculate the inverse Hessian matrix. Another method much simpler is the weight magnitude [48], which depends on pruning all weights below a defined threshold θ .

$$w \leftarrow \begin{cases} w & \text{if } w \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

The Optimal Brain Damage method and the Optimal Brain Surgeon method are widely used but are computationally expensive.

Another method called Karnin Sensitivity [49], which depend on Sensitivity value SV as pruning value. The idea is to excising weights without affecting the network.

The SV value obtained using this equation:

$$S_{ij} = -\sum_0^{N-1} [\Delta w_{ij}(n)]^2 \frac{w_{ij}^f}{w_{ij}^f - w_{ij}^i} \quad (2.11)$$

Where w_{ij}^f is the value of weight after training, w_{ij}^i is the initial value of the weight, and n is the number of epochs.

2.4.1.2. Growing algorithms

The growing neural network algorithm was initially proposed by Vinod et al [50] start by creating a network contain only the input layer and the output layer. The starting network does not contain hidden layers and then the network grows incrementally by adding one neuron at a time. Based on a predefined criterion the algorithm stops adding neurons to the network [51].

In the following, the Tower algorithm, pyramid algorithm, the cascade correlation algorithm, Meiosis Networks, and Automatic Structure Optimization are introduced.

1. Cascade correlation algorithm

In 1990, Fahlman presents the Cascade correlation algorithm [52] in the context of networks that compute with a smooth signal function such as the sigmoidal signal function. Cascade correlation algorithm generates a cascading architecture.

Initialization: the network start with a minimal network structure. The network composed of n input neurons and p output neurons with full feedforward connectivity without hidden layers.

Training: the network is trained using backpropagation learning until no further reduction in error takes place.

Candidate generation: each generated hidden neuron is connected to all input neurons without connecting to the other candidate neurons and output neurons. This neuron receives $n+1$ input including the bias. In this step, the generated candidate neuron still not connected to the p output neurons.

Correlation maximization: The weight of the generated neuron is adjusted using the backpropagation. The benefit of training is to maximize the correlation C between the signal of hidden neuron and the residual output error:

$$C = \sum_{j=1}^p | \sum_{k=1}^Q (S(z_h^k) - S_{av})(\delta_j^k - \Delta_j) | \quad (2.12)$$

Where $S(z_h^k)$ is the signal of the hidden neuron h corresponding to the input pattern X_k ; $S_{av} = \frac{1}{Q} \sum_{k=1}^Q S(z_h^k)$; δ_j^k is the residual output error at neuron j for pattern k ; and $\Delta_j = \frac{1}{Q} \sum_{k=1}^Q \delta_j^k$ is the average scaled error on the entire pattern set. When the correlation C no longer increases, this step ends.

Candidate selection: once this correlation measure has been maximized, the candidate neurons freeze its incoming weights and add connection to the output neurons. If the error is high than desired, retrain the above steps is repeated and a new candidate hidden neurons will be generated.

The generated neuron receives its inputs from all input neurons, and from the previously added hidden neurons. The frozen weights never change again. These steps are repeated until the acceptable level of error is achieved.

2. Tower and pyramid algorithms

Tower and pyramid algorithms [53] and upstart algorithm are a constructive algorithm, which incrementally builds a network.

Figure 2-10 presents the topologies Generated by Tower, Pyramid and Upstart Algorithms:

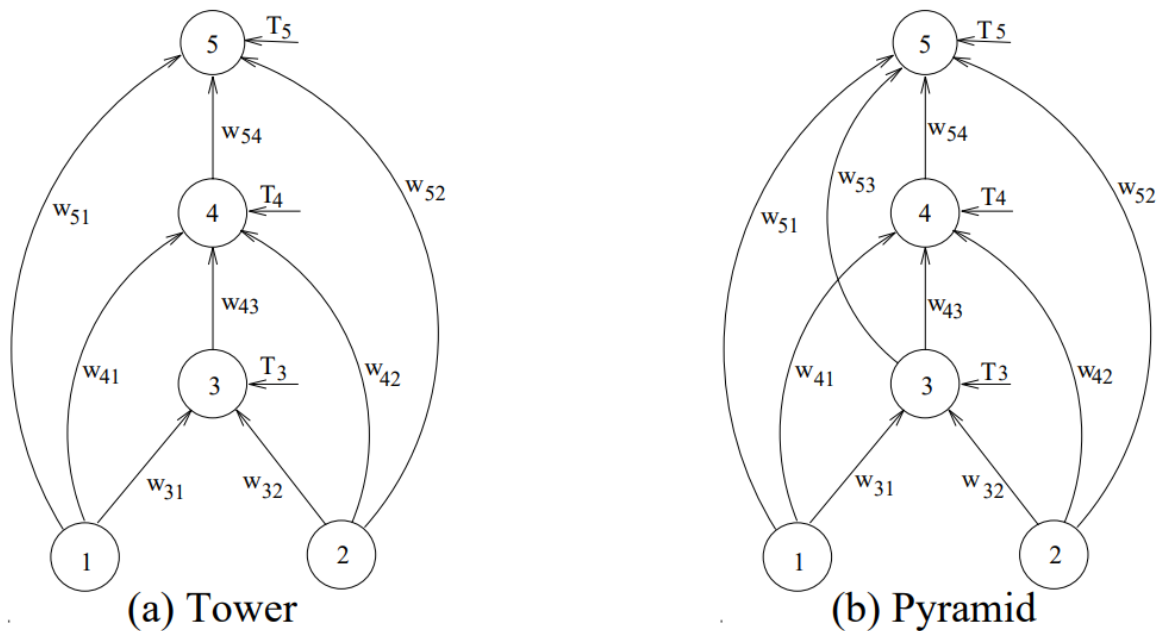


Figure 2-10 the topologies Generated by Tower, Pyramid and Upstart Algorithms

where w_{ji} is the weight between neurons j and i and T_j be the threshold of neuron j .

Tower algorithm: the tower algorithm constructs a tower based on a Multi-layer network of threshold logic units. The bottom-most neuron in the tower receives n inputs, one for each component of the pattern vector. The k^{th} neuron of a tower receives n inputs of the pattern and in addition, receives the output of the $(k-1)$ neuron immediately below in Figure 2-10 (a). The structure of the tower is built by training one neuron at a time until the desired classification accuracy is attained or addition of a neuron ceases to reduce the classification error.

Pyramid algorithm: the pyramid algorithm constructs a network based on a Multi-layer network of threshold logic units. The pyramid algorithm is like the tower algorithm which trains one neuron at a time except for the k^{th} neuron of a tower receives n inputs of the pattern and in addition, receives the outputs from each of the $(k-1)$ neurons below in Figure 2-10 (b).

3. Meiosis Networks

Hanson introduces meiosis networks in 1989 [54]. The meiosis networks vary with the Multi-layer perceptron and conventional neural network in how to determine the weights. Each weight w_{ij} follows a normal distribution:

$$w_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2) \quad (2.13)$$

For every connection have two learned parameters: μ_{ij} and σ_{ij}^2 .

The idea behind this algorithm is to allow neurons to perform Meiosis, which mean cell division.

4. Automatic Structure Optimization

Bodenhausen introduces Automatic Structure Optimization in 1993 [55]. This algorithm uses the confusion matrix to guide the topology learning.

2.4.1.3. Evolutionary algorithms

Evolutionary algorithms are a generic population-based metaheuristic optimization algorithm. Evolutionary algorithms use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. There is a different type of Evolutionary algorithms among them Genetic algorithm, Genetic programming, Evolutionary programming, Gene expression programming, Evolution strategy, Differential evolution, Neuroevolution, and Learning classifier system. Many researchers find the Genetic algorithm a good solution to identify the optimum MLP architecture.

A genetic algorithm is the most widely known type of evolutionary algorithms. A genetic algorithm was initially conceived by Holland as a mean of studying adaptive behavior [56]. Genetic algorithm encodes the solution space as genes.

Genetic algorithm starts by encoding the features of the neural networks into specific chromosomes. The chromosome represents a sequence of bits with values 0 or 1. Genetic algorithm evolves the solution according to the following basic pattern:

- 1) Randomly generate the first population of solutions.
- 2) Application of a fitness function.
- 3) Selection of the best solutions.
- 4) Generation of new solutions using crossover and mutation.
- 5) Repetition of steps 2–3–4 for n iterations.
- 6) Selection of the best-found solution.

Many techniques to generate a neural network using genetic algorithm among them NEAT [57], HyperNEAT [58] and ES-HyperNEAT [59].

2.4.1.4. Reinforcement learning

Reinforcement learning is a sub-field of machine learning that is inspired by behaviorist psychology. Reinforcement learning is a type of programming that train algorithms to maximize reward in a particular situation. Reinforcement learning algorithm (agent) learn by interacting with its environment through maximizing some notion of cumulative reward. The agent obtains rewards by performing correctly and penalties for performing incorrectly. The agent learns by minimizing its penalties and maximize its rewards without intervention from a human.

Reinforcement learning can be a solution to determine the structures of the neural network. The recurrent neural network is presented as an agent, which can generate bitstrings. Those variable-length bitstrings encode neural network topologies.

Reinforcement learning approach was applied to designing neural network architectures for computer vision in [60]. Another research, which used the Reinforcement learning algorithm from [61] to train state of the art, models for CIFAR-10 and the Penn Treebank dataset in [62].

2.4.1.5. Convolutional Neural Fabrics

Convolutional Neural Fabrics are introduced in [63]. This method tries to select the optimal topology through learning an ensemble of different CNN architectures. The idea is to propose a “fabric” that embeds an exponentially large number of architectures. The fabric consists of a 3D trellis that connects response maps at different layers, scales, and channels with a sparse homogeneous local connectivity pattern.

2.4.1.6. Optimization of MLP architecture Using Heuristic techniques

There are many heuristic techniques for optimization of the ANN architecture among them genetic algorithm, Taguchi, Tabu search and decision trees. These algorithms are capable to identify the features of the ANN which able to minimize the prediction error.

1. Taguchi method

The Taguchi method was developed by Genuchi Taguchi in 1950s. The method is based on the statistical analysis of data and offers a simple means of analysis and optimization of complex systems. This method can determine the optimal neural network architecture by optimizing the performances searching [64] [65] [66].

2. Tabu search

Tabu search is a metaheuristic technique, which represents a solution for many optimization problems [67]. Tabu search has the ability to escape from local minima allowing to exhaustively exploring the solutions space.

3. Decision trees

The Decision trees is a widely used learning method. This method is very popular due to its ease of use, low computational time and the ability to quickly analyzing the results. Decision trees were used for classification problems [68] and to create stratified regression problems models by Neville in 1998 [69].

To train a decision tree a set of neural networks with different properties are initially tested by measuring the average error rate obtained on the validation set. Then select the leaf node with the best performance. This leaf could contain different neural networks architecture, which will be tested searching to find the one with the smallest average percentage error.


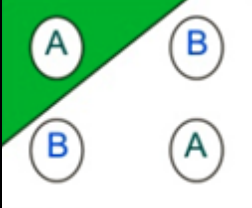




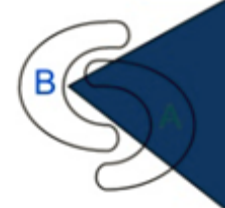


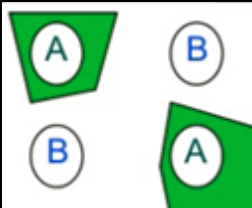


2.4.2. Ability learning neural network architectures depending on number of hidden layers

In experiments made by various authors in the literature specified for specific issues such as the optimal architecture, but they give results only under certain conditions. Identifying the architecture of the ANN is a complex issue and a critical step in the design of the ANN, due to the lack of generally valid theories. At present, a viable theory for designing the ANN based on the level of complexity of the considered problem does not exist. Theoretically, it can be shown that an MLP composed of one hidden layer [70] can approximate linear functions found in most practical applications, and an MLP composed of two hidden layers is able to approximate any nonlinear function. In general, the complexity of the problem affects the structure of the ANN [7] whereas the quantity of hidden layers/units is affected [71]. The benefits of using multiple hidden layers are improving the predictive ability of the network, in fact, due to the increase of the size of the network imposed by the nonlinear properties of the problem the training time increases based on the rising of the complexity of the neural network structure. In practical applications can be started with an MLP composed of one hidden layer, and if not achieved the expected results, it is recommended to increase the quantity of hidden layers. Alternatively, it initially starts with complex ANN architecture and then based on the experimental results attempts to reduce the quantity of hidden layers/units.

It is generally considered that an MLP composed of three hidden layers is sufficient to solve any problem (although the human brain, whose function we want to simulate has an enormous number of neurons and an untold number of layers) [72].

Table 2-2 presents some types of regions of decision that can be learned by different neural network architectures.

Table 2-2: Ability learning neural network architectures depending on number of hidden layers

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

2.5. Conclusion

This section discusses the structure of the ANN, its influence of hidden layers in the network's capacity to learn. Network size influences the nature of a network's complication, hence the network's competency.

Several widely used methods to identify the architecture of the ANN were discussed and the weaknesses of these methods are presented. It concluded that until now there is no general method to identify the architecture of the ANN based on the complexity of the problem to be solved.

The training time and the generalization of the network is the main problem, which confronts us in the process of the design of the ANN structure. The quantity of hidden layers depends on the complexity of the problem to achieve good classification accuracy. A high quantity of the hidden layers may cause overfit to the training dataset.

Avoiding the overfitting and the underfitting is a challenge in the design of the ANN structure. Overfitting accrues when the model unable to generalize. If the difference between the value of training error and test error becomes large, it will cause overfitting. We obtain Underfitting when the model is not able to obtain a sufficiently low error in the training set.

Chapter III

3. Determination of optimal Neural Network Architecture using clustering techniques

This chapter elaborates on a given method to reach the optimum MLP structure. Using Clustering techniques on neural network training data and based on given criteria, different clusters can be identified. The results can work as an indicator to identify the deepness of the problem at hand. From the number of clusters, one can determine how many layers are hidden in a given multilayer perceptron. This paper analyzes how different datasets are adopted, for example, ordinal, definite and collection of various variables or even interval, that affect the measurement to identify groups per any set of data. This chapter outlines more information on the extent of this projected method. It minimizes chances for the formation of a complicated ANN architecture through the formation of several hidden layers. On the contrary, it is worthy to note that adopting fewer hidden layers has some adverse outcomes. It becomes challenging to determine the nature of the problem.

3.1.Introduction

Identifying the optimum architecture of the neural network is very important due to several reasons. An Excessive quantity of hidden layers can make the ANN more sophisticated and increase the training time and can give the network more specialization and decrease generalization. By using a few numbers of hidden neurons, the neural network cannot achieve satisfactory performance. On the contrary, using a large quantity of hidden units, the ANN will memories the patterns instead of learning from the training set [73], which impacts the generalization ability of the ANN to interpolate and extrapolate data that it has not seen before. To achieve the highest possible level of generalization performance trading of the training error against network complexity is necessary.

The proposed clustering method use pattern recognition techniques to define the optimal architecture of an MLP.

The idea behind this clustering technique is to group the neural network's training set by adopting relevant ways of pattern identification [3] [74] as per specific guidelines.

Based on the information collected from clustering the training dataset we conclude that the quantity of groups obtained in the case of at least 90% of input forms clustered is equal to the best possible quantity of hidden layers for an MLP.

The quantity of groups must be taken as few as possible in order to obtain a network with the lowest number possible of hidden layers to decrease the complexity of the network.

Considering the stability of this grouping method when reference distance (RD) is increased, existing clusters will vary as long as extreme cases are avoided. i.e. a large RD or short RD. every element stands for the given cluster. Here, we need to understand the linkage distances.

At this stage, we take into consideration linkage distances, such as complete-linkage clustering, single-linkage clustering, mean distance clustering, and the proportion of training elements that are grouped. The total quantity of groups recorded will represent total hidden layers in a given ANN.

3.2.Optimization problem

To date, there are no exact theories to choose a precise structure of a neural network appropriate to the complexity of the involved problem. A general and applicable theory for determining the structure of an ANN not yet defined. Under these circumstances, the design of a neural network structure depends on exhaustive simulations and designer experience. Each issue has a specific structure and requires certain conditions to achieve the desired results.

Point out some of the most widely used methods to identify the architecture of MLP such as Trial and error, Heuristic search [75], Pruning, and constructive algorithms [76], Rules of Thumb and so on. Majority of designers start with an arbitrary size of network whether starts from a small size or large size then iteratively try to modify the size up or down until a perfect size achieved.

3.3.The projected method for optimization

The idea behind the projected method is to determine the complexity of the concerning problem by clustering the learning data of MLP. Based on a set of criteria it is possible to correlate the pattern revealed in the training dataset with the optimum quantity of hidden layers. Agglomerative Hierarchical Clustering (AHC) algorithm used to solve the optimization problem [77]. The proposed optimization algorithm to identify the optimum structure of an MLP follows several steps to respond to the imposed criteria [78]. The following steps explain our proposed optimization algorithm in detail:

1. Initialize the problem by setting up training data by the selection of the suitable normalization technique. Normalization helps to eliminate conflicting records. The next step is the selection of the Distance Measure technique. This process is made successful considering data types adopted. Next step calls for the selection of clustering technique:
 - Normalization: Preparation of the data set used to train the network by the elimination of noise, incomplete records, and records with large dissimilarities to other data.
 - Select the distance measure technique: define the type of data to select the distance measure suitable with the type of data used.

- Select Clustering Techniques: select the linkage method suitable for the type of data.
2. Implementation of AHC algorithm to generates a Dendrogram. The algorithm starts by taking each data point as a single cluster and iteratively merge pairs of closest clusters at each step. AHC algorithm calculates the dissimilarities and distances between instances using the selected distance measure technique then represents clusters as a Dendrogram:
 - Input: objects list of singleton clusters $C_1, C_2 \dots C_n$.
 - Calculating the similarity between objects using the selected distance measure technique.
 - For each pair of closest clusters $\{C_i, C_j\}$ merge C_i and C_j
 - Remove C_i and C_j from the list of set and replaced with $C_i \cup C_j$
 - Iterates while all objects are in a single cluster.
 - Output: Generated tree of clusters (Dendrogram).
 3. The generated tree will be cut at a given value of the distance between clusters. This distance value will be referred to as "Reference Distance". The Reference Distance will be chosen based on a set of criteria, which must be realized. The algorithm will respond to the criterion until the optimal value of Reference Distance is defined [79]. The first criterion depends on the percentage of clustered items in the training set which must not be less than 90%. The second criterion imposes that by increasing the Reference Distance the number of obtained clusters does not change. With taking into consideration, the excessive cases which will be avoided. Such as a very short distance, for which each element represents a group, or a relatively large distance, for which all elements are placed in a single group:
 - Calculate the optimal cut:
 - Criterion 1: while the clustered items less than 90% increase Reference Distance. Then cut-off tree and return quantity of groups.
 - Criterion 2: the quantity of obtained groups must not change by increasing the RD obtained based on criterion 1.
 - Return the quantity of obtained groups based on the defined value of RD.
 4. Determine the quantity of hidden layers based on results obtained from grouping of the learning data.
 5. Determine the quantity of hidden units for each hidden layer and generate the neural network structure.

3.4. Projected Method Stages

The projected method seeking to evaluate the level of complexity of the concerning problem by clustering the learning data of an MLP and then interpret the obtained cluster to define the number of hidden layers. By grouping the learning data of the MLP based on conventional methods of pattern recognition [80] [81] following a set of criteria [82] [83] [84] to generate a number of clusters. In this case, we can take the quantity of obtained groups as the optimum quantity of hidden layers for the MLP structure [85]. This method will take into consideration a number of criteria which be discussed in this chapter. The algorithm used to design the MLP is developed over several stages.

3.4.1. Establishing the learning data

This stage calls for creating training data for neural network learning. At first, eliminate conflicting records, noise and disparate records is necessary to determine the distance measure suitable with the type of data used.

3.4.2. Establishing the quantity of input units

The quantity of input units will be equivalent to the quantity of features [41].

3.4.3. Identifying the quantity of output units

The quantity of output units will be equivalent to the quantity of classes when we have to solve a classification problem.

3.4.4. Identifying the quantity of hidden layers

In order to get a quantity of groups equal to the optimum quantity of hidden layers through clustering the training dataset it is necessary to meet the following condition:

- The clustering of training dataset must be performed with a covering of at least 90% of the elements of the dataset. Because with at least 90% of training dataset elements we have adequate representation of the dataset forms and the result can be extrapolated to the entire dataset.
- The number of clusters obtained must be constant if we increase the value of RD, which indicates that the grouping is stable. The extreme cases are not taken in to account among them a very short value of reference distance in which each element of the training dataset is considered as a cluster, or a relatively large value of

reference distance in which all elements of the training dataset are clustered into one single cluster.

By applying, the criteria described above the quantity of groups obtained will be considered as the optimum quantity of hidden layers for an MLP. The main aim of this study is to attain the best criteria, which lead to getting a quantity of groups equal to the optimum quantity of hidden layers.

The linkage distances, such as complete-linkage clustering, single-linkage clustering, mean distance clustering, and the percentage of input forms that are clustered below and above the average distance are taken into consideration in the step of designing of the MLP.

Based on experiment results it was observed that the quantity of groups identified by grouping the learning data based on the criteria described above is approaching to be equal to the optimum quantity of hidden layers.

3.5.Data mining techniques used

Data Mining techniques are used to analyses datasets, extract useful information, and establish relationships based on pattern discovered from data to find solutions for some problems. Based on information extracted from data using data mining we can make reasonable predictions concerning relationships between reviled patterns in data.

In this chapter, several Data Mining techniques are used to analyze the training dataset to define the optimum MLP structure, among them clustering, Regression method, classification algorithms, and prediction.

3.5.1. Clustering Algorithm Used

The learning data of MLP is clustered based on a set of pattern recognition techniques to extract common features [14], [15].

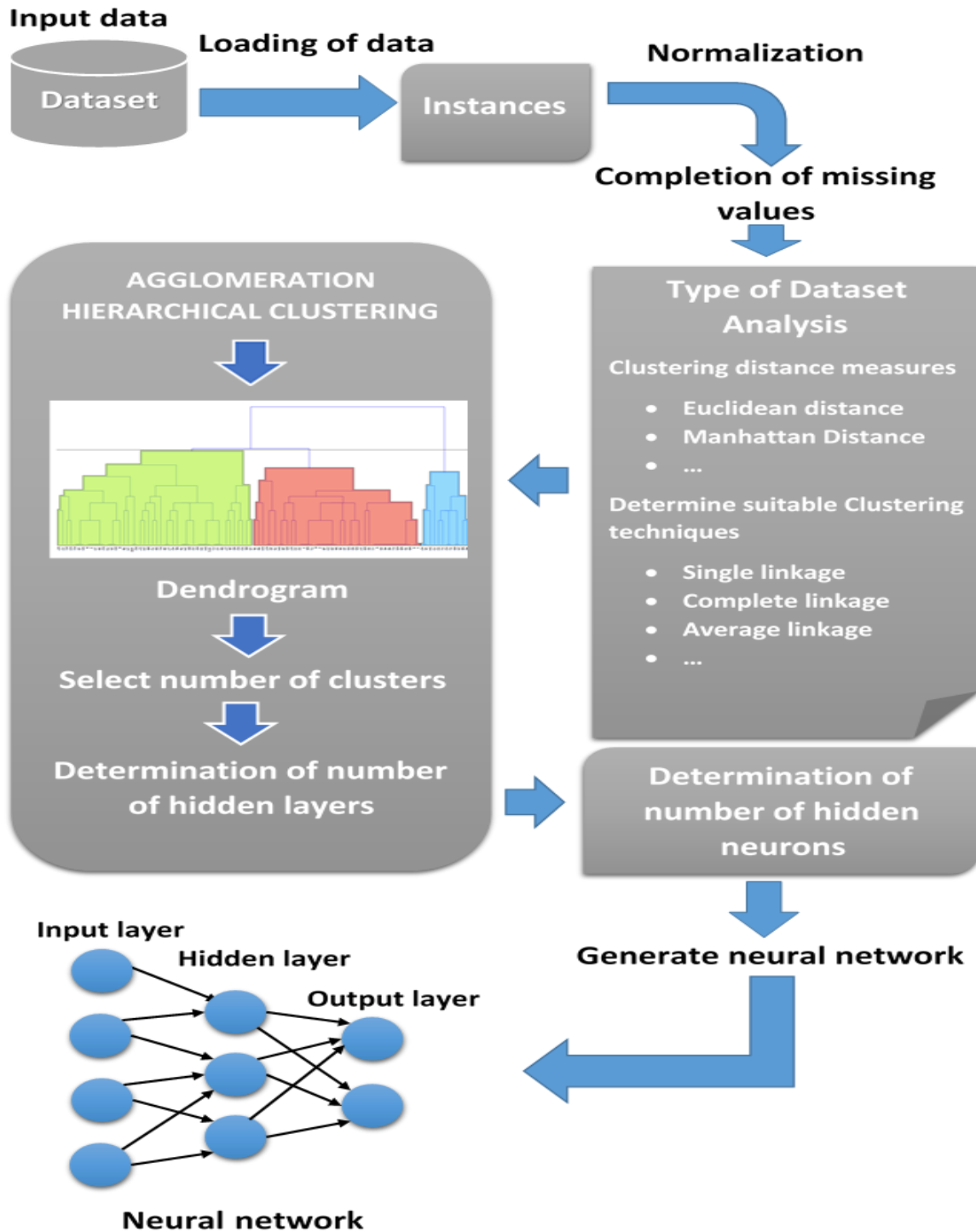


Figure 3-1 the proposed framework for clustering method

The grouping method directs in setting the neural network's best configuration through the outcome from using the "Agglomerative Hierarchical Clustering (AHC) Algorithm".

The AHC algorithm helps to analyze data to create clusters based on the information obtained from data. Based on the relationship between elements of a dataset and the information extracted from each element the data are grouped into clusters.

Clustering objects of a dataset is a determination of how much the objects in one cluster is similar and at the same time how much the objects in different clusters are dissimilar.

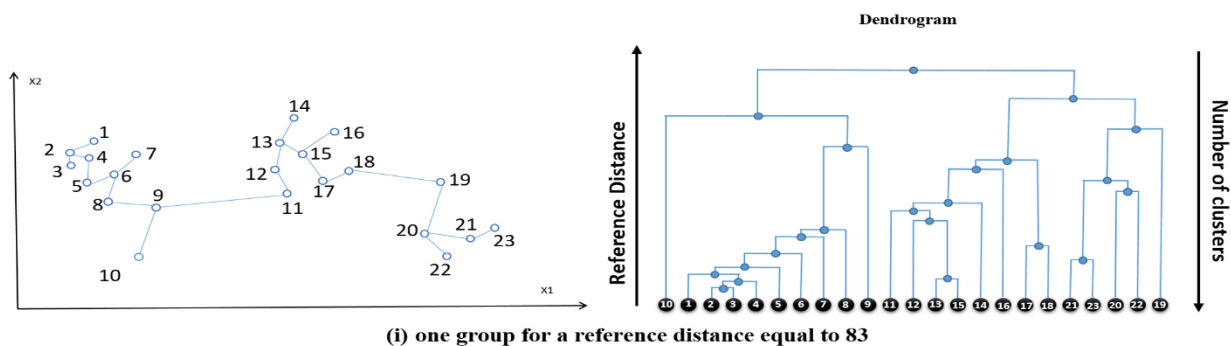
A hierarchical grouping is presented by a cluster which contains sub-clusters. The clustering is made such that it simulates a tree. Each node of the tree will represent a cluster of its sub-cluster. The roots are like the mother cluster and support all clusters and data set. The leaves can stand for an element of a given cluster. This arrangement of clusters can allow for the division of clusters in a series.

AHC algorithm [86], [87], [88], [89], [90] perform the clustering of the dataset by taking each element of the dataset single handedly then connect adjacent clusters repeatedly till a solitary cluster is obtained.

The hierarchical clustering can be divided into two techniques, the first technique is Agglomerative hierarchical clustering (AHC) and the second is Divisive hierarchical clustering. The AHC has widely used clustering method. The hierarchical clustering can be visualized as a Dendrogram, which represents the tree of clusters. Figure 3-1 [91] demonstrate the hierarchical clustering using an example.

The value of distance selected for cutting all segments with greater value will be called in this chapter by "Reference Distance". Based on the value of RD the quantity of groups is determined. Based on the quantity of obtained groups we can identify the structure of the MLP.

Figure 3-1 Presents the quantity of groups obtained for various values of RD.



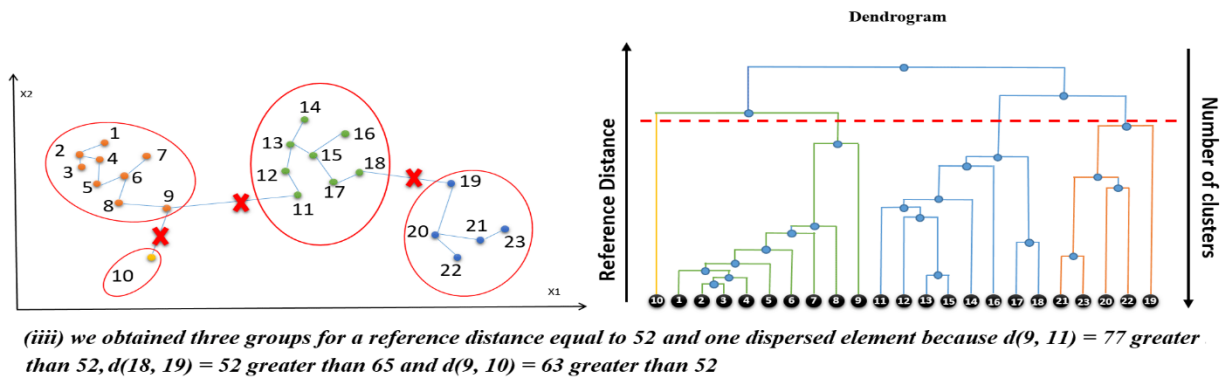
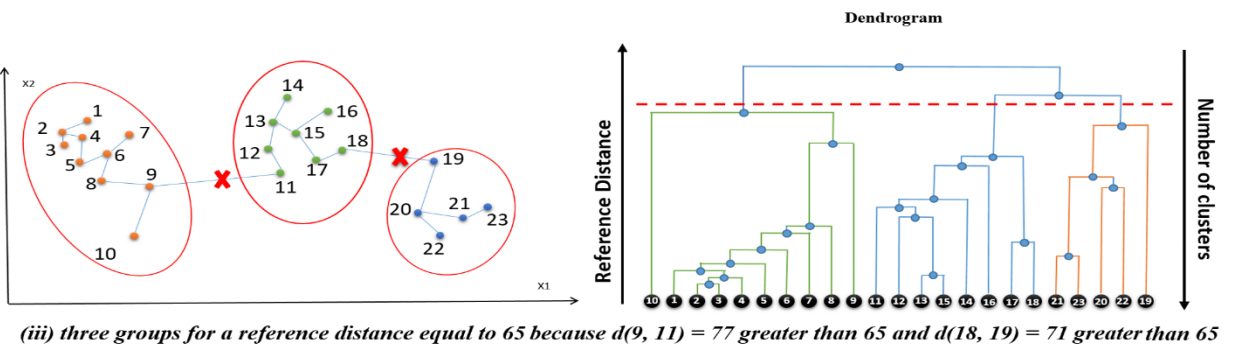
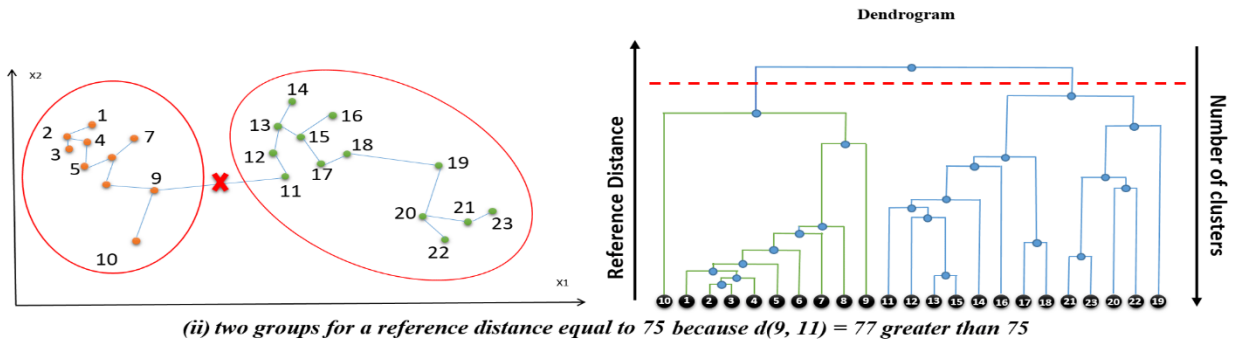


Figure 3-2 number of clusters obtained for various values of RD [92].

3.6. Creating and training neural networks using data mining techniques.

The data group is a more explicit technique of determining data patterns, structure, and design. From the grouping method applied and considering the outcomes, we can understand the problem by grouping the neural network's training data set. Hidden layers are adjustable as per the rigidity of the issue at hand. Therefore, this technique takes obtained clusters to be the best quantity of hidden layers.

3.6.1. Presentation of the case studies used to validate theoretical outcomes

Here, we present six case studies helping in the validation of theoretical outcomes. The empirical data are from different categories to ensure all areas are considered.

Sonar dataset (Sonar, Mines vs. Rocks) is a connectionist Bench Dataset was contributed to the benchmark collection by Terry Sejnowski at the Salk Institute and the University of California at San Diego. The dataset was developed in collaboration with R. Paul Gorman of Allied-Signal Aerospace Technology Center [93].

The dataset contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. In which it contains 97 patterns obtained from rocks. All signals obtained from different angles to distinguish between cylinder and rocks.

Each instance contains 60 attributes in the range of 0.0 to 1.0. Each attribute has a value of energy based on a defined frequency band for a time span.

3.6.1.1. The analysis of an ECG signal

ECG signal is the electrical activity of the heart, this is a bioelectric signal, and the analog periodic voltage with a maximum value of about 1 Millivolt, a signal representing a cycle of contraction of the heart. The Electrocardiogram ECG is used in conjunction with other clinical data to diagnose various heart diseases [94].

ECG signal processing consists in extracting useful information in a record gross signal is initially filtering the signal and then finding the three main components of the signal P wave, QRS complex and T wave.

3.6.1.2. The analysis of a signal from a Sonar

Sonar is a technology similar to radar site that uses sound to explore the waters of the oceans and seas to determine the position and nature of objects in the water. A sonar, Sound, Navigation and Ranging is a device used to detect and possibly interpreted the underwater sound, more specifically to detect and locate objects in water to categorize the nature of these objects. Sonar is a device used by fishing vessels, research vessels, military ships, submarines, etc.

The study originated from a sonar signal after pre-processing is converted into an electrical signal continuously and irregularity to classify an obstacle they meet. In this study, the received signal should be classified as Rock or Submarine.

3.6.1.3. Classification of Statlog (Landsat Satellite) Data Set

This database represents Multi-spectral values of pixels in a satellite image with the classification associated [95].

The original Landsat data for this database was generated from data purchased from NASA by the Australian Centre for Remote Sensing and used for research at the Centre for Remote Sensing University of New South Wales Australia.

Each frame of Landsat MSS imagery has four digital images belonging to the same scene in various spectral bands. Each pixel in the digital image is an 8-bit binary word, zero corresponds to black and 255 corresponds to white. A pixel has a spatial resolution of about 80m x 80m. Each image has 2340x3380 pixels. The database is a subarea of a scene composed of 82x100 pixels. Every line of data corresponds to a 3 x 3 square neighborhood of pixels inside the 82 x 100 sub-area. Every line has the pixel values in the different four spectral bands of the nine pixels in the 3 x 3 neighborhood, and a number refers to the label of classification. This table represents the corresponding number of each class:

Table 3-1: Decision classes, the number is a code for the following classes

Number	Class
1	<i>red soil</i>
2	<i>cotton crop</i>
3	<i>grey soil</i>
4	<i>damp grey soil</i>
5	<i>soil with vegetation stubble</i>
6	<i>mixture class (all types present)</i>
7	<i>very damp grey soil</i>

3.6.1.4. Glass Identification Dataset

The Glass Identification Dataset [96] created for validation of theoretical results. Using Glass Identification dataset, it is possible to predict the type of glass to know if it is a "float" glass or another type.

3.6.1.5. Description of case studies

Initial case studies represent the ECG signal. It is made successful with the bioelectric signal. Analysis outcomes from the method help in identifying cardiac disorders. As well as the results of ECG signal analysis for accurate diagnosis thereof need to be calculated, and other parameters, heart rate, duration rate, the number of P-waves in the cardiac cycle (in certain cardiovascular diseases for the occurrence of complex two P waves are required QRS), assessment of intervals and segments. Using these results, it has been trained a neural network to recognize the most common cardiovascular disease.

Landsat satellite analysis. In this instance, scenes become analyzed through a combination of maps, radar, and data from the various spectrum, among others.

The last case study relates to a signal from a sonar. This is an analog ECG signal as the initial preprocessing requires that consists of a pre-amplification and filtering. In order to be analyzed by a neural network as the signal will be sampled to obtain a vector form consisting of 60 features.

In each case study was presented to the neural network architecture proposed by the simulator NeuroShell the quantity of inputs, outputs, quantity of hidden layers and the activation functions [97] for each layer of the proposed neural network. It was observed that it proposes the same architecture for all data sets analyzed the difference is only in terms of the inputs number, the number specific to each data set analyzed.

3.6.2. Determination of optimum MLP Architecture using clustering algorithms

The idea behind this method of determining the best clustering of the set of N training forms for an MLP. When Traditional pattern recognition methods are adopted, training forms can be grouped in a given order to achieve uniform classes. Any unclassified elements are gotten rid of. In the case of a stable system, the total quantity of classes identified represents a neural network's optimum hidden layers' number [14] [15].

3.6.2.1. Determination of the optimum quantity of hidden layers for an MLP to classify Statlog (Landsat Satellite) Dataset

To identify the quantity of hidden layers we use the clustering technique described above. Presents the quantity of clusters obtained for Landsat Satellite images for a distance with reference of 75.

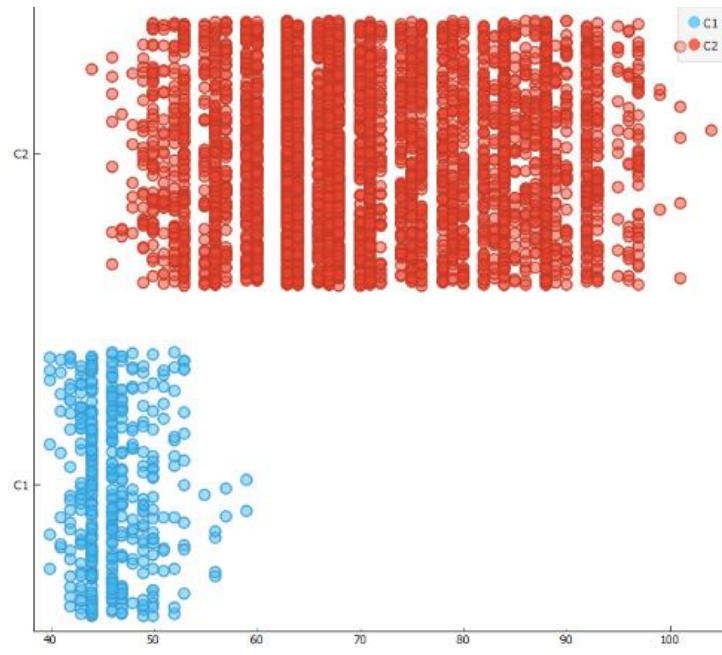


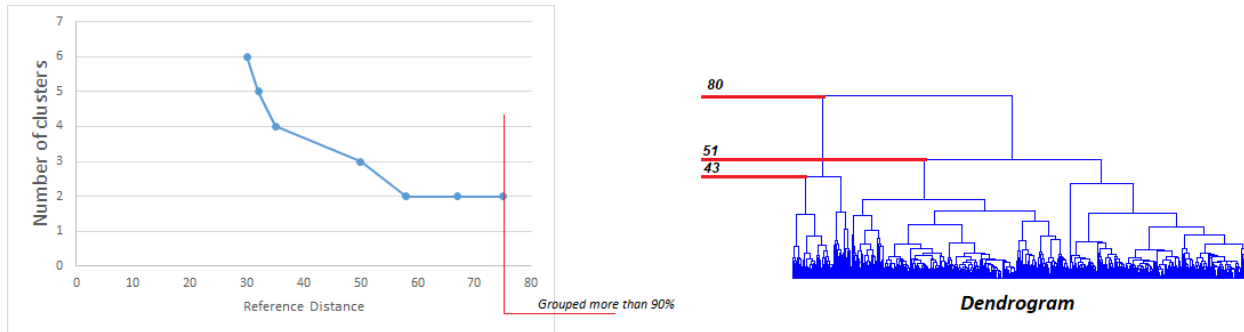
Figure 3-3: The quantity of clusters obtained for Landsat Satellite images with a reference distance $d = 75$, 2 groups were obtained.

In Table 3-2 are given results with the program of grouping to classify Landsat Satellite images of the same scene in different spectral bands.

Table 3-2: Data obtained with the program of grouping for the Landsat Satellite images dataset

Reference Distance	Number of clusters	Number of items grouped in%
75	2	90%
67	2	80%
58	2	70%
50	3	59%
35	4	41%
32	5	38%
30	6	36%

Figure 4-1 shows the quantity of groups obtained based on the values of RD for the Landsat Satellite images dataset and the percentage of clustered elements.



Results obtained using the Euclidian distance

Figure 3-4: The relationship between the number of groups and the value of RD for Landsat dataset

From the above, it is noted that with increasing distance of RD obtained increasingly fewer groups and a higher percentage of grouped elements. For a large enough distance of the RD, all forms of input are clustered into one group.

The idea is to stop at the reference from which we get a grouping of as many forms (a few forms that cannot be clustered), namely more than 90%. The number of clusters so obtained we are going to consider the optimum quantity of hidden layers for the MLP.

Therefore, in the table above we can conclude that the analysis of Landsat Satellite images would have an MLP consist of 2 hidden layers.

3.6.2.2. Identifying the optimum quantity of hidden layers for an MLP used to analyze the components of an ECG signal

ECG signal is composed of three components, P wave, QRS complex, and the T wave. The more advantageous ECG signal components to be analyzed using three distinct neural networks. The advantage comes from the fact that the three components of the ECG signal have different characteristics, so instead of using a complex neural network can use a simpler neural network, tailored to the characteristics of each component.

To identify the quantity of hidden layers we use clustering technique described above. Figure 3-3 presents the number of groups obtained for P wave for a distance with reference of 11.4.

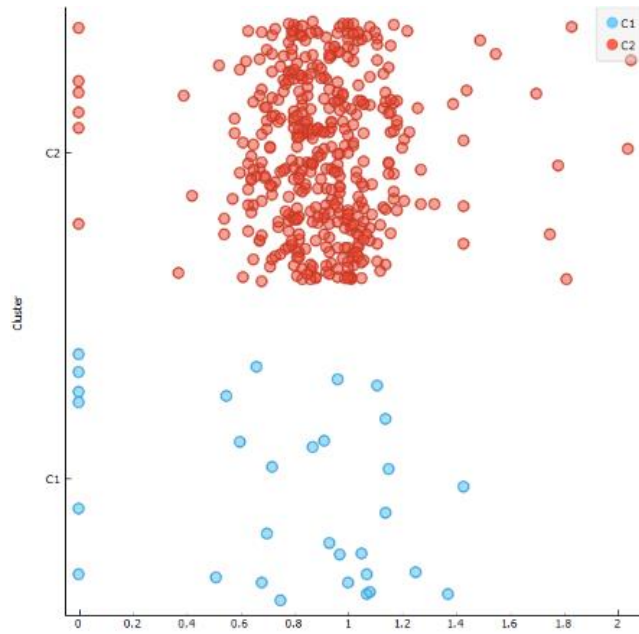


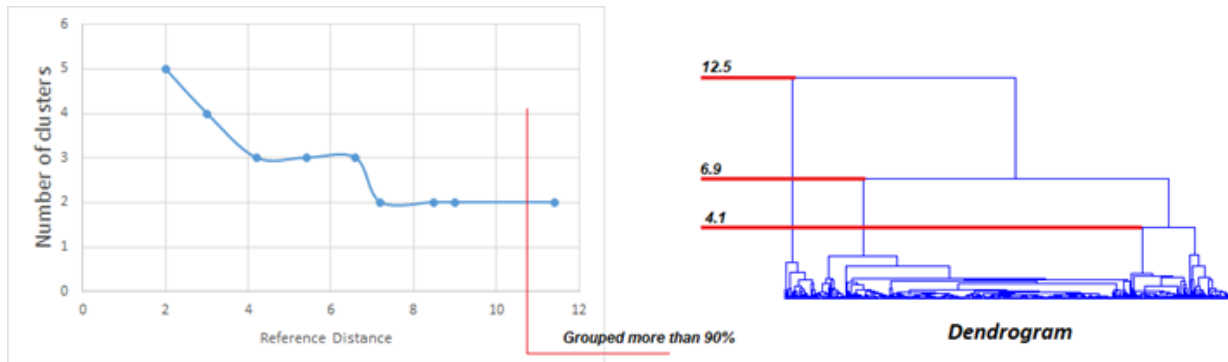
Figure 3-5: The quantity of clusters obtained for P wave with a reference distance $d = 11.4$, 2 groups obtained.

Table 3-3 shows the quantity of clusters and percentage of grouped items for various distances values of the RD.

Table 3-3: Data obtained with the program of grouping for the P wave of the ECG signal

Reference Distance	Number of clusters	Number of items grouped in%
11.4	2	90%
9	2	75%
8.5	2	67%
7.2	2	57%
6.6	3	52%
5.4	3	43%
4.2	3	34%
3	4	22%
2	5	16%

Figure 3-4 shows the quantity of clusters obtained based on the values of RD for the P wave dataset and the percentage of clustered elements.



Results obtained using the Euclidian distance

Figure 3-6: The relationship between the number of groups and the value of RD for the P-wave dataset

Therefore, in the table above we can conclude that the analysis of the P-wave component of an optimal ECG signal would have an MLP consist of 2 hidden layers.

Next, we analyze in the same manner QRS complex and the T wave of ECG signal components, specify that both the QRS complex and the T wave have 452 input forms, as well as for P wave.

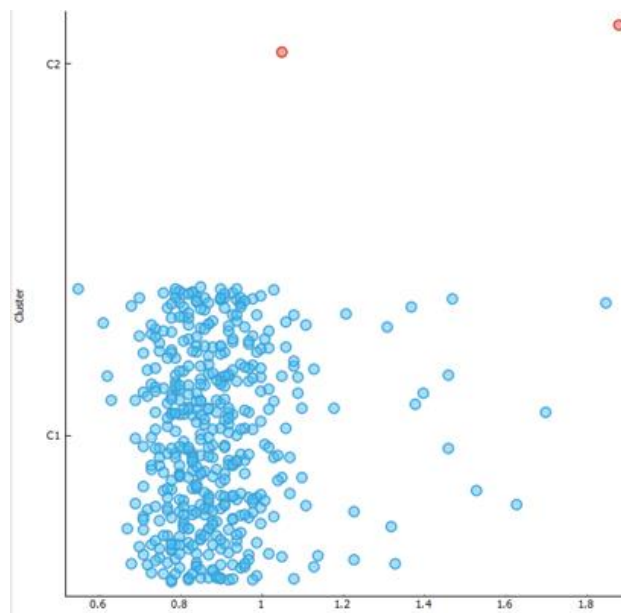
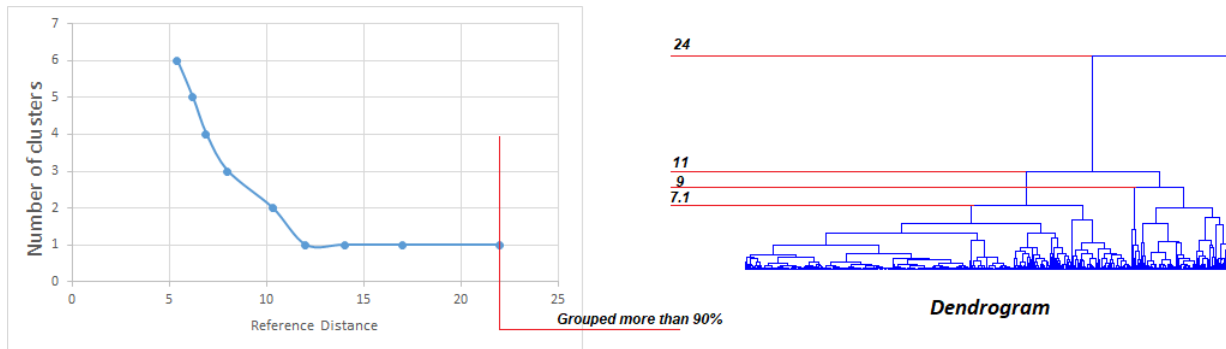


Figure 3-7: The quantity of clusters obtained for QRS complex with a reference distance $d = 22$, we have one group and 2 elements dispersed.

Table 3-4: the obtained with the program of clustering the QRS complex, of the ECG signal

Reference Distance	Number of clusters	Number of items grouped in%
22	1	90%
17	1	70%
14	1	57%
12	1	49%
10.3	2	43%
8	3	33%
6.9	4	28%
6.2	5	25%
5.4	6	21%

Figure 3-6 shows the connection between the reference distances for which the group and the quantity of groups obtained and the percentage of grouped elements for the QRS complex.



Results obtained using the Euclidian distance

Figure 3-8: The relationship between the number of groups and the value of RD for the QRS dataset.

As shown in Figure 3-7, Figure 3-8 and Table 3-4 for a reference distance of 22 was obtained one group and only two elements are dispersed. Therefore, it is concluded that an MLP composed of 1 hidden layer represents the best structure for the analysis of QRS.

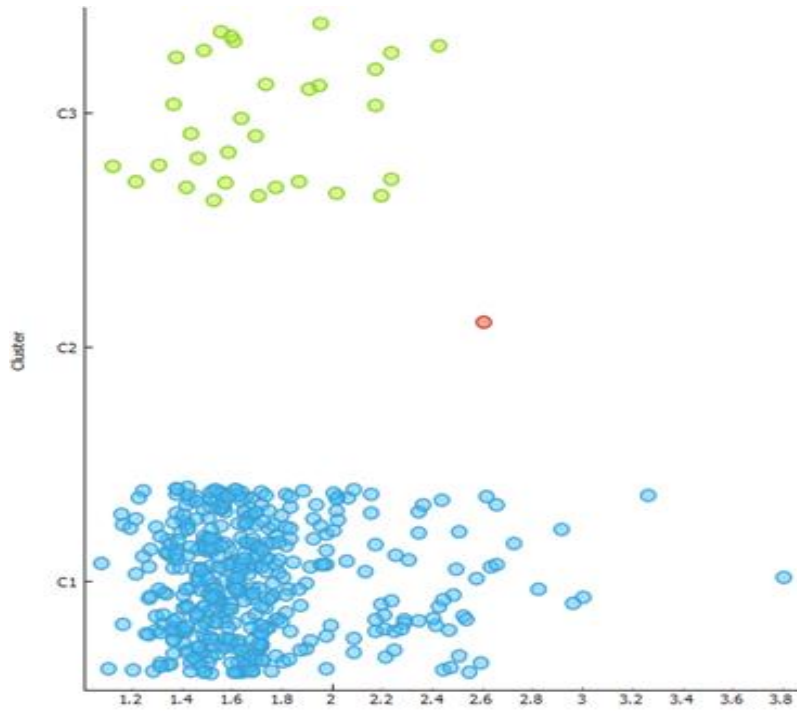
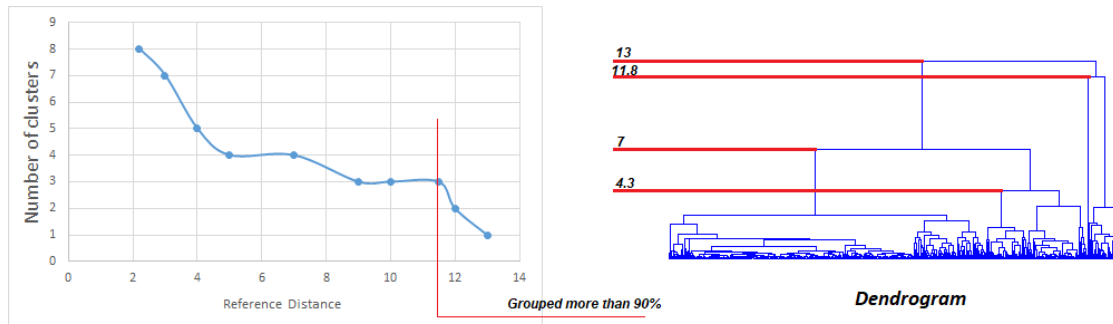


Figure 3-9: the quantity of clusters obtained for T wave with a reference distance $d = 11.58$, was obtained three groups.

Table 3-5: Data obtained with the program of grouping for the T wave, of the ECG signal

Reference Distance	Number of clusters	Number of items grouped in%
13	1	100%
12	2	94%
11.5	3	90%
10	3	78%
9	3	70%
7	4	54%
5	4	39%
4	5	30%
3	7	22%
2.2	8	18%

Figure 3-10 shows the quantity of groups obtained based on the values of RD for the T wave dataset and the percentage of clustered elements.



Results obtained using the Euclidian distance

Figure 3-10: The relationship between the number of groups and the value of RD for the T wave dataset

Therefore, for a reference distance of 11.58 are obtained three groups, so an MLP composed of three hidden layers would be optimal for analysis of the T-wave component of the ECG signal.

3.6.2.3. Identifying the optimum quantity of hidden layers for an MLP used in the diagnosis of ECG signal

The outputs of the three neural networks that analyze components of the ECG signal plus some additional inputs will be input into a neural network used in the diagnosis of ECG signal (can diagnose many heart diseases, the most common in medical practice).

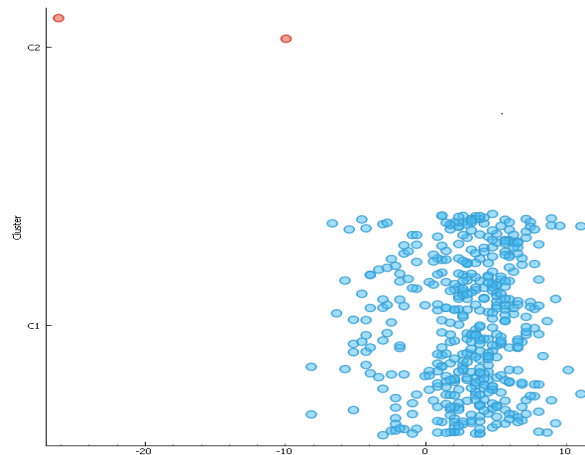
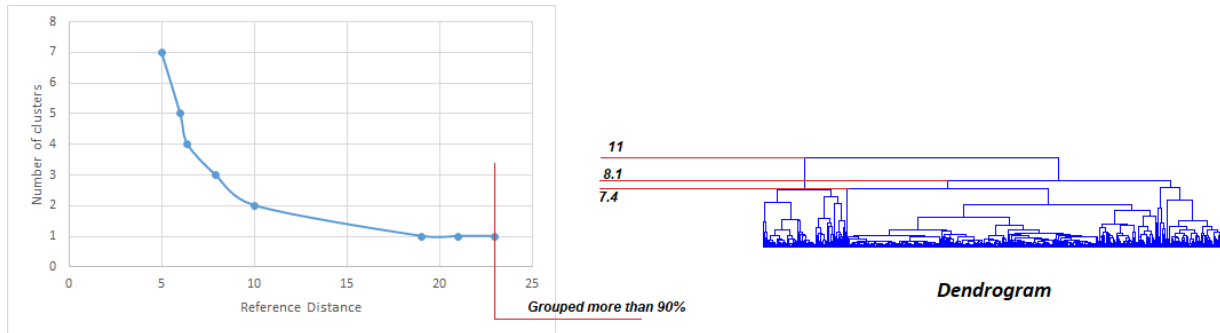


Figure 3-11: The quantity of clusters obtained for ECG signal with a reference distance $d = 23$, was obtained 1 group and 2 of elements that could not be classified

Table 3-6: Data for the diagnosis of an ECG signal

Reference Distance	Number of clusters	Number of items grouped in%
23	1	90%
21	1	80%
19	1	70%
10	2	40%
7.9	3	31%
6.4	4	26%
6	5	23%
5	7	20%

Figure 3-12 shows the connection between the values of RD for which the group and the quantity of clusters obtained, and the percentage of forms clustered.



Results obtained using the Euclidian distance

Figure 3-12: The relationship between the number of groups and the value of RD for ECG dataset

From Figure 3-11, Figure 3-12 and Table 3-6 it is observed that for a reference distance $d = 23$, was obtained 1 group and 2 elements which have not been classified.

Therefore, the reference for a distance of 23 we have only one group. We conclude that a MLP composed of 1 hidden layer for ECG signal dataset would be optimal, because when we obtain the group forms of input into one group to a reference distance of 23, the number of elements that not be grouped, is less than 10% (two forms dispersed of 453 forms of input).

3.6.2.4. Identifying of the optimum quantity of hidden layers for an MLP used for the analysis of a signal from a Sonar

To analyze the signal from a sonar will use the same techniques to analyze the ECG signal.

In Figure 3-13 shows the quantity of clusters obtained with the program of grouping for a reference distance 2.1.

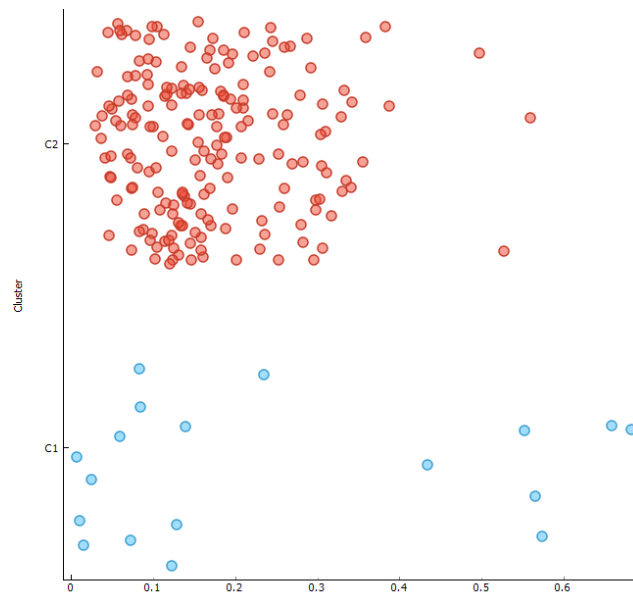
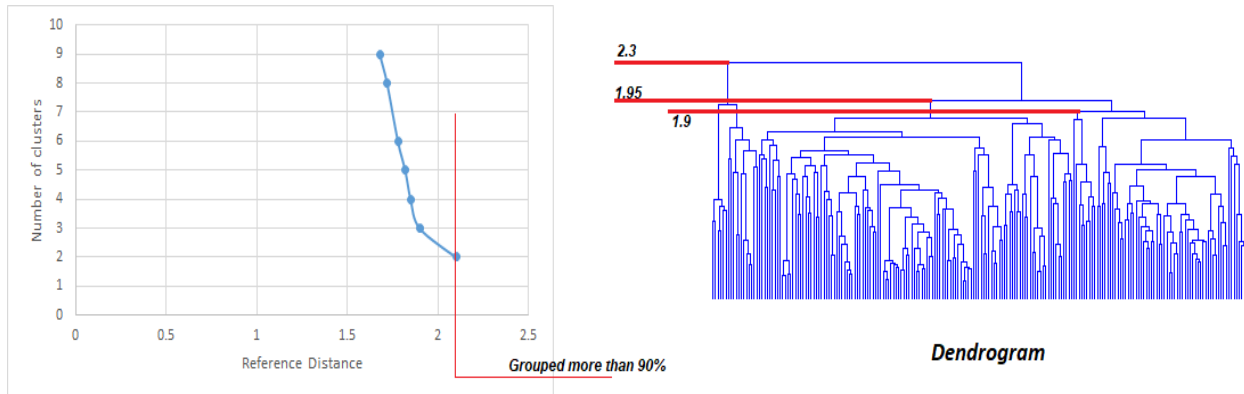


Figure 3-13: the quantity of clusters obtained for sonar with a reference distance $d = 2.1$, obtain two group

Table 3-7: Data obtained with the program of grouping for the analysis signal from a sonar

Reference Distance	Number of clusters	Number of items grouped in%
2.1	2	90%
1.9	3	83%
1.85	4	81%
1.82	5	78%
1.78	6	76%
1.72	8	74%
1.68	9	72%

Figure 3-14 shows the link between the values of RD and the quantity of clusters obtained and the percentage of clustered elements obtained for the analysis of a signal from a sonar.



Results obtained using the Euclidian distance

Figure 3-14: The relationship between the number of groups and the value of RD for a sonar dataset

Shown in Figure 3-13, Figure 3-14 and Table 3-7 that for a reference distance of less than 2.1 have a large number of dispersed elements (over 10%). For a reference distance of 2.1 give two groups and for greater distances to obtain all the group but fewer elements dispersed. Given that from the moment in which the group of more than 90% of the forms of input considers we have a right group so in this case, we should use an MLP with 2 hidden layers to analyze the signal from the sonar.

3.6.2.5. Conclusions

Using clustering techniques highlighted several common characteristics input forms, on which they (input forms) can be classified into groups (clusters).

A technical group made based on AHC algorithm using a tree with minimum distances. Before attempting to group forms of input they must be normalized operation which consists in determining the ranges in which they are expressed and reporting at this interval.

To calculate distances, you can use Euclidean distance. The group is given a reference distance to an object belonging to a group. It can be seen that for a short distance are obtained many classes, but also many elements that do not belong to classes (elements dispersed) as we grow the distance, we obtained fewer classes (fewer forms that cannot be grouped). It has been considered that when the group was carried out to at least 90% of the forms in the input database were grouped together, the results can be used for designing the structure of ANN. We felt that the number of groups that can be obtained is the optimal number (in terms of final error, the difference between the desired output and produced) of hidden layers required a multilayer neural network to solve the problem.

We analyzed with the same manner the total of six datasets, namely ECG signal component analysis (P wave, QRS complex, and the T wave), ECG signal analysis to diagnose some heart conditions, achieving classification of the Landsat Satellite images dataset and analyzing a signal from a sonar, they were chosen so as to provide a general nature of the analysis. Following this method, we conclude that 2 hidden layers are the optimum quantity for the P-wave signal and

Sonar signal datasets. ECG and QRS need only one single hidden layer and three hidden layers are the optimal number for T wave dataset to train a neural network. For Landsat Satellite images dataset, we need two hidden layers to train a neural network.

3.7. Experimental Results

In the following sub-chapter, we test each data set presented previously to identify the structure of the MLP. These datasets will be tested with various architectures multilayer neural networks using a supervised learning method in the particular backpropagation learning algorithm.

The first part of simulation and training MLP was done using simulation program “*Neuroshell ver. 2.0*” of Ward Systems Group company program [45]. The second part of simulation and training MLP was done using the platform “*Determine the NN Architecture*” is a third-party platform of Weka was developed and used.

For the analysis of datasets, we used three different architectures of the MLP, with one hidden layer, two hidden layers, and three hidden layers.

Figure 3-15 shows the structure of 3 architectures of MLP used.

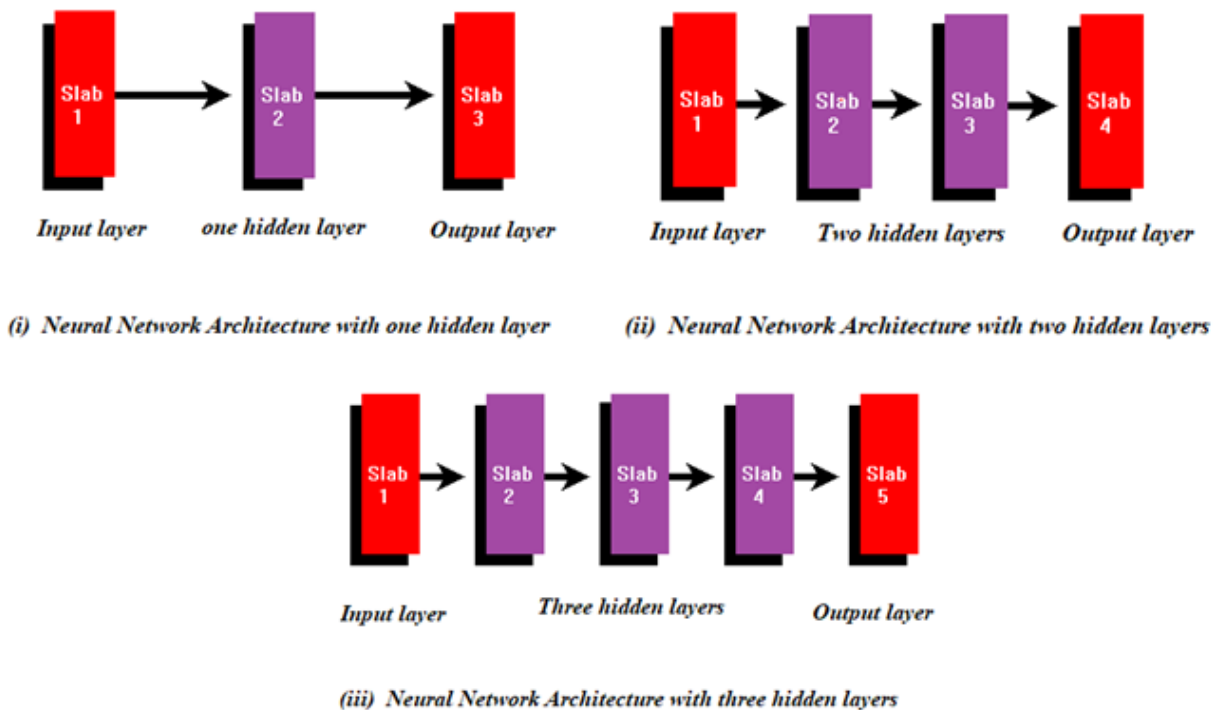


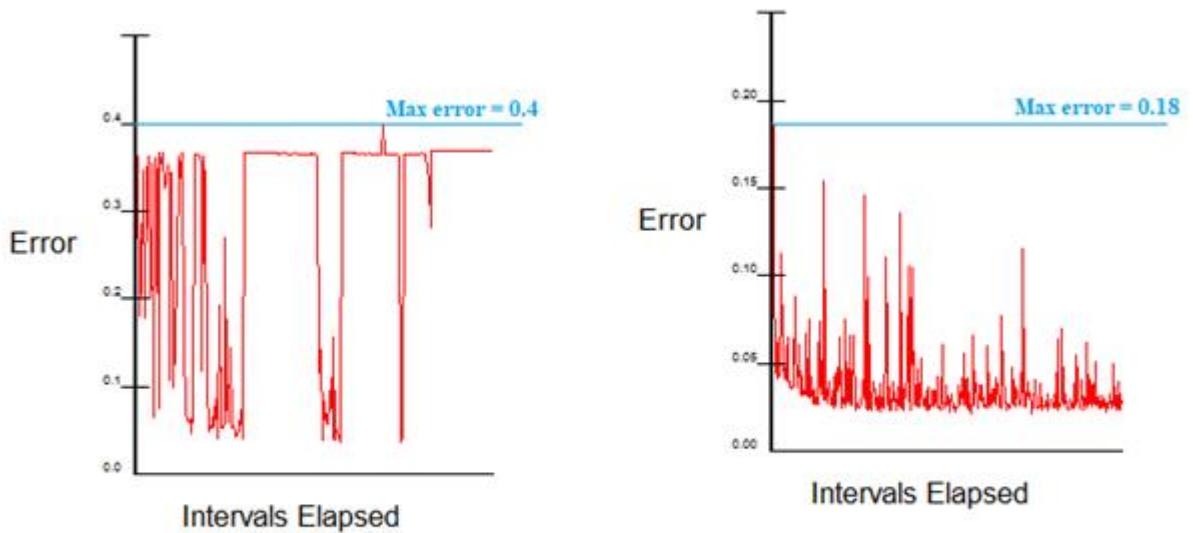
Figure 3-15 MLP architectures

To prove the validity of the projected clustering method to identify the optimum architecture of an MLP we try to compare the results of clustering method with the experimental results based on values of errors for the selected datasets.

3.7.1. Experimental results for clustering techniques

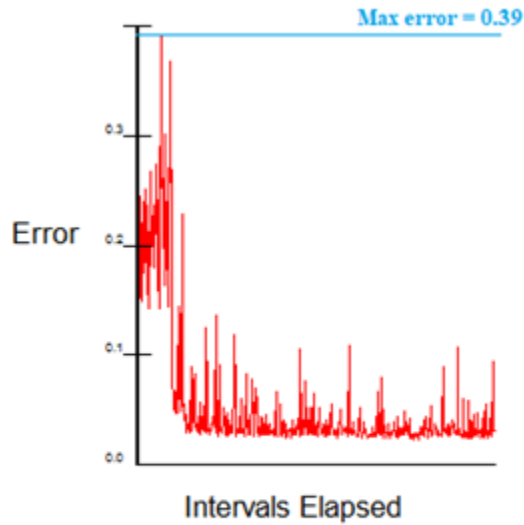
3.7.1.1. Experimental results for Landsat Satellite

Figure 3-16 shows a comparison of obtained errors obtained for Landsat data set using various quantity of hidden layers.



(i) The average error for Neural Network Architecture with one hidden layer

(ii) The average error for Neural Network Architecture with two hidden layers



(iii) The average error for Neural Network Architecture with three hidden layers

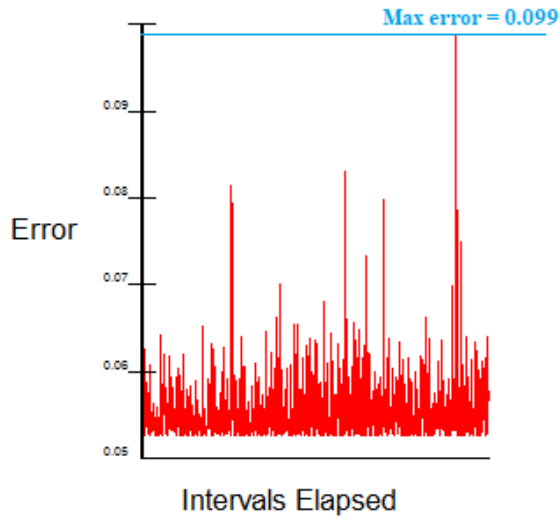
Figure 3-16 the relative error for the Landsat data set using various quantity of hidden layers

Based on the results obtained for an MLP architecture composed of 1 hidden layer we have a max error value equal to 0.30 while an MLP architecture contains two hidden layers has one answer with a larger error less than 0.18. The max error obtained for MLP architecture composed of three hidden layers is equal to 0.30 obtained for two times.

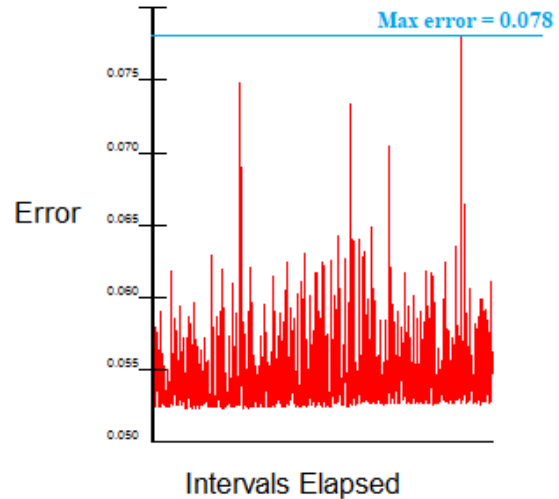
From the results of the analysis of Landsat Satellite images, we notice that the best performs is for an MLP architecture composed of two hidden layers. The results obtained confirm those obtained by clustering program that presented previously.

3.7.1.2. The experimental results for P-wave

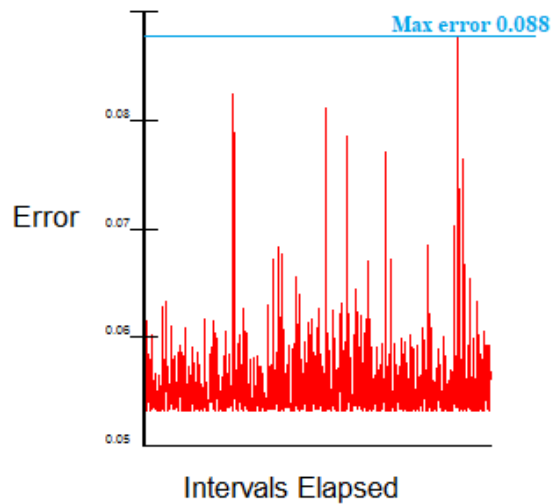
Figure 3-17 shows a comparison of errors for P wave dataset obtained by a different number of hidden layers.



(i) The average error for Neural Network Architecture with one hidden layer



(ii) The average error for Neural Network Architecture with two hidden layers



(iii) The average error for Neural Network Architecture with three hidden layers

Figure 3-17 the relative error for the P-wave using various quantity of hidden layers

Please note that for results of an MLP architecture composed of one hidden layer have four errors, which exceeded 0.08 with a max error equal to 0.099 and for MLP architecture composed of two hidden layers the value of errors obtained does not exceed 0.078. For MLP architecture composed of three hidden layers has three errors with a max value equal to 0.088.

From the results of the analysis of a P-wave signal, we notice that the best performs is for MLP architecture composed of two hidden layers.

The results obtained confirm those obtained by clustering program that presented previously.

3.7.1.3. Experimental results for QRS complex

Figure 3-18 shows a comparison of errors obtained for QRS signal dataset obtained by a different number of hidden layers.

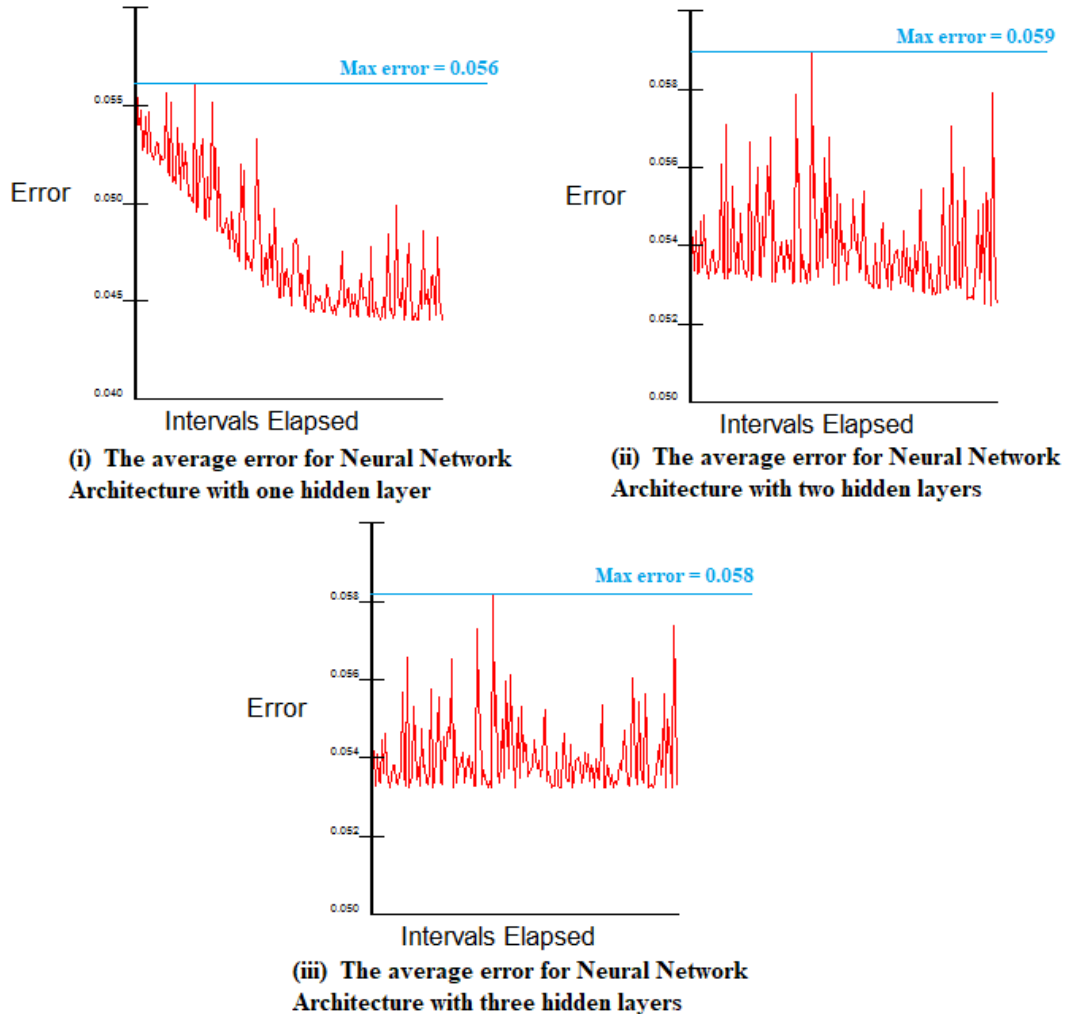


Figure 3-18 the relative error for the QRS signal using various quantity of hidden layers

From the above figure is seen quite clearly that the errors are less than 0.056 for three hidden layers neural network compared to the other neural network architectures in which errors are more than 0.057.

Analyzing graphs can be seen easily that the best results are obtained with an MLP architecture composed of one hidden layer and the weakest a network with two and three layers.

The results obtained confirm those obtained by clustering program that presented previously.

3.7.1.4. Experimental results for the T wave signal

Figure 3-19 shows a comparison of errors obtained for T-wave signal dataset obtained by a different number of hidden layers.

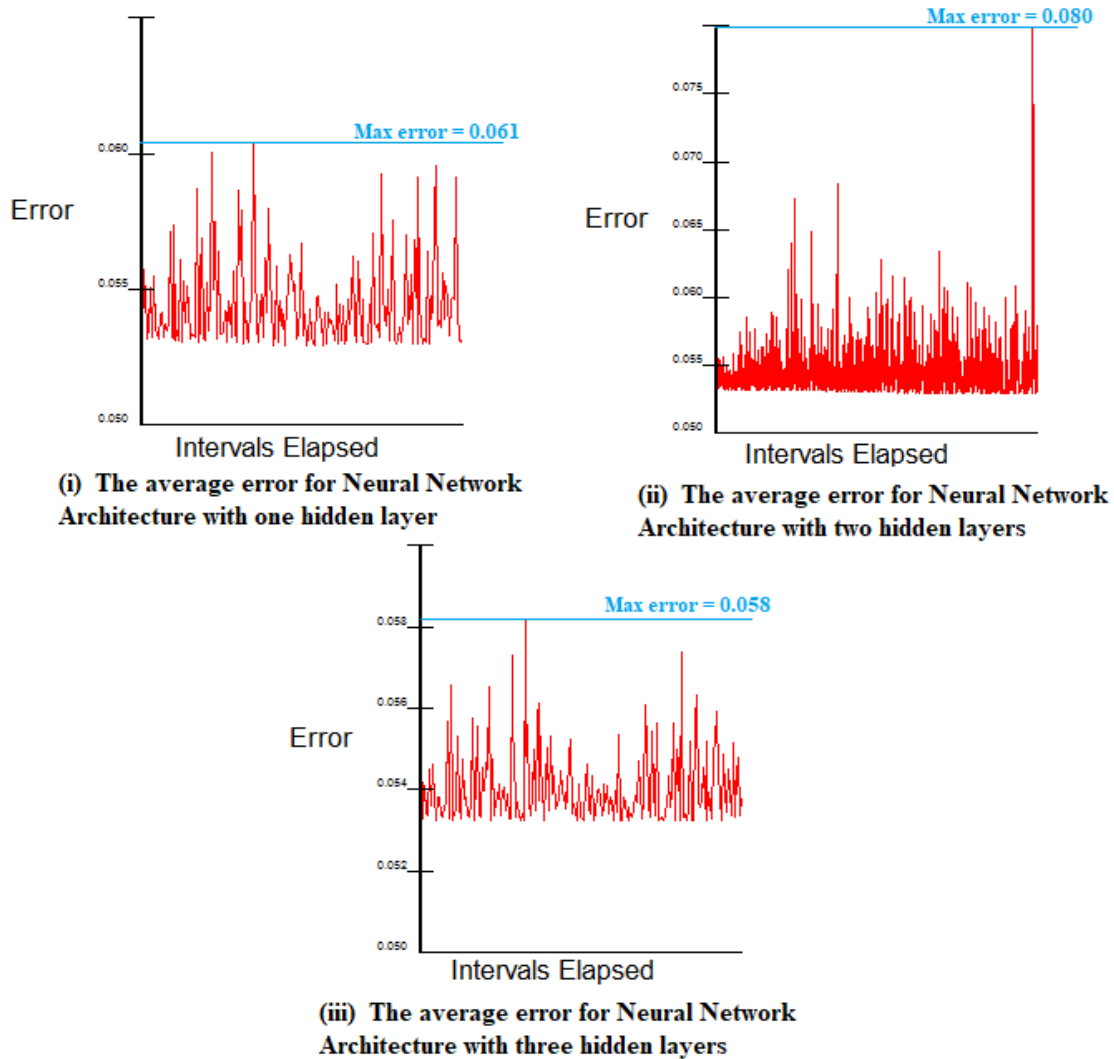


Figure 3-19 the relative error for the T-wave signal using various quantity of hidden layers

As observed from Figure 3-19 an MLP architecture composed of three hidden layers have the lowest value of errors.

Analyzing graphs can be seen easily that the best results are obtained with MLP architecture composed of three hidden layers. The weakest is MLP architecture composed of one and two hidden layers.

The results obtained confirm those obtained by clustering program that presented previously.

3.7.1.5. Experimental results for a diagnostic ECG signal

Figure 3-20 shows a comparison of errors obtained for ECG signal dataset obtained by a different number of hidden layers.

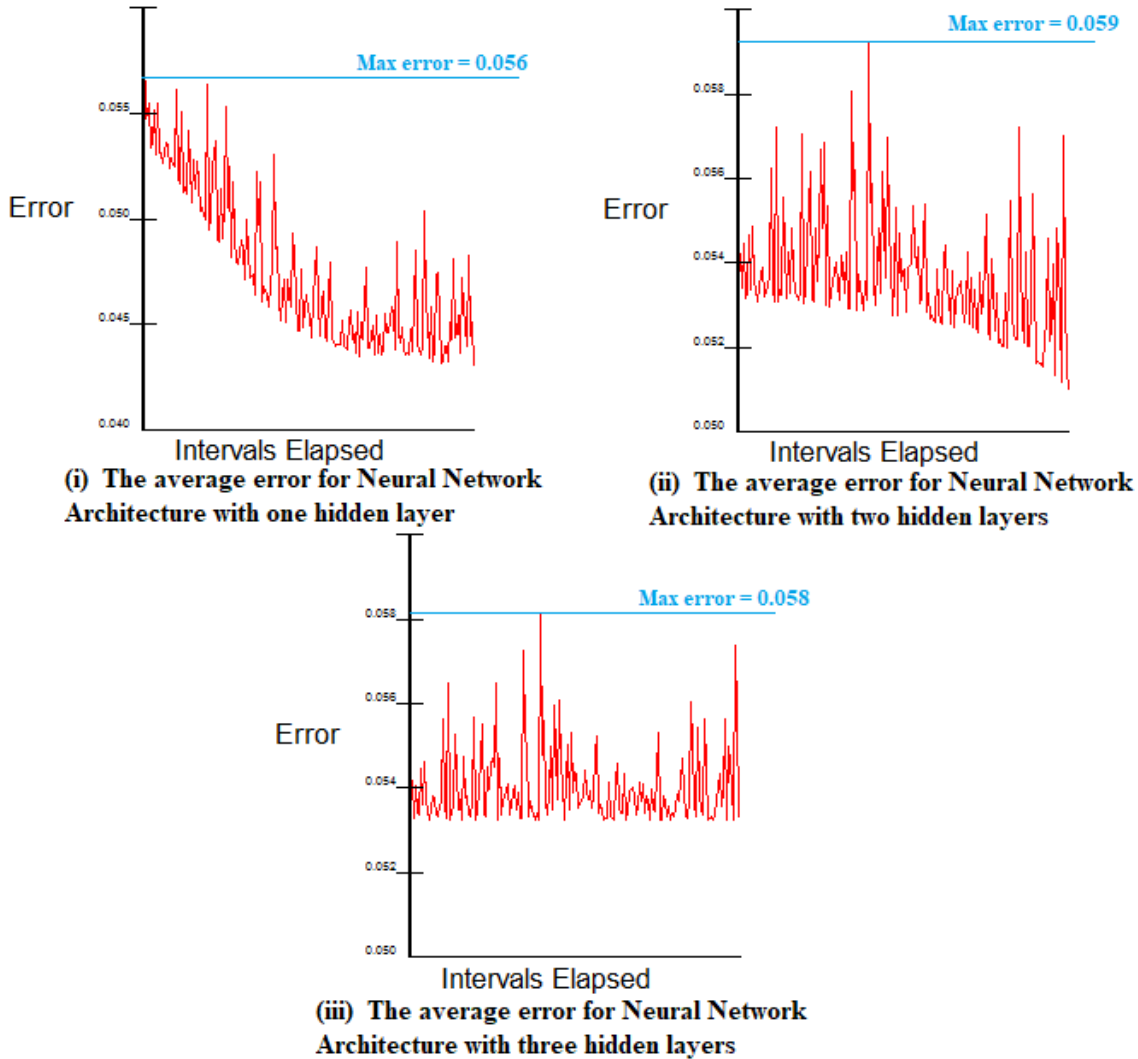


Figure 3-20 the relative error for the ECG signal using various quantity of hidden layers

As observed from Figure 3-20 that the value errors are less than 0.056 for an MLP architecture composed of one hidden layer compared to the other MLP architecture in which errors are more than 0.057.

Therefore, the dataset ECG signal needs MLP architecture composed of one hidden layer as optimum architecture, while MLP architecture composed of two and three layers have the weakest results.

The results obtained confirm those obtained by clustering program that presented previously.

3.7.1.6. Experimental results for the analysis of a signal from a sonar

Figure 3-21 shows a comparison of errors obtained for Sonar signal dataset obtained by a different number of hidden layers.

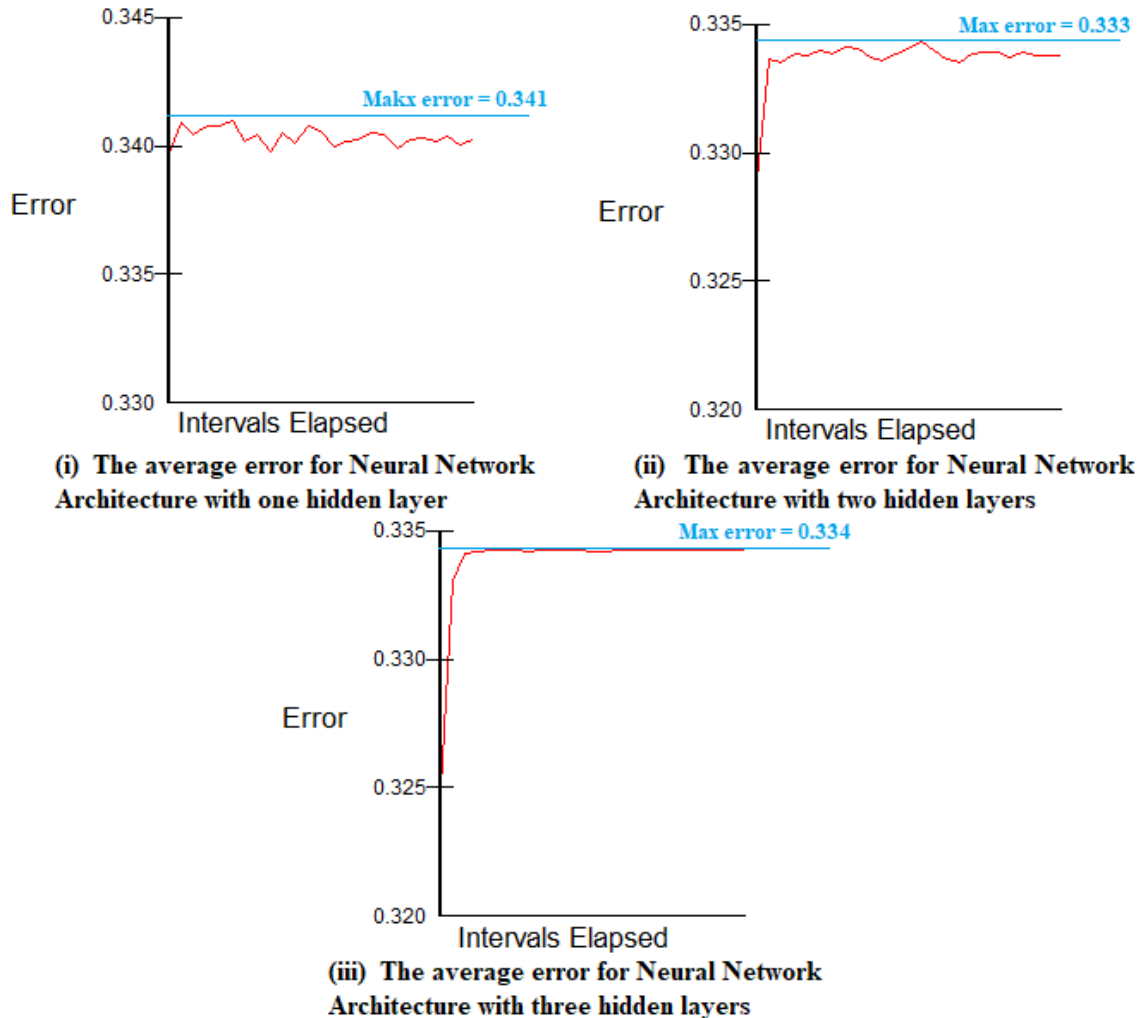


Figure 3-21 the relative error for the Sonar signal using various quantity of hidden layers

As observed from Figure 3-21 an MLP architecture composed of two hidden layers have the lowest value of errors.

Analyzing graphs can be seen easily that the best results are obtained with MLP architecture composed of two hidden layers and the weakest MLP architecture composed of one and three layers.

Table 3-8: comparison between the best quantity of hidden layers obtained by different MLP architectures and the number of clusters obtained by clustering method for all datasets.

Data Set	Number of clusters using grouping program	Best number of layers obtained through experimental results of different neural network architecture using Neuroshell
Landsat Satellite	2	2
P-wave	2	2
QRS	1	1
T-wave	3	3
ECG	1	1
Sonar	2	2

From Table 3-8 the experimental results obtained confirm those obtained by clustering program that presented previously which proves the validity of the proposed method.

3.8. Comparative Study of Clustering Distance Measures to identify the MLP Structure

The clustering method used to determine the structure of the MLP can be considered a general method where it depends on a set of pattern recognition techniques and based on defined criteria to achieve a perfect prediction of the quantity of hidden layers/units [98] [99] [100] [101]. Among pattern recognition techniques to discover data and reveal patterns, we select the clustering algorithm. The proposed method depends on the complexity of the considered problem to identify the structure of the MLP. The complexity level of the problem can be predicted from the learning data of the MLP. The pattern discovered from the dataset will be evaluated to identify the complexity of the problem. In general, the quantity of hidden layers/units is affected by the complexity of the problem to be solved [102], and also the structure of MLP affect the generalization capabilities of the MLP [103]. Therefore, by grouping the given learning set, the overall quantity classes identified represent a neural network's optimum hidden layers' number based on given conditions. The point of this research is to determine the suitable criteria to correlate the number of the obtained cluster with the optimal number of hidden layers, which requires a perfect selection of distance measures through a comparison review [104].

This section represents linkage clustering. Sample of clustering include; complete-linkage clustering, single-linkage clustering, average distance, and input forms figures higher or lower than mean range. They must be put into consideration in designing ANN that needs relative consideration grouping measurements. As per these requirements, MLP ANN's hidden layers can be identified by the number of total clusters.

3.8.1. Distance Measures

Using the proposed clustering method, the accuracy of results depends on the accuracy of the cluster analysis. A comparative study of clustering distance measures is required to identify the suitable distance measure to obtain the optimal structure of an MLP neural network. To calculate the distance between observations we select Manhattan and Euclidean distance metrics. In addition, the distance between objects is computed using the linkage function to create clusters. We select the widely used linkage methods for the AHC Algorithm such as complete-linkage clustering, single-linkage clustering and mean distance clustering.

3.8.1.1. Influence of distance measure on the AHC Algorithm results

AHC Algorithm performs grouping of sets of data by adopting each element in a set of data as an independent cluster continuously joining them till a single cluster is obtained. AHC creates a Dendrogram composed of nested clusters. Considering the distance, the dendrogram will be divided so as to meet the desired clusters. The dividing range is called "Reference Distance". Therefore, the quantity cluster will depend on "reference distance". So that shorter clusters will be combined.

Due to the different types of patterns in datasets, it is hard to decide which linkage method is suitable. This represents one of the disadvantages of AHC Algorithm. For example, there are types of patterns not compatible with single linkage however we can obtain good results using average and complete linkage methods [105]. a comparative study of clustering distance measures can be a solution to define the suitable distance measure.

When clustering distance-measure is considered, the general ability of the projected method is improved. More support is given to training data.

3.8.1.2. Clustering Distance Measure Techniques Used

clustering algorithms depends heavily on the distance measure to cluster data. The widely used clustering algorithms rely on the distance measures to reveal patterns in data such as k-means and k-medoids clustering algorithms. Each data type needs a specific distance measure to reveal patterns from data [106]. In this study, several distance measures are selected for evaluation. The following distance measures are chosen to be used to accomplish the comparative study:

(1) *Euclidean distance*

For numerical data, a popular distance adopted is "Euclidean Distance". It helps in the analysis intact and differentiated clusters [107] [108]. the Euclidean distance will be the best choice for this type of data. Euclidean distance has a disadvantage because of the high sensitivity to noise. However Euclidean distance is extremely used for clustering data. The Euclidean distance formula sums the results obtained from the square of the distance between each variable, then finding the

square root of that sum. The formula for this distance between a point X (X1, X2, ...) and a point Y (Y1, Y2, ...) is:

$$d = \sqrt{\sum_{j=1}^n (x_j - y_j)^2} \quad (3.1)$$

(2) **Manhattan Distance**

For discrete data, it is suitable to use Manhattan Distance. The sensitivity to outliers is the most important problem of Manhattan distance. Manhattan Distance calculate distances by summing the absolute distances of each element of the dataset. The formula for this distance between a point X=(X1, X2 ...) and a point Y= (Y1, Y2 ...) with a number of n variables is:

$$d = \sum_{i=1}^n |x_i - y_i| \quad (3.2)$$

(3) **AHC Techniques**

Linkage methods are used to calculate the distance between clusters and other elements of the dataset. the most successful linkage methods used for Hierarchical Agglomeration Clustering techniques are single, complete, and average linkages. The single linkage method depends on the proximity between the two closest elements included in two different clusters to determine the proximity between these two clusters. Complete linkage method depends on the proximity between two most distant elements of two different clusters to determine the proximity between these two clusters. Average linkage method depends on the average of pairwise proximity between elements included in two different clusters to determine the proximity between these two clusters.

a) Single Linkage

The distance between the two closest elements included in two different clusters (Ci, Cj) is the distance between these two clusters.

$$d_{min}(C_i, C_j) = \min d(p, q) \quad (3.3)$$

$$p \in C_i, q \in C_j$$

This figure represents an example of a single linkage:

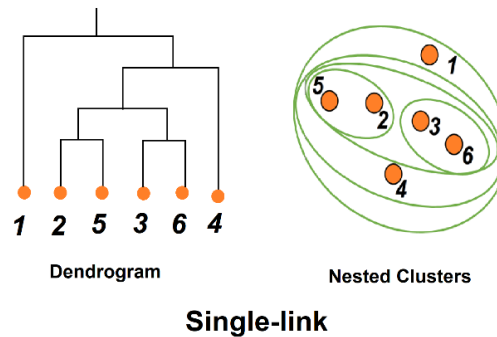


Figure 3-22: Example of Single linkage Clustering

b) Complete linkage

The distance between the two most distant elements of two different clusters (C_i, C_j) is the distance between these two clusters.

$$d_{min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q) \quad (3.4)$$

This figure represents an example of complete linkage:

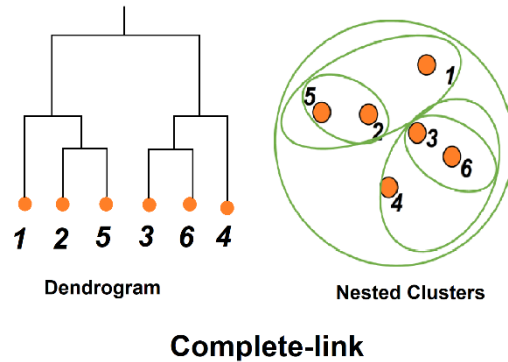


Figure 3-23: Example of Complete linkage Clustering

c) Average linkage

The average of the pairwise distance between elements included in two different clusters (C_i, C_j) is the distance between these two clusters.

$$d_{min}(C_i, C_j) = \text{avg } d(p, q) \quad (3.5)$$

$$p \in C_i, q \in C_j$$

Figure 3-24 represents an example of Average linkage:

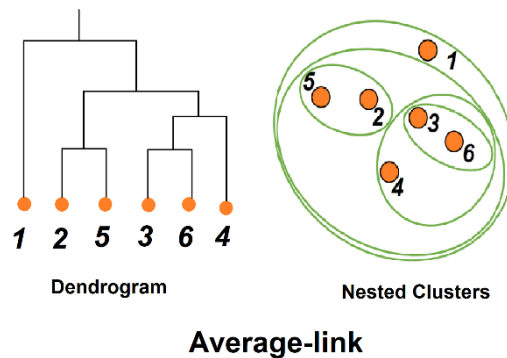


Figure 3-24: Example of Average linkage Clustering

d) Normalization

Data normalization is important for data clustering and especially it has a positive influence when using distance measures. The normalization method must be carefully chosen because it has a strong impact on the clustering results. Normalization helps to restructure data under a series of so-called normal forms to minimize data redundancy and optimize integrity [109] [110].

In this chapter, we select the Min-Mix and the Z-score Normalization techniques. The Min-Mix Normalization technique performs a linear alteration on the original data by transforming the data from measuring units to a new interval. Min-Mix Normalization of a feature F in a range $[\min A, \max A]$ gives a new range $[\text{new_min}A, \text{new_max}A]$ as per given formula:

$$v' = \frac{v - \min_F}{\max_F - \min_F} (\text{new_max}_F - \text{new_min}_F) + \text{new_min}_F \quad (3.6)$$

where v is the current value of feature F

The Z-score Normalization technique transforms the data by converting the values to a common scale with an average of zero and a standard deviation of one. A value v of A is normalized to v' by using this formula:

$$v' = \frac{v - \bar{F}}{\sigma_F} \quad (3.7)$$

Where F and σF are the mean and standard deviation of feature F respectively.

e) Contribution of the Study

To obtain the optimal MLP neural network it is necessary to determine the convenient distance measure suitable for the grouping method used which requires a perfect selection of distance measures through a comparison review. The training dataset will be clustered based on different distance measure techniques until we define the best distance measure for this dataset. Since the number of clusters obtained affect the structure of the MLP neural network so the accuracy of clustering the training dataset has an important influence on the results obtained. In general, it is very difficult to define a particular distance compatible with all types of training datasets which requires a call for a relative analysis of the measures per data type.

3.8.2. Trial results for choosing the best distance measure to the clustering method

The selected distance metrics (Manhattan, Euclidean) and widely used linkage techniques (Minimum, Maximum, average linkage) will be used to in several case study until we can evaluate the clustering distance measures in order to identify the optimum quantity of clusters.

We developed a third party platform of the data mining open source program called “Weka” which helped in accuracy classification, error checking and clustering. ANN structure is assessed according to accuracy and error checking.

3.8.2.1. First case study

Case study frame:

Table 3-9 First case study

Normalization	Clusters Distance	Distance Metrics
Standard	Complete linkage	Euclidean

Figure 3-25 shows the cut of Dendrogram for a value of RD $d = 0.376$ selected based on the criterion described above for the training dataset Glass Identification and the corresponding quantity of groups obtained.

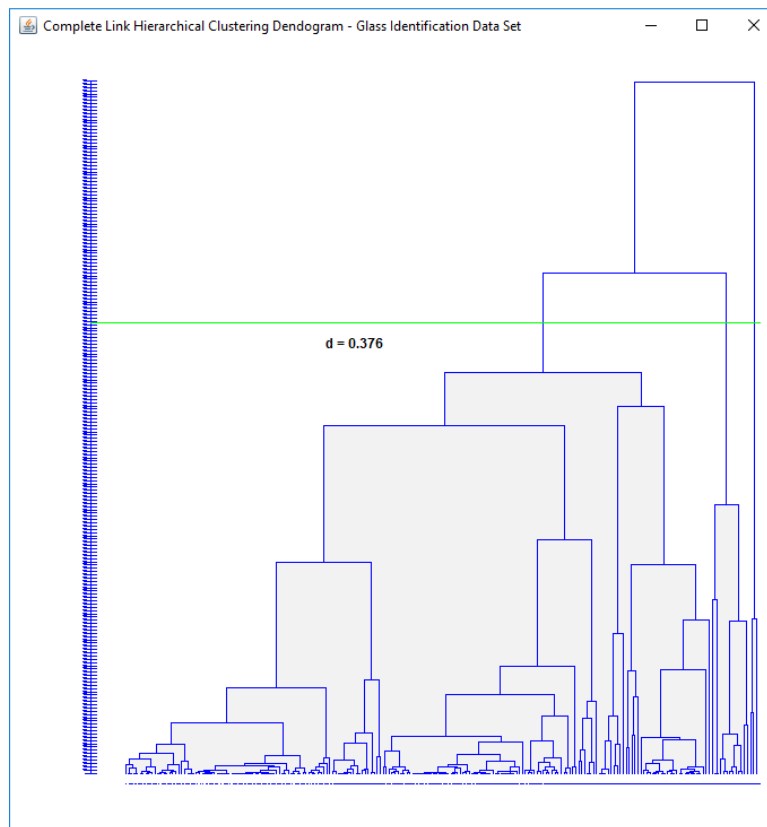


Figure 3-25: Complete linkage Clustering (Euclidean) [111]

Table 3-10 shows results obtained of accuracy classification, error checking, and period covered obtained as per the study. Hidden layers equate cluster numbers by grouping a training

dataset as per the described conditions. By adopting the clustering technique, we can derive at total clusters. Using rules of thumb, one can gauge hidden units in a specific hidden layer.

Table 3-10: The accuracy and error values [111]

Formula number	Number of hidden layers	Hidden neurons per layer	Accuracy [%]	Error / Epoch
(2.4)	2 Hidden Layers	8	76.1682	0.0419012
(2.5)		3	67.2897	0.0595847
(2.6)		10	80.5611	0.0430242
(2.7)		1	62.1495	0.0853624
(2.8)		31	78.972	0.0375195
(2.9)		23	81.7757	0.0365077
(2.4)	3 Hidden Layers	8	68.2243	0.0550806
(2.5)		3	64.0187	0.0724355
(2.6)		10	67.757	0.0514105
(2.7)		1	35.514	0.1064695
(2.8)		31	71.9626	0.0506866
(2.9)		23	75.7009	0.0505723
(2.4)	6 Hidden Layers	8	35.514	0.1064742
(2.5)		3	35.514	0.1064728
(2.6)		10	35.514	0.1064745
(2.7)		1	35.514	0.1064695
(2.8)		31	35.514	0.1064745
(2.9)		23	35.514	0.1064745

Figure 3-26 presents the chart of values of error per epoch for 2, 3 and 6 hidden layers and for a different number of hidden units (8, 3, 10, 1, 31, 23):

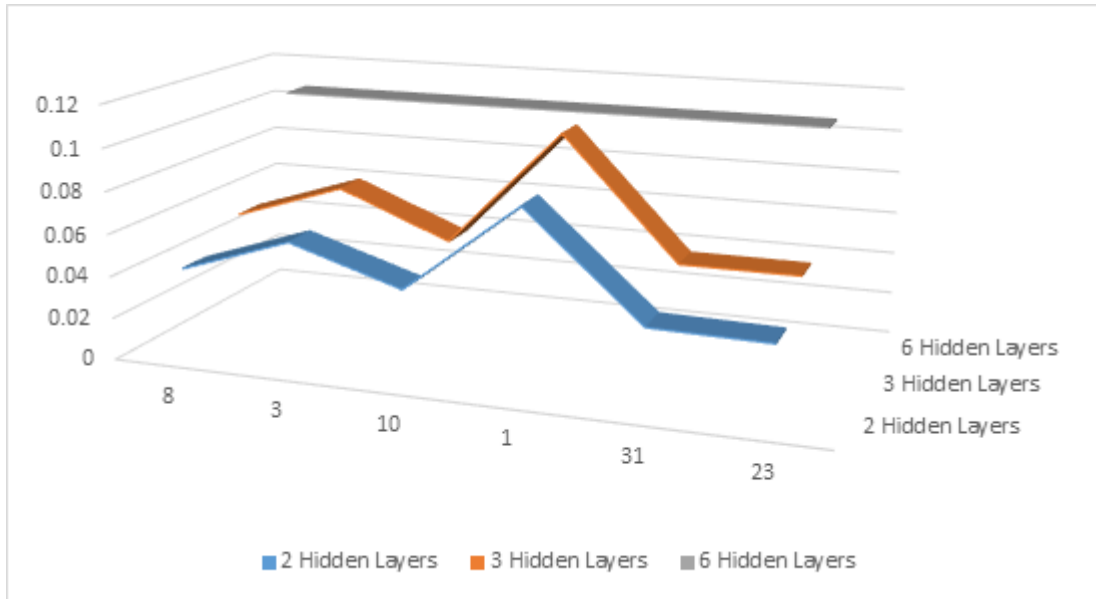


Figure 3-26: Errors for various quantity of hidden layers and hidden units for the first case study [111]

As observed from Table 3-10, for two hidden layers we have the max percentage of classification accuracy which equals to 81.77%. We have the minimum value of error per epoch equal to 0.0365077 obtained for the quantity of hidden layers equivalent to two.

The quantity of hidden layers is identified by using the proposed clustering method. Therefore, we can compare the result of the various quantity of hidden units to define the best quantity.

Figure 3-27 shows the error indicator for various quantity hidden units:

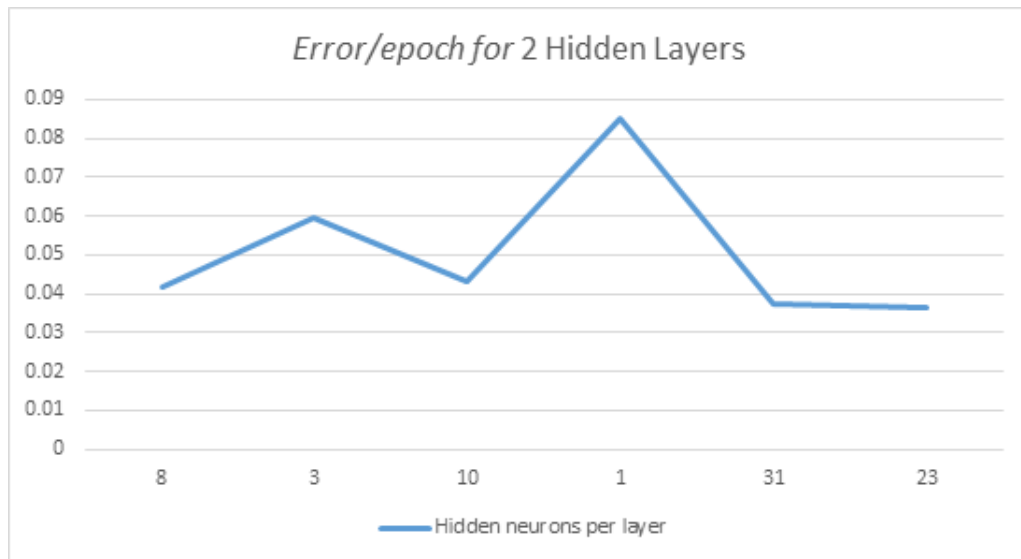


Figure 3-27: the error indicator for various quantity hidden units [111]

From Figure 3-27, it is seen that the lowest value of error per epoch is for 23 hidden units.

Therefore, the best structure for the current case study must have two hidden layers and 23 hidden units for each layer for the training dataset Glass Identification based on the results obtained.

3.8.2.2. Second Case Study

Case study frame:

Table 3-11 Second Case Study

Normalization	Clusters Distance	Distance Metrics
Standard	Complete linkage	Manhattan

Table 3-12 Table 3-10 indicate results from accuracy classification and error checking in relation to the present study.

Table 3-12: The accuracy and error values [111]

Formula number	Number of hidden layers	Hidden neurons per layer	Accuracy [%]	Error / Epoch
(2.4)	2 Hidden Layers	8	76.1682	0.0419012
(2.5)		3	67.2897	0.0595847
(2.6)		10	80.5611	0.0430242
(2.7)		1	62.1495	0.0853624
(2.8)		31	78.972	0.0375195
(2.9)		23	81.7757	0.0365077
(2.4)	4 Hidden Layers	8	35.514	0.1064742
(2.5)		3	35.514	0.1064728
(2.6)		10	35.514	0.1064743
(2.7)		1	35.514	0.1064695
(2.8)		31	35.514	0.1064739
(2.9)		23	35.514	0.1064742
(2.4)	7 Hidden Layers	8	35.514	0.1064743
(2.5)		3	35.514	0.1064728
(2.6)		10	35.514	0.1064744
(2.7)		1	35.514	0.1064695
(2.8)		31	35.514	0.1064744
(2.9)		23	35.514	0.1064746

Figure 3-28 presents the chart of values of error per epoch for 2, 4 and 7 hidden layers and for a different number of hidden units (3, 10, 31, 23):

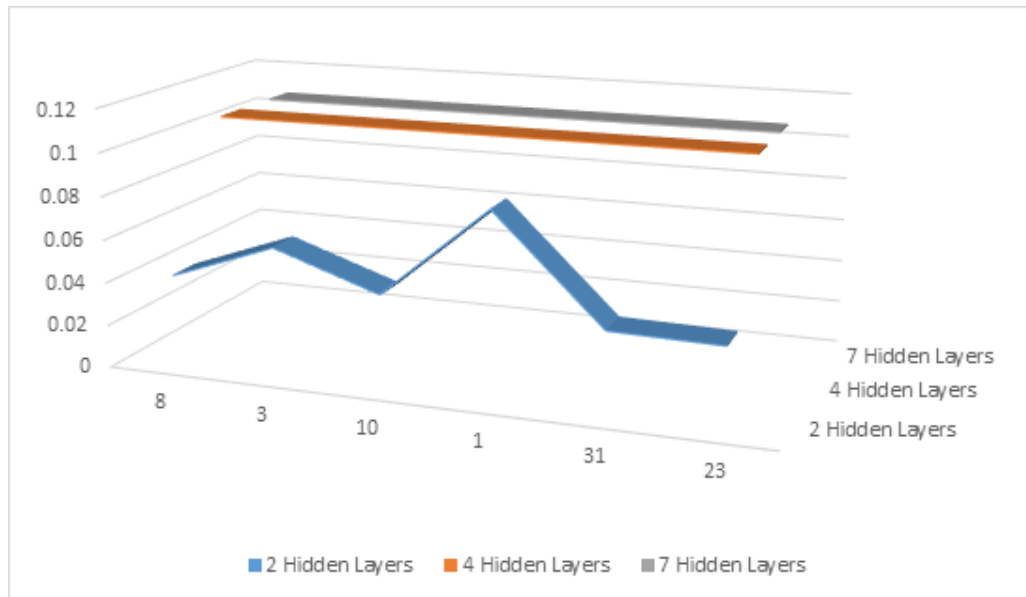


Figure 3-28: Errors for various quantity of hidden layers and hidden units for the second case study

As observed from Table 3-12 and Figure 3-28, the optimum quantity of hidden layers is 2 for the second case study.

3.8.2.3. Third Case Study

Case study frame:

Table 3-13 Third Case Study

Normalization	Clusters Distance	Distance Metrics
Standard	Average linkage	Manhattan

Figure 3-29 shows the obtained Dendrogram and the corresponding quantity of groups obtained according to the value of RD $d = 0.442$ for the third case study for Glass Dataset.

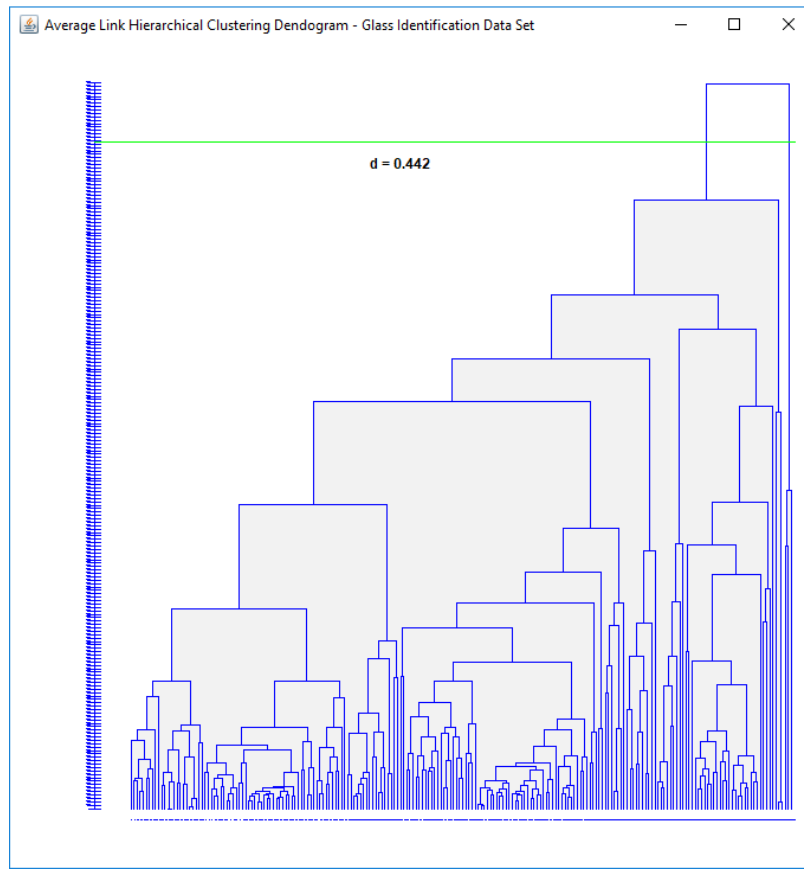


Figure 3-29: Average Linkage Clustering (Manhattan)

Figure 3-30 shows the errors of 2 and 3 hidden layers and various quantity of hidden units:

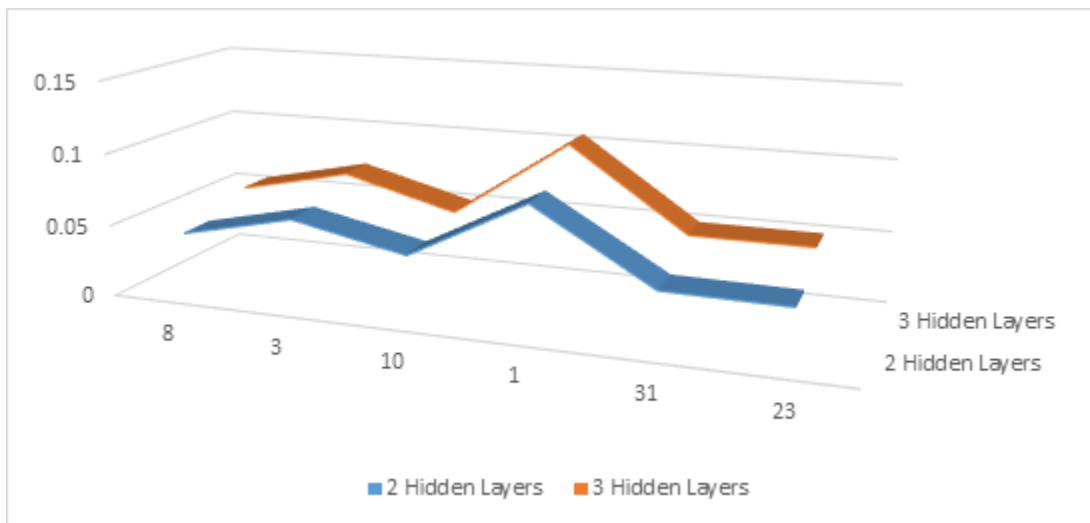


Figure 3-30: Errors for various quantity of hidden layers and hidden units for the third case study [111]

As observed in Figure 3-30, the best structure for the current case study must have two hidden layers.

3.8.2.4. Fourth Case Study

Case study frame:

Table 3-14 Fourth Case Study

Normalization	Clusters Distance	Distance Metrics
Standard	Single linkage	Manhattan

Figure 3-31 shows the cut of Dendrogram for a value of Reference Distance $d = 0.192$ selected based on the criterion described above for the training dataset Glass Identification and the corresponding quantity of groups obtained.

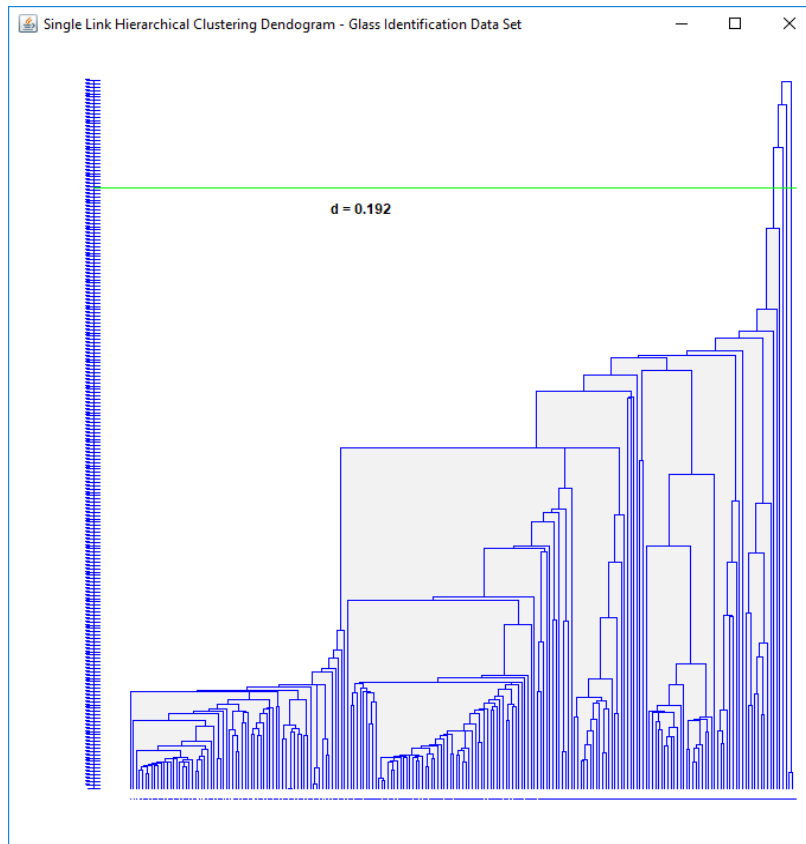


Figure 3-31: Single Linkage Clustering (Manhattan)

Table 3-15 shows results obtained of accuracy classification, errors checking, and period covered obtained as per the fourth study.

Table 3-15: The accuracy and error values [111]

Formula number	Number of hidden layers	Hidden neurons per layer	Accuracy [%]	Error / Epoch
(2.4)	4 Hidden Layers	8	35.514	0.1064742
(2.5)		3	35.514	0.1064728
(2.6)		10	35.514	0.1064743
(2.7)		1	35.514	0.1064695
(2.8)		31	35.514	0.1064739
(2.9)		23	35.514	0.1064742
(2.4)	5 Hidden Layers	8	35.514	0.1064742
(2.5)		3	35.514	0.1064728
(2.6)		10	35.514	0.1064743
(2.7)		1	35.514	0.1064696
(2.8)		31	35.514	0.1064743
(2.9)		23	35.514	0.1064745

Figure 3-32 presents the chart of values of error per epoch for 4 and 5 hidden layers and for a different number of hidden units (8, 3, 10, 1, 31, 23):

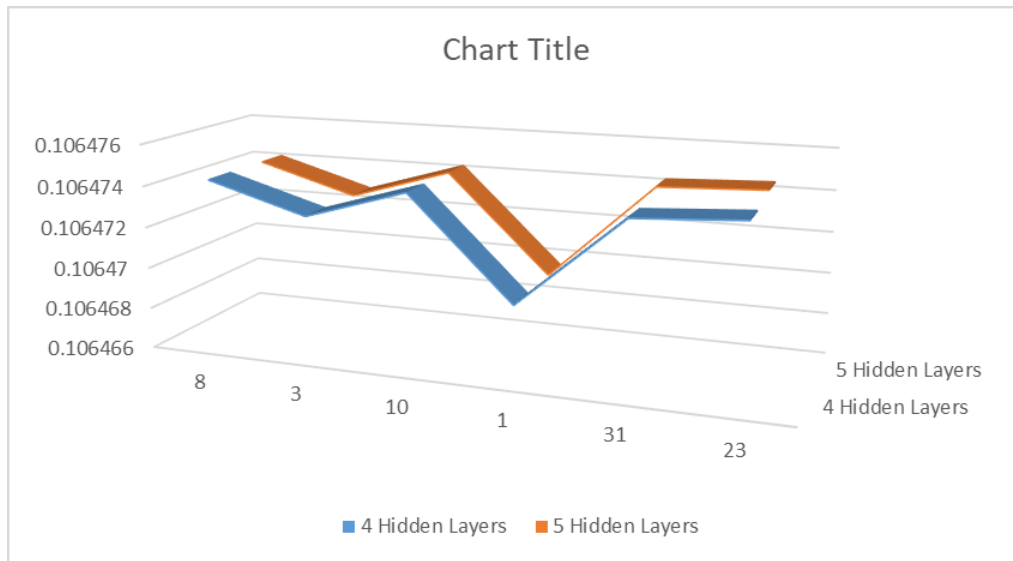


Figure 3-32: Errors for various quantity of hidden layers and hidden units for the fourth case study [111]

As observed in Table 3-15, for four hidden layers we have the max percentage of classification accuracy which equals to 35.51%. We have the minimum value of error per epoch equal to 0.1064695 obtained for the quantity of hidden layers equal to two. The best quantity of hidden layers for this fourth study is four.

The quantity of hidden layers is obtained by using the projected clustering method. Therefore, we can compare the result of the various quantity of hidden units to define the best quantity.

Figure 3-33 shows the error indicator for various quantity hidden units:

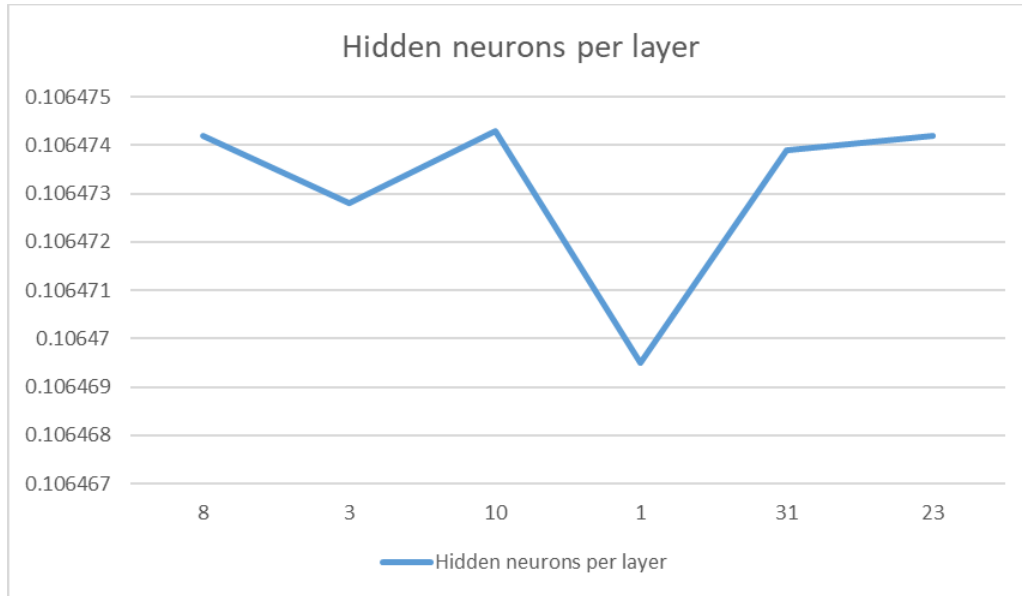


Figure 3-33: the error indicator for various quantity hidden units [111]

From Figure 3-33, it is seen that the lowest error value is for 1 hidden units and 4 hidden layers.

Therefore, the best structure for the current case study must have four hidden layers and 1 hidden unit for each layer for the training dataset Glass Identification based on the results obtained.

3.8.2.5. Discussion of Obtained Results

The case study with these parameters: Maximum Linkage, Euclidean distance, and standard Normalization has the best results compared to other case studies. Based on this case study the best MLP structure have two hidden layers and 23 hidden units for the training dataset Glass Identification. The best percentage of classification accuracy obtain for this structure is 81.77% and have the minimum value of error per epoch equal to 0.0365077.

Based on the previous case studies we obtain different results through changing the clustering algorithm parameters such as type of distance, clusters distance and normalization type while the case study Maximum Linkage, Euclidean distance, and standard Normalization achieved the best results as a percentage of classification accuracy and value of error per epoch.

3.9. Experimental results using “*Determine the NN Architecture*” platform

The second case study based on an experiment that uses the platform “*Determine the NN Architecture*” divided into three parts:

- The first part implements of the AHC algorithm, the quantity of groups obtained based on a specific cut of the obtained Dendrogram will be the quantity of hidden layers for an MLP.
- Part two is for calculating the quantity of hidden units.
- Third part uses the information obtained from the other two components of the application and performs a graphical display of the neural network.

Case studies made using the six datasets defined previously. The Dendrogram was generated and the number of clusters was determined, the number of neurons on the hidden layer was calculated using the first six formulas presented in Chapter 2.

For the AHC algorithm, various parameters such as the type of normalization, the type of distance and the distance between the clusters have been set.

Using the information obtained from the Dendrogram and the number of neurons calculated using the six formulas presented previously, the platform “*Determine the NN Architecture*” generate the architecture of neural network to determine the Error/epoch of the neural network. The learning rate was set to 0.3 and the number of epochs equal to 500.

An example of error/epoch calculation for a 2 hidden-layers neural network and 14 neurons on each hidden layer is shown in Figure 3-34.

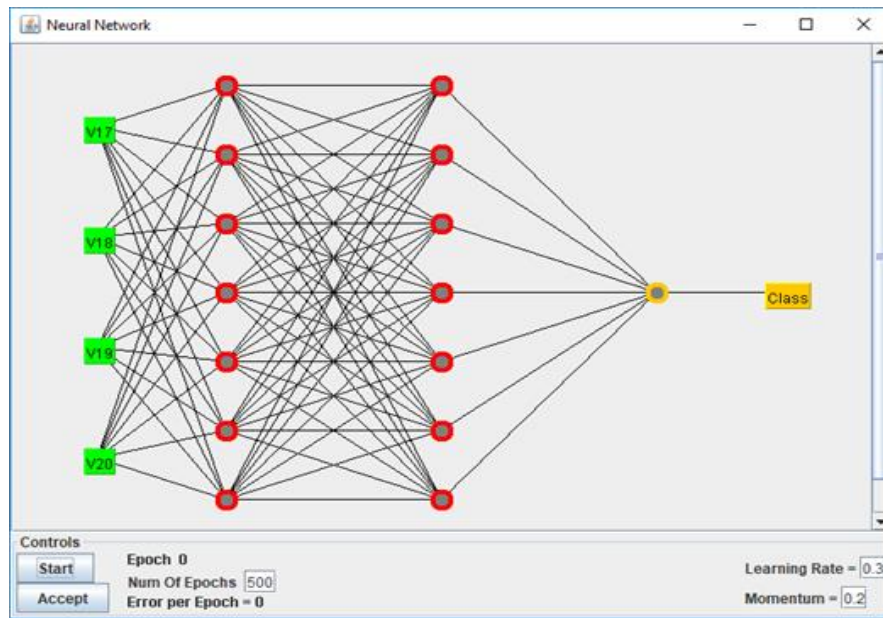


Figure 3-34 Neural Network architecture

The second case study based on an experiment that uses the platform “Determine the NN Architecture” to define the quantity of hidden units through evaluating the percentage of the classification accuracy and the value of error per epoch using formulas (2.4)-(2.9) defined previously.

3.9.1. Experimental results for Landsat Satellite

Table 3-16 presents results obtained for Landsat Satellite dataset.

Table 3-16 accuracy and error/epoch values based on formulas (2.4)-(2.9) for Landsat Satellite dataset

Dataset	Formula number	Number hidden layer	Hidden neurons per layer	Accuracy [%]	Error / Epoch
Landsat Satellite	(2.4)	2	2	76.7535	0.046556218
	(2.5)	2	1	50.3006	0.087793818
	(2.6)	2	3	80.5611	0.037348209
	(2.7)	2	2	76.7535	0.046556218
	(2.8)	2	65	82.5651	0.028518718
	(2.9)	2	13	82.3647	0.028037136

3.9.2. The experimental results for P-wave

Table 3-17 presents results obtained for the p-wave dataset.

Table 3-17 accuracy and error/epoch values based on formulas (2.4)-(2.9) for p-wave dataset

Dataset	Formula number	Number hidden layer	Hidden neurons per layer	Accuracy [%]	Error / Epoch
P-wave	(2.4)	2	4	53.7611	0.042172181
	(2.5)	2	1	53.9823	0.042182490
	(2.6)	2	6	53.9823	0.042169481
	(2.7)	2	1	53.9823	0.042182490
	(2.8)	2	117	53.9823	0.0422299
	(2.9)	2	15	54.2035	0.042180509

3.9.3. Experimental results for QRS complex

Table 3-18 presents results obtained for QRS dataset.

Table 3-18 accuracy and error/epoch values based on formulas (2.4)-(2.9) for QRS dataset

Dataset	Formula number	Number hidden layer	Hidden neurons per layer	Accuracy [%]	Error / Epoch
QRS	(2.4)	1	9	59.0708	0.035101472
	(2.5)	1	3	58.8496	0.035733663
	(2.6)	1	12	58.8496	0.035060290
	(2.7)	1	2	59.5133	0.036108963
	(2.8)	1	234	59.9558	0.035463254
	(2.9)	1	30	58.8496	0.035133572

3.9.4. Experimental results for the T wave signal

Table 3-19 presents results obtained for the T-wave dataset.

Table 3-19 accuracy and error/epoch values based on formulas (2.4)-(2.9) for the T-wave dataset

Dataset	Formula number	Number hidden layer	Hidden neurons per layer	Accuracy [%]	Error / Epoch
T-wave	(2.4)	3	3	53.9823	0.041593409
	(2.5)	3	1	54.2035	0.042493381
	(2.6)	3	4	53.9823	0.041322463
	(2.7)	3	2/3	-	-
	(2.8)	3	78	56.1947	0.039277418
	(2.9)	3	10	56.6372	0.039284545

3.9.5. Experimental results for a diagnostic ECG signal

Table 3-20 presents results obtained for ECG dataset.

Table 3-20 accuracy and error/epoch values based on formulas (2.4)-(2.9) for ECG dataset

Dataset	Formula number	Number hidden layer	Hidden neurons per layer	Accuracy [%]	Error / Epoch
ECG	(2.4)	1	11	57.9646	0.031626881
	(2.5)	1	3	59.292	0.034692581
	(2.6)	1	14	59.292	0.031409209
	(2.7)	1	2	60.8407	0.035248990
	(2.8)	1	89	59.292	0.031574463
	(2.9)	1	32	59.292	0.030975954

3.9.6. Experimental results for the analysis of a signal from a sonar

Table 3-21 presents results obtained for Sonar dataset.

Table 3-21 accuracy and error/epoch values based on formulas (2.4)-(2.9) for Sonar dataset

Dataset	Formula number	Number hidden layer	Hidden neurons per layer	Accuracy [%]	Error / Epoch
Sonar	(2.4)	2	15	81.25	0.014593654
	(2.5)	2	2	80.76	0.035628836
	(2.6)	2	20	80.76	0.013602354
	(2.7)	2	0	74.5192	0.0465036
	(2.8)	2	3	81.73	0.013424481
	(2.9)	2	22	81.25	0.014582663

Based on the clustering of the Landsat Satellite data set we obtain two clusters. Therefore, we consider two as the best quantity of hidden layers. So after specifying the quantity of hidden layers we determine the quantity of hidden units based on formulas (2.4)-(2.9).

As it is observed from Table 3-16, the best classification accuracy for the data set Landsat Satellite is acquired for 13 hidden units in each layer with Accuracy equal to 82%. The smallest error/epoch equal to 0.02803713636 identified for a quantity of 13 neurons. This enables us to deduce that the optimum quantity of hidden units is 13 which means that formula (2.9) is the preferable choice.

That means formula (2.9) is the best compared to other formulas. Since the Neuroshell program uses formula (2.9), which is used in the previous experimental tests that confirm the performance of the simulation software is used and this enhances the accuracy of previous results.

For the other datasets formula (2.9) is the best in most of the results and so close to best in the rest.

3.10. Documentary related technique used for establishing the architecture of neural networks

This Documentary Related Technique for investigating how the quantity of hidden layers and the quantity of hidden neurons affect the classification accuracy of a MLP. We try to show that the proposed method in this chapter is correct and leads to the best architecture of neural networks.

Weka platform indicates the accuracy level in different categories per number of neurons. The research adopted “Weka” [112] to develop a third party platform. It is a machine learning program and offers unique conditions for data mining. It is also open-source [113].

3.10.1. Experimental result for percentage of classification accuracy of datasets

(1) Sonar Dataset

Table 3-22 below shows the results in terms of percentage of classification accuracy for the training of Sonar dataset with a different number of the epochs.

Table 3-22 Classification accuracy for the training of Sonar dataset with a different number of epochs

Epoch	Accuracy	Epoch	Accuracy
6	86.0577	600	99.5192
10	89.4231	650	99.5192
20	95.1923	700	99.5192
30	97.1154	750	99.5192
50	97.1154	800	99.5192
100	99.5192	850	99.5192
150	99.5192	900	99.5192
200	99.5192	950	99.5192
250	99.5192	1000	99.5192
300	99.5192	1500	99.5192
350	99.5192	2300	99.5192
400	99.5192	2900	99.5192
450	99.5192	3500	99.5192
500	99.5192	28000	99.5192
550	99.5192	30000	99.5192

Figure 3-35 represents a chart of epoch number and the corresponding percentage of classification accuracy:

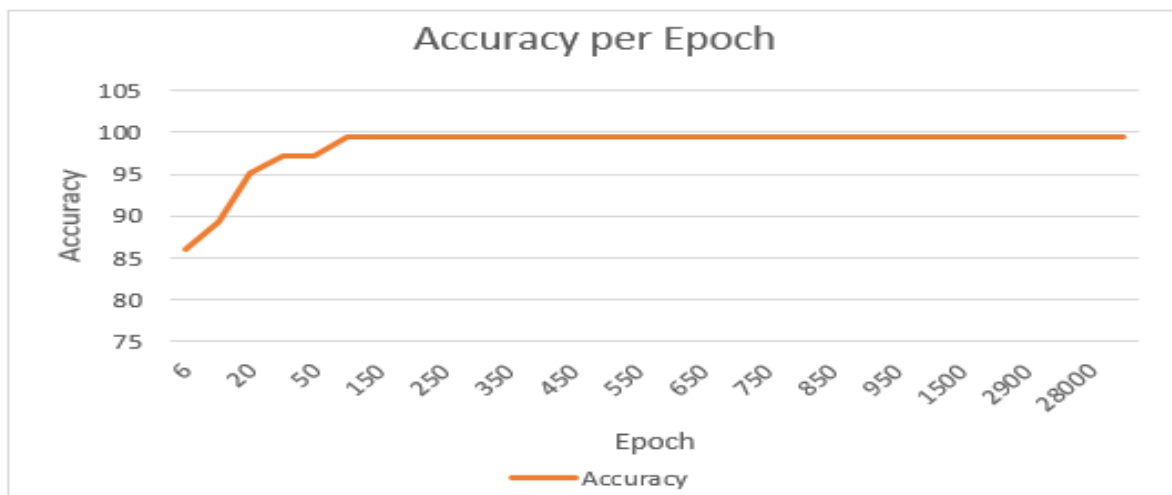


Figure 3-35 Chart of epoch number and the corresponding percentage of classification accuracy

As observed in Table 3-22 and Figure 3-35, the best accuracy is 99.5192% obtained with a quantity of 300 epoch. For this dataset, we can use 300 epochs for the experiment to identify the quantity of hidden layers/units.

Table 3-23 accuracy for the training of Sonar dataset with a various quantity of hidden layers/units

Hidden Units Per Layer	6 neurons	20 neurons	40 neurons	60 neurons	80 neurons	100 neurons
one Hidden layer	99.5192	99.5192	99.5192	99.0385	99.0385	99.0385
two Hidden layers	100	99.5192	98.0769	98.5577	98.0769	99.0385
three Hidden layers	99.0385	99.0385	99.0385	99.0385	99.0385	98.0769
four Hidden layers	53.3654	53.3654	98.0769	53.3654	97.5962	97.5962

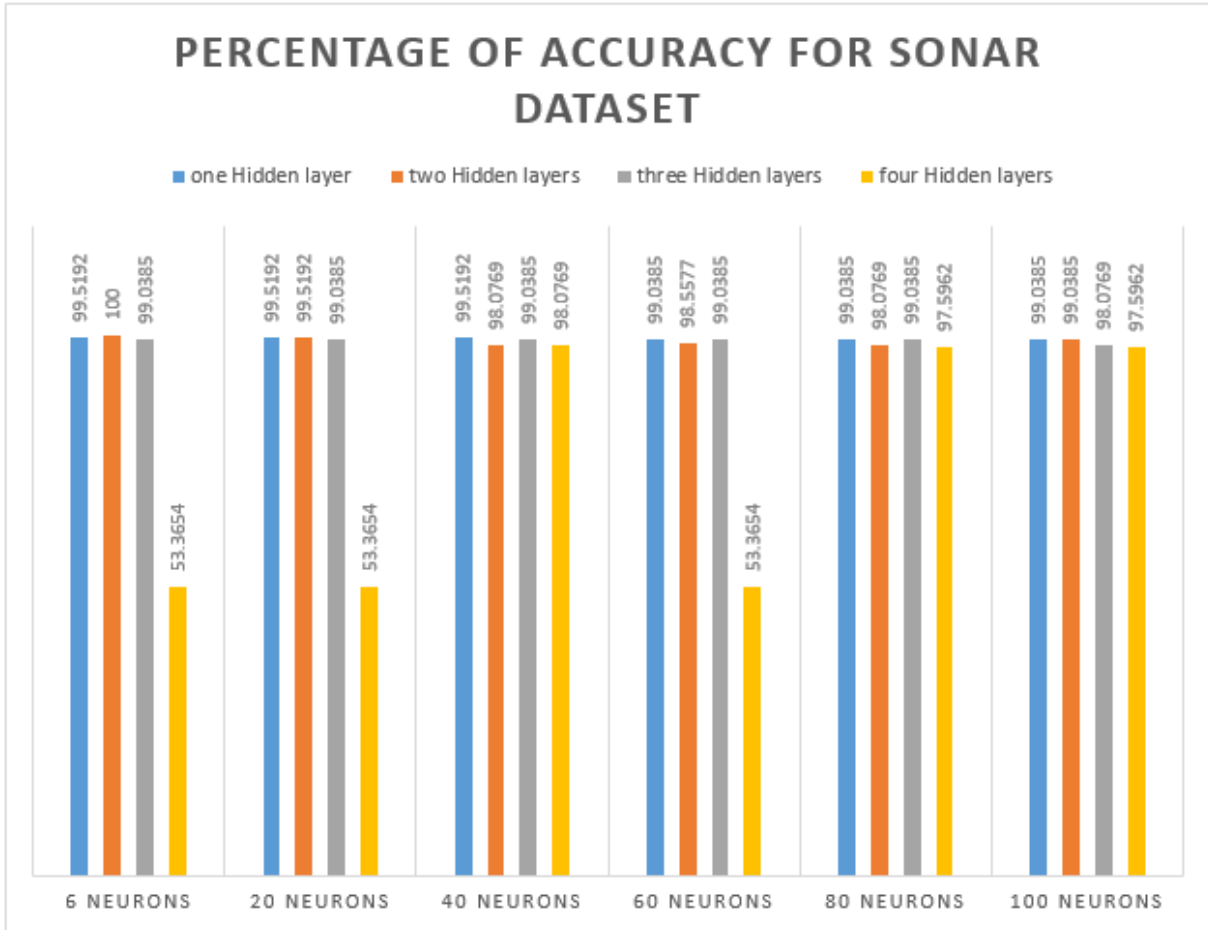


Figure 3-36 accuracy for the training of Sonar dataset with various quantity of hidden layers/units

As observed in Table 3-23 and from Figure 3-36, the best accuracy is 100% obtained with two layers. Therefore, we can confirm that 2 is the best quantity of hidden layers. The selected quantity of hidden layers is equal to the obtained quantity of groups by clustering the training dataset according to the criteria described previously. The quantity of obtained groups is based on the proposed method in the previous section.

(2) Glass Identification Dataset

Table 3-24 below shows the results in terms of percentage of classification accuracy for training of Glass Identification dataset with different number of epochs.

Table 3-24 Classification accuracy for the training of Glass Identification dataset with different number of epochs

Epoch	Accuracy	Epoch	Accuracy	Epoch	Accuracy
50	63.0841	1300	82.7103	3600	89.2523
100	68.2243	1400	83.6449	3700	89.7196
150	71.4953	1500	83.6449	3800	89.7196
200	73.3645	1600	83.6449	3900	90.1869
250	75.7009	1800	83.6449	4000	90.1869
300	76.6355	1900	85.9813	4500	90.6542
350	78.5047	2000	85.0467	5000	89.2523
400	78.0374	2100	87.3832	5500	89.2523
450	79.9065	2200	86.4486	6000	89.7196
500	79.4393	2300	86.9159	6500	89.2523
550	80.3738	2400	87.3832	7000	90.6542
600	79.9065	2500	87.8505	7500	90.6542
650	80.3738	2600	89.2523	8000	90.6542
700	80.3738	2700	89.2523	8500	90.6542
750	81.3084	2800	89.2523	9000	90.6542
800	81.3084	2900	89.2523	9500	90.6542
850	81.3084	3000	89.2523	10000	90.6542
900	81.3084	3100	89.2523	21000	91.1215
950	82.243	3200	89.2523	23000	91.1215
1000	82.243	3300	89.2523	26000	88.3178
1100	83.6449	3400	89.2523	28000	91.1215
1200	82.7103	3500	89.2523	30000	90.1869

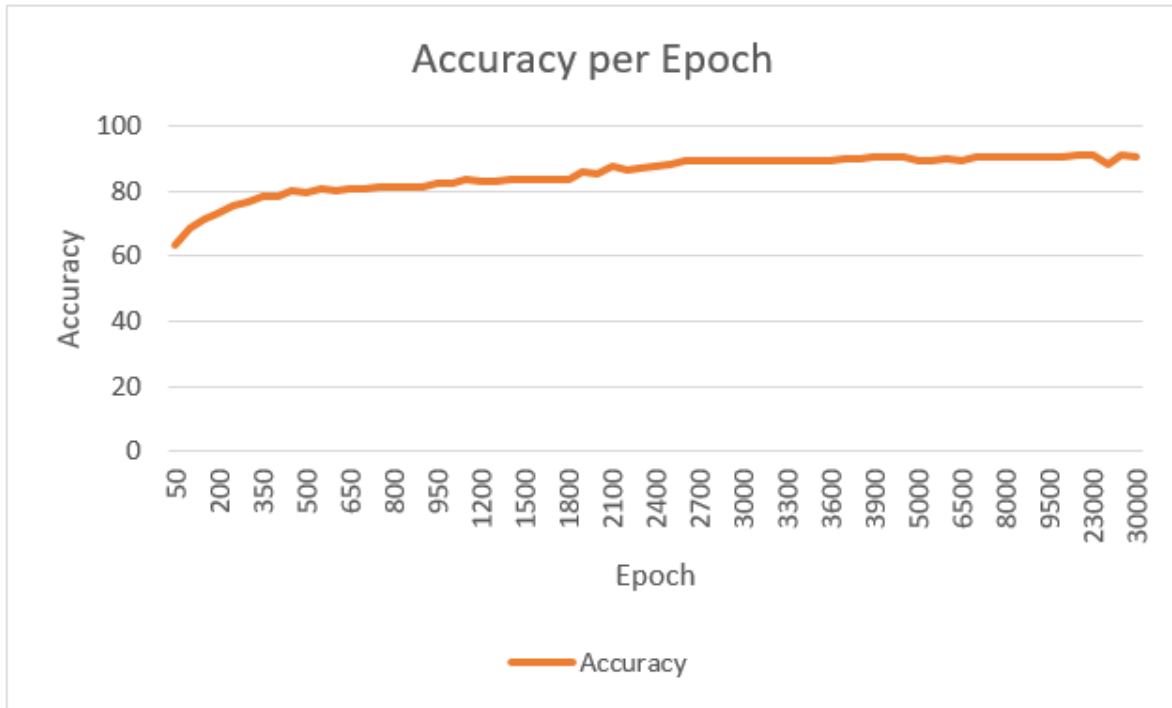


Figure 3-37 Chart of epoch number and the corresponding percentage of classification accuracy

As observed in Table 3-24 and Figure 3-37, the best accuracy is 91.1215% obtained with a quantity of 21000 epoch. For this dataset, we can use 21000 epochs for the experiment to define the best quantity of hidden layers/units.

Table 3-25 accuracy for the training of Glass Identification dataset with various quantity of hidden layers/units

Hidden Units Per Layer	6 neurons	20 neurons	40 neurons	60 neurons	80 neurons	100 neurons
one Hidden layer	84.1121	94.8598	95.3271	96.2617	96.2617	94.8598
two Hidden layers	89.7196	97.1963	98.1308	98.1308	97.6636	98.5981
three Hidden layers	87.3832	97.6636	98.5981	98.5981	97.6636	97.6636
four Hidden layers	35.514	92.5234	80.8411	92.0561	91.5888	85.0467

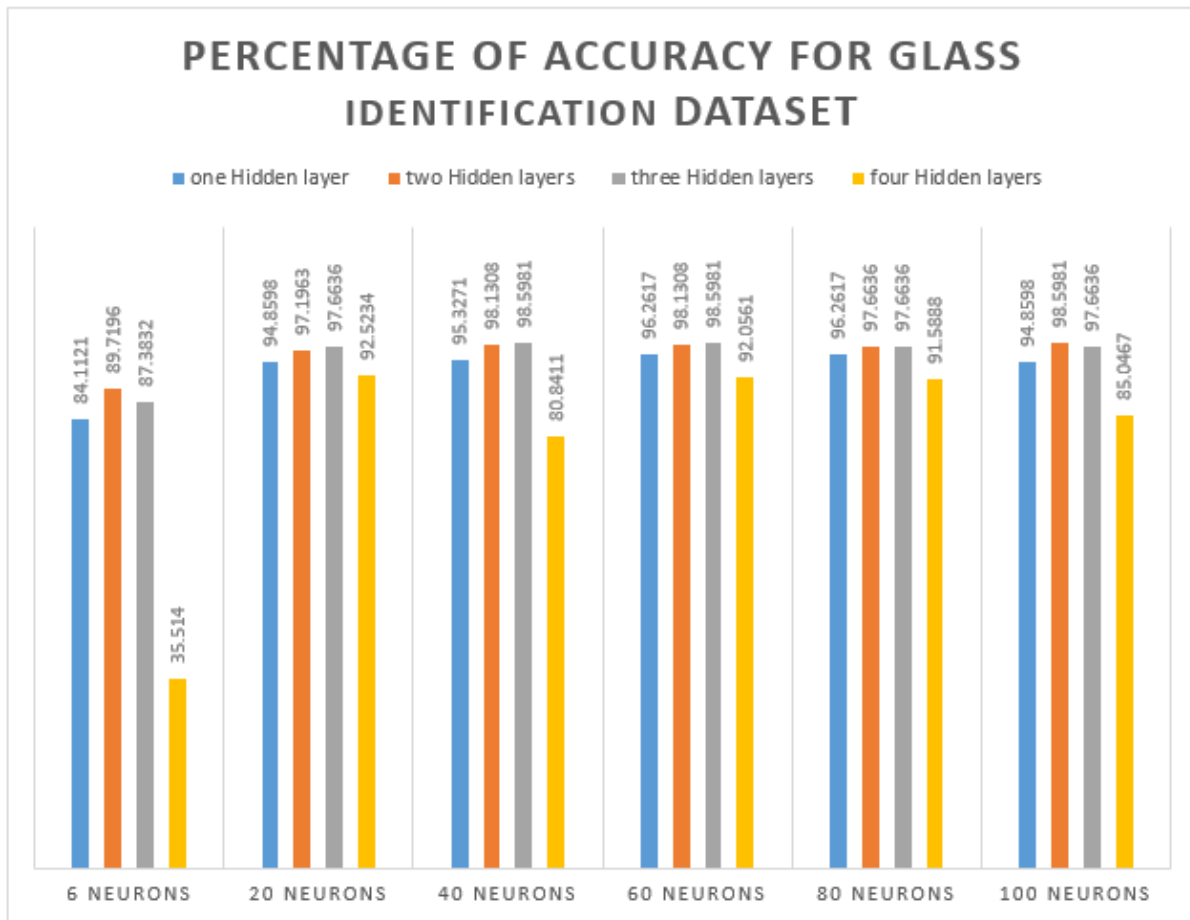


Figure 3-38 accuracy for the training of Glass Identification dataset with various quantity of hidden layers/units

As observed from Table 3-25 and from Figure 3-38, using 2 hidden layers we obtain the best accuracy which equals to 98.5981% that make as deduce that the optimum quantity of hidden layers is two for the presented case study. The selected quantity of hidden layers is equal to the obtained quantity of groups by clustering the training dataset according to the criteria described previously. The quantity of obtained groups is based on the method proposed in the previous section.

3.10.2. Comparison of experimental result of classification accuracy of datasets

In this case study, a comparison of the result obtained from the training of datasets with a various quantity of hidden layers/units will be done.

(1) *MLP composed of a single Hidden layer*

Table 3-26 below shows the percentage of classification accuracy for an MLP architecture which has single hidden layer.

Table 3-26 the percentage of classification accuracy for training of datasets with one hidden layer

Datasets	Percentage of accuracy for one Hidden layer					
	6	20	40	60	80	100
Sonar	99.5192	99.5192	99.5192	99.0385	99.0385	99.0385
iris	98.6667	98.6667	98.6667	98.6667	98.6667	98.6667
Landsat	86.7304	87.2086	86.9695	87.2086	86.7304	86.6707
Glass Identification	84.1121	94.8598	95.3271	96.2617	96.2617	94.8598

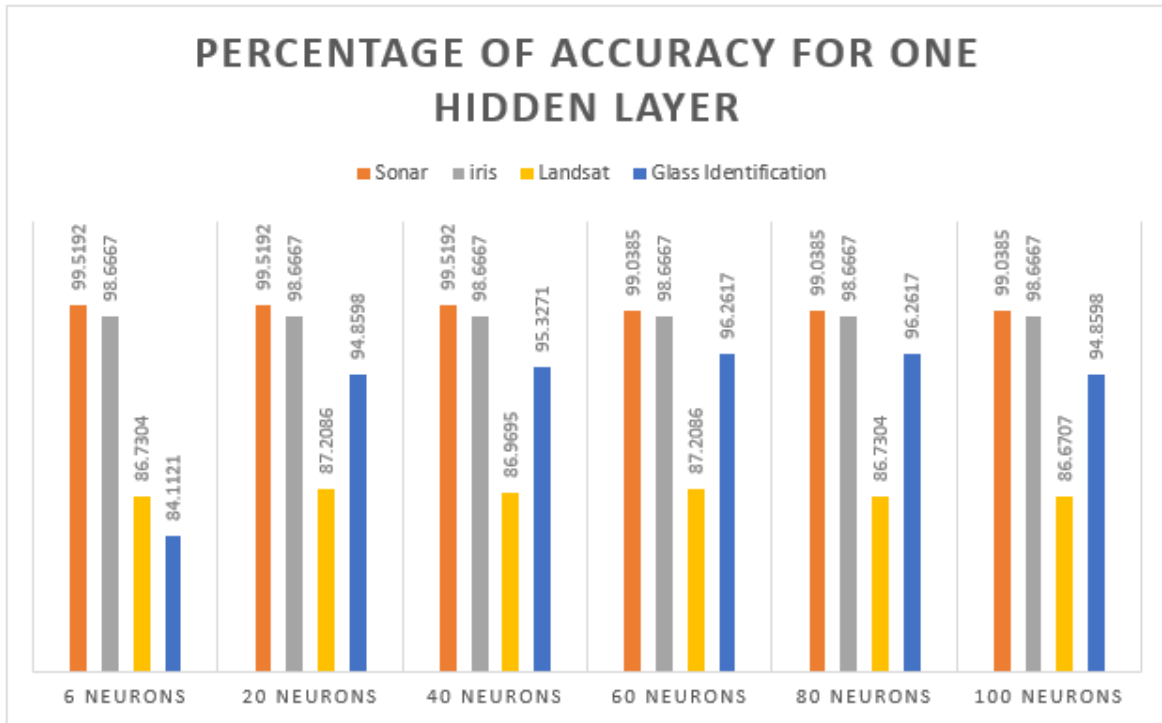


Figure 3-39 accuracy for training of datasets with one hidden layer

Based on obtained results from MLP architecture composed of a single hidden layer and various quantity of hidden units, we can confirm that the optimum quantity of hidden units for sonar dataset and iris dataset is six hidden neurons with 99.51 % classification accuracy for Sonar dataset and 98.66 % classification accuracy for iris dataset. The optimal number of hidden neurons for Landsat dataset is 20 hidden neurons with 87.20 % classification accuracy and the optimum quantity of hidden units for Glass Identification dataset is six hidden neurons with 96.26 % classification accuracy.

(2) *MLP composed of two Hidden layers*

Table 3-27 below shows the percentage of classification accuracy for an MLP architecture which has a single hidden layer.

Table 3-27 the percentage of classification accuracy for training of datasets with two hidden layers

Datasets	Percentage of accuracy for two Hidden layers							
	6	20	30	40	60	69	80	100
Hidden Units Per Layer	6	20	30	40	60	69	80	100
Sonar	100	99.5192	98.0769	98.0769	98.5577	98.0769	98.0769	99.0385
iris	98.6667	98.6667	98.6667	98.6667	98.6667	98.6667	98.6667	98.6667
Landsat	86.6707	86.7304	86.6109	86.4913	86.5511	87.2086	86.5511	86.6109
Glass Identification	89.7196	97.1963	98.5981	98.1308	98.1308	97.6636	97.6636	98.5981

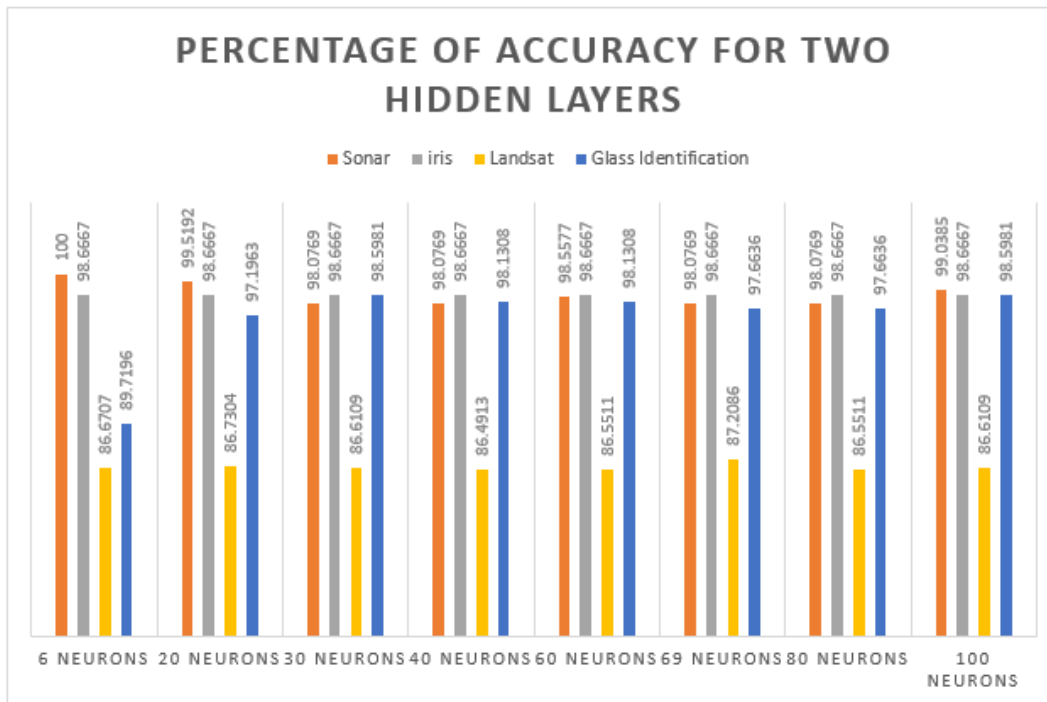


Figure 3-40 accuracy for training of datasets with two hidden layers

Based on obtained results from MLP architecture composed of two hidden layers and a various quantity of hidden units, we can confirm that the optimum quantity of hidden units for sonar dataset and iris dataset is six hidden neurons with 100% classification accuracy for Sonar dataset and 98.66 % classification accuracy for iris dataset. The optimum quantity of hidden units for Landsat dataset is 69 hidden neurons with 87.20 % classification accuracy and the optimum quantity of hidden units for Glass Identification dataset is 30 hidden neurons with 98.59% classification accuracy.

(3) MLP composed of three Hidden layers

Table 3-28 below shows the percentage of classification accuracy for an MLP architecture which has a single hidden layer.

Table 3-28 the percentage of classification accuracy for training of datasets with three Hidden layers

Datasets	Percentage of accuracy for three Hidden layers					
	6	20	40	60	80	100
Hidden Units Per Layer	6	20	40	60	80	100
Sonar	99.0385	99.0385	99.0385	99.0385	99.0385	98.0769
iris	99.3333	98.6667	98.6667	98.6667	98.6667	98.6667
Landsat	85.7143	86.6109	86.312	86.7902	86.1327	86.4913
Glass Identification	87.3832	97.6636	98.5981	98.5981	97.6636	97.6636

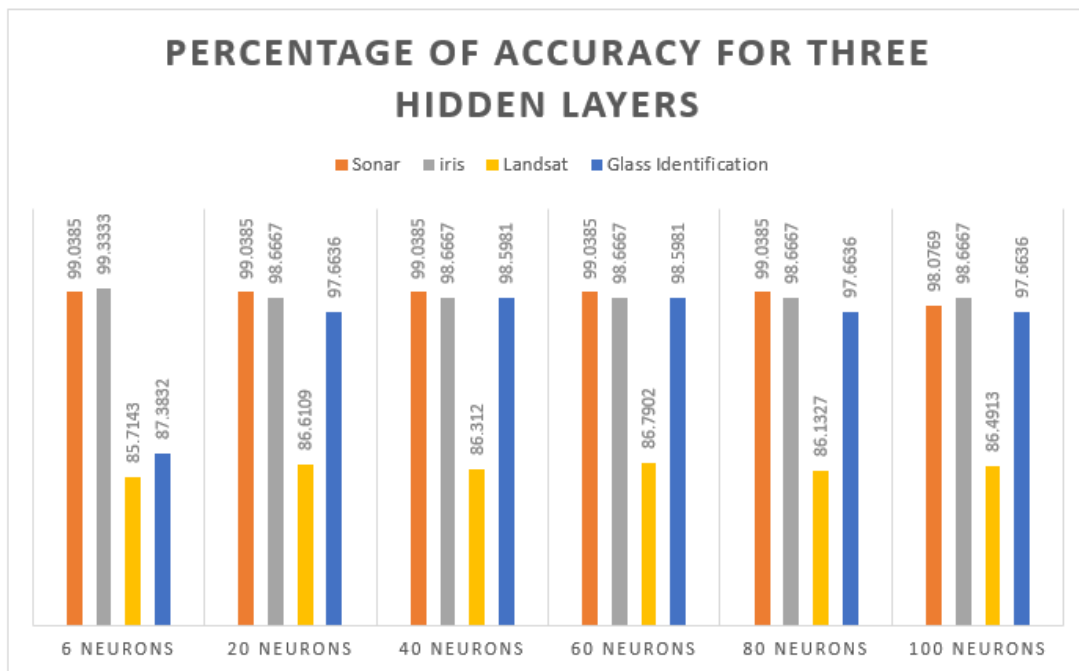


Figure 3-41 accuracy for training of datasets with three Hidden layers

Based on obtained results from MLP architecture composed of three hidden layers and various quantity of hidden units, we can confirm that the optimum quantity of hidden units for sonar dataset and iris dataset is six hidden neurons with 99.33% classification accuracy for Sonar dataset and 99.03 % classification accuracy for iris dataset. The optimum quantity of hidden units for Landsat dataset is 60 hidden neurons with 86.79% classification accuracy and the optimum quantity of hidden units for Glass Identification dataset is 40 hidden neurons with 98.59% classification accuracy.

(4) MLP composed of four Hidden layers

Table 3-29 below shows the percentage of classification accuracy for an MLP architecture which has a single hidden layer.

Table 3-29 the percentage of classification accuracy for training of datasets with four Hidden layers

Datasets	Percentage of accuracy for four Hidden layers					
Hidden Units Per Layer	6	20	40	60	80	100
Sonar	53.3654	53.3654	98.0769	53.3654	97.5962	97.5962
iris	33.3333	33.3333	33.3333	33.3333	33.3333	33.3333
Landsat	86.1925	86.0132	86.4913	86.0132	86.2522	86.2522
Glass Identification	35.514	92.5234	80.8411	92.0561	91.5888	85.0467

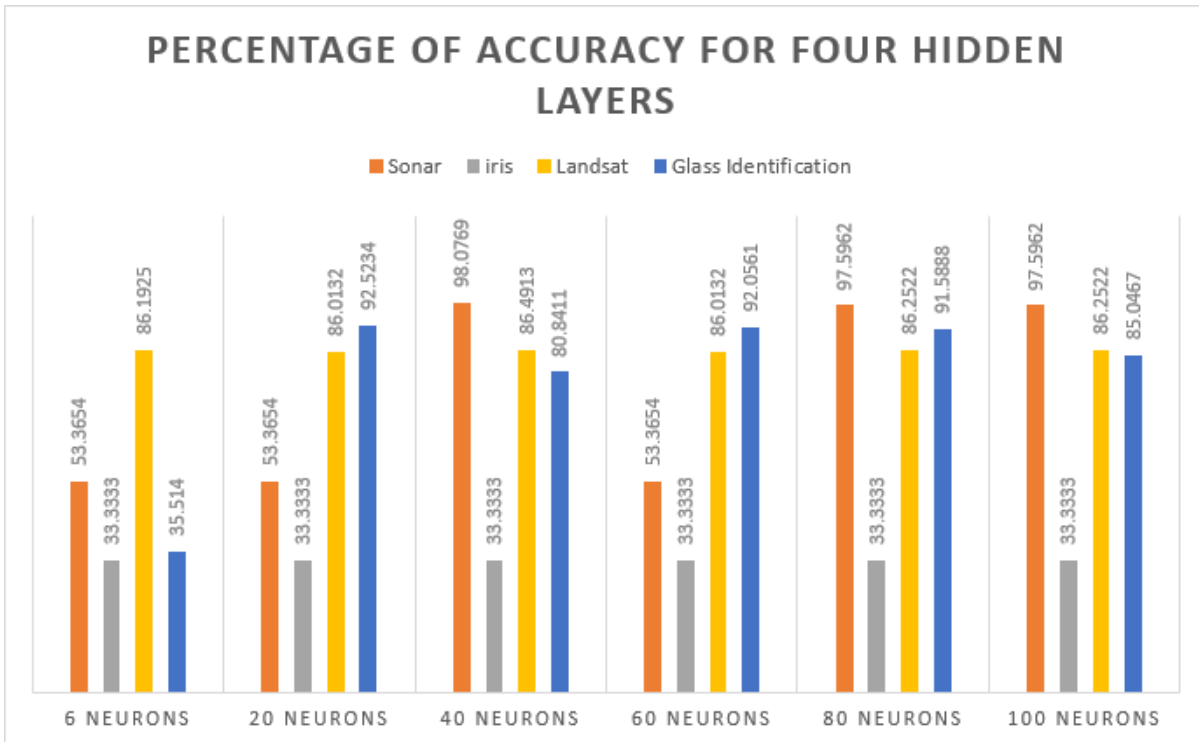


Figure 3-42 accuracy for training of datasets with four Hidden layers

Based on obtained results from MLP architecture composed of four hidden layers and a various number of hidden units, we can confirm that the optimum quantity of hidden units for sonar dataset and iris dataset is 40 hidden neurons with 98.07% classification accuracy for Sonar dataset and 86.49 % classification accuracy for Landsat dataset. The optimum quantity of hidden units for Glass Identification dataset is 20 hidden neurons with 92.52 % classification accuracy.

3.10.3. Documentary related technique conclusion

Based on obtained results from various case studies and compared to result obtained using the clustering method, it is observed that the proposed method is correct and leads to the best architecture for a neural network.

3.11. Conclusion

Using the proposed method, it is possible to identify the architecture of the MLP based on the complexity level of the problem to be solved.

Using clustering techniques, it is possible to highlight several common characteristics of input forms, on which they (input forms) can be classified into groups (clusters). The clustering of input forms generates a number of useful factors, which help to identify the optimum MLP architecture.

Based on the results obtained in this section it is observed that clustering algorithms used are affected by several factors which influence on the obtained quantity of hidden layers using the projected method. The most important factor is the RD, the accuracy of the value of this factor is necessary to achieve the optimum quantity of hidden layers.

The comparative study of clustering distance measures lends on the proposed method more effectiveness and accuracy through the perfect selection of the convenient distance measures for clustering technique used to identify the structure of the MLP neural network.

The comparative study presented in this section reinforce this method to support ordinal, definite, or combination of different data type or even interval that perfect recognition of clusters in a given dataset hence an improvement on the method's capability.

Many datasets are analyzed where it was concluded that the experimental results obtained confirm those obtained by clustering program, which proves the validity of the proposed method.

When you compare this method to traditional methods, we conclude that this method is the best for most datasets. Its flexibility is high, regardless of data size.

Using the proposed method, the design time and effort to identify the structure of the MLP are reduced. This method can make the design simple and easy and possible for a Non-specialist designer.

Chapter IV

4. Determination of optimal neural network Architecture using regression techniques

In this chapter, a new method of designing the structure of ANN using regression analysis is discussed. When we group a given set of data, we obtain features helping to understand the structure of an ANN. Features from the set of data can be assumed as independent variables to arrive at the regression function Data mining helps understand patterns in a given set of data and the nature of operation handled by the ANN. The projected regression method works unsupervised and provides the best results.

4.1.Introduction

This chapter shows how you can identify the architecture of ANN using regression techniques.

In order to identify the quantity of hidden layers/units of an ANN, a Multiple Linear Regression (MLR) models based on the parameters obtained from the clustering method described above is defined. In addition, a quality measure factor of the ANN structure is defined based on the interconnection of layers is used.

4.1.1. Statistical Modeling

Regression analysis is a statistical model to explains the connection between dependent and Independent variables. It depends on measurement observations. To understand the relationship between the two variables, the coefficient determinant must be availed R^2 [114], the coefficient will indicate change rate as a result of the relationship between the two variables. The coefficient value lies between 1 and 0. For the relationship to be high, the coefficient must be near 1. To have a high intensity of the relationship amongst variables, then R^2 must be close to one and must have some connection with statistical indicators.

Experimental information obtained through the grouping program are utilized to define regression functions to calculate the quantity of hidden layers/units of an MLP. The parameters obtained using the clustering program are enough according to the statistical indicators to define the regression models and to guarantee the quality and accuracy of the defined regression models. The MLR has this form:

$$Y = a + a_1x_1 + \dots + a_ix_i + \dots + a_nx_n \quad (4.1)$$

The models defined to calculate the quantity of hidden layers/units are based on the parameters include; the value of RD, the quantity of clusters, the quantity of input forms of the MLP, The quantity of grouped elements and the quantity of units in the input layer. Each regression model consists of 4 factors taken from the previous parameters.

4.1.2. The Confirmatory Data Analysis

“Hierarchical The statistical hypothesis testing” [115] is adopted to understand the limits arrived at hence helping in deciding the regression formula. For confirming the significance of the regression formula adopted a hypothesis test was conducted. The null hypothesis H_0 represented no significant connection between the variables.

H_0 : “connection does not exist between grouping results and multilayer perceptron design”

H_a : “connection exists between grouping results and multilayer perceptron design.”

The P-value of independent variables lies below 0.05, according to the projected limits in the regression model.

Proving the dependability of a multilayer perceptron architecture on identified factors will be done using F-Test [116]. The F-Test analysis the variance.

4.1.3. Designation of Factors

A set of inferences are derived from the previously cited studies:

- Based on the statistical hypothesis testing [117], the probability coefficients of independent variables for the proposed models is below five percent. Which prevents the possibility of rejecting these independent variables from the model.
- Depending on the statistical test F-Test, we made sure that the quantity of hidden layers/units rely on the considered factors.
- There are relationships between all selected factors. Furthermore, a comparatively slightly positive/negative connections occur; however, they are not important [118].
- The Multiple Determination Coefficient and the Multiple Correlation Coefficient results obtained [119] the two measures results are near to one. It proves the validity and efficiency of the defined models and the conciliator selection of factors included in the models.

Based on results obtained from algorithms used to identify the structure of ANN through information taken from the training examples it is seen that there is a connection correlate the parameters used to determine the structure of MLP and the results obtained with clustering algorithm:

Table 4-1 Selected parameters

Parameters
The clusters number
The percentage of clustered elements
Reference Distance
The number of training forms
The number of input layer units

An independent factor must exist to make the assumptions valid. In this case, quality measurement of the structures was adopted to fill the independent factor position- it considered layer interconnection [120].

The regression model comprises two methods that identify hidden layers/units.

4.2. Regression Method used

Regression methods are data mining techniques applied in this chapter to define the structure of the neural network [121], [122]. The experimental results obtained by grouping the learning dataset considered helpful according to the purpose, to define the structure of the MLP in terms of the quantity of hidden layers/units.

The regression methods bestow on the clustering method described in the previous chapter more accuracy and more relevance with the anomalies in the patterns found in the training dataset and the capability to predict the number of hidden neurons without resorting to the rules of thumb.

To transform a relationship between independent variables and a dependent variable into a function we need the data mining technique presented by the regression methods. If the analytical methods incapable to determine the changeable dependencies between variables and if so, the regression techniques will be the best solution.

Based on experimental observations or measurements it is possible to gather a set of data used to identify the regression model describing the dependency among the dependent/independent variables.

The regression function consists of the independent variable X and the dependent variable Y and the unknown parameter β . The function is $Y = f(X, \beta)$ helps to determine the value of the variable Y based on the known value of X . In general, Regression Analysis helps in predicting results. AI relies on the regression model to solve basic problems. The proposed method depends on the regression techniques to achieve the best possible prediction of the structure of the MLP neural network.

Regression functions to determine the quantity of hidden layers/nodes will be obtained by using a set of parameters extracted by grouping the data allocated to learn the network.

4.2.1. Stages of the Method Used

The proposed regression method depends on the results of pattern recognition algorithms applied on the learning data of ANN seeks to define the quantity of hidden layers/units is evolved through several stages:

- 1) Setting up the learning dataset by eliminating noise, deficient records and the contrasted records to other data.
- 2) Setting up the quantity of input units which will be equal to the features number in the training dataset of a neural network [41].
- 3) Setting up the number of output units which will be either equal to the number of classes if we have a dataset containing data for classification.
- 4) Setting up the quantity of hidden layers which will be equal to the quantity of clusters obtained by grouping the learning data of the ANN based on the criteria described below:
 - Clustering of at least 90% of elements included in the learning data of the MLP.
 - The clustering must be stable.

In this study, was concluded based on the experiment results that there is a link between the quantity of hidden layers and the quantity of clusters identified by clustering the learning data if the previously cited criteria are applied.

- 5) To prepare hidden nodes quantity: the MLR method is used to establish hidden nodes quantity. It does so by using a set of parameters extracted by grouping the data allocated to learn the network. A quality measure factor that considers the setting and layers linkage is defined.

Figure 4-1 below is a structure derived from the projected regression method. This model illustrates how results from grouping the learning data are used in determining models used in identifying neurons and hidden layers quantity.

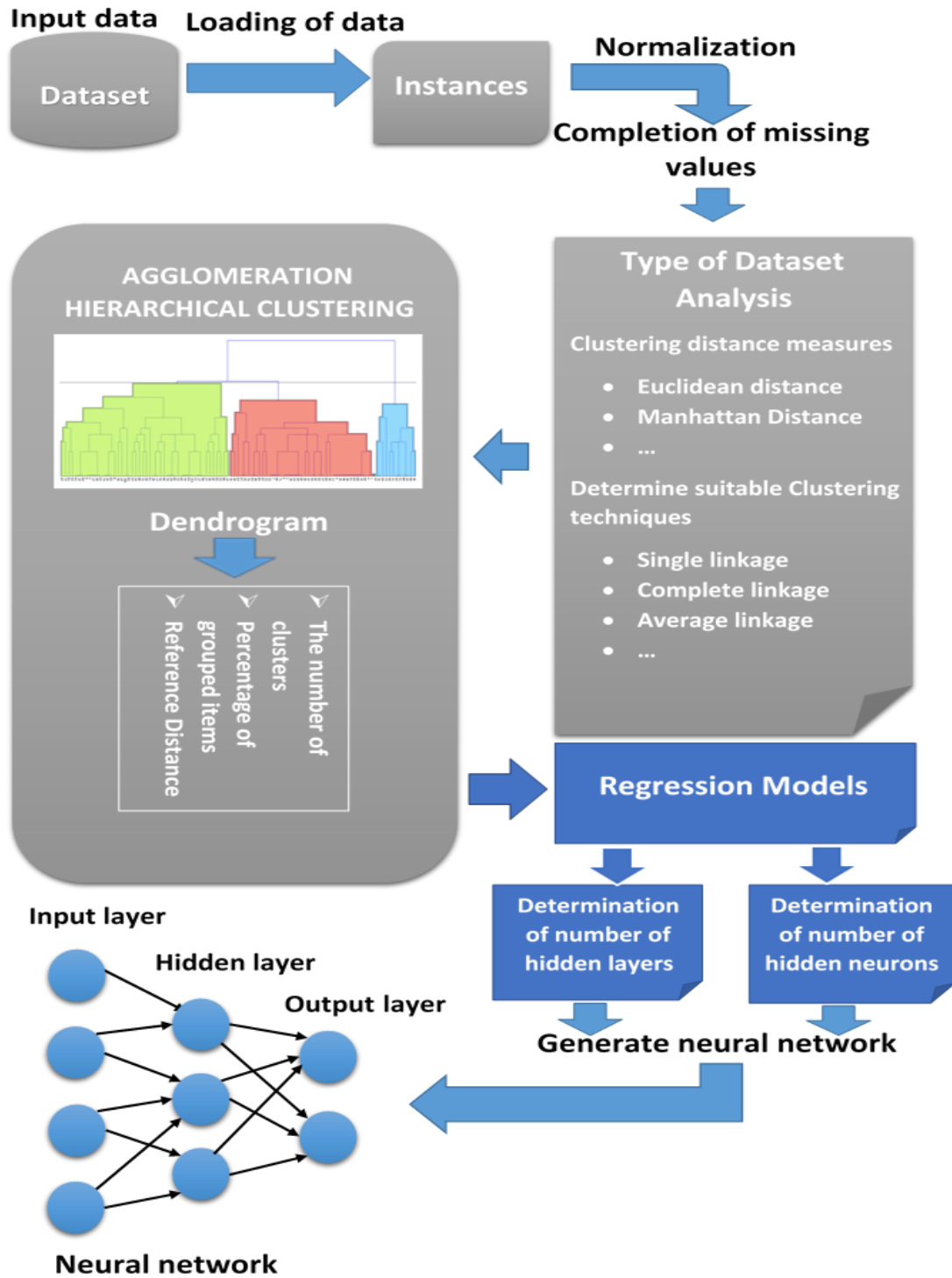


Figure 4-1 Proposed framework using regression methods

4.2.2. Factors to identify the quantity of hidden layers of an ANN

To determine a regression function, we need a set of experimental data showing the variation of the dependent variable Y depending on changes in independent variables Xi. If the available data are a set of examples for a ANN consist of an input vector and possibly output vector based on it the ANN train. Based on the experimental results of clustering the training dataset the selected parameters as independent variable of a regression model are useful in accordance with the purpose of identifying the structure of an ANN [41]:

Table 4-2 Selected factors to identify the hidden layers quantity

Number	INDEPENDENT VARIABLES
1	Number of groups obtained multiply by the Reference Distance
2	Reference distance
3	Clustered items percent
4	Quality measure for neural network.

The independent variables 1, 2 and 3 are obtained by using the clustering method described above are presented in the Table 4-3below.

To this, weight factors listed above in defining the quantity of hidden layers for the MLP was constructed a Multiple Linear Regression model of the form:

$$Y = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \quad (4.2)$$

The dependent variable Y represents the quantity of hidden layers of the ANN and the independent variables X₁, X₂, X₃, and X₄ are presented in the table below.

Table 4-3: The independent variables [123]

independent variable	Meaning
X₁	Number of groups × Reference Distance
X₂	Reference Distance
X₃	Clustered items percent
X₄	Quality measure of neural network

The figure below represents the absolute values of partial correlation coefficients which describe the influence of X₁, X₂, X₃ and X₄ on Y based on the proportion of the contribution of independent variables within the regression model.

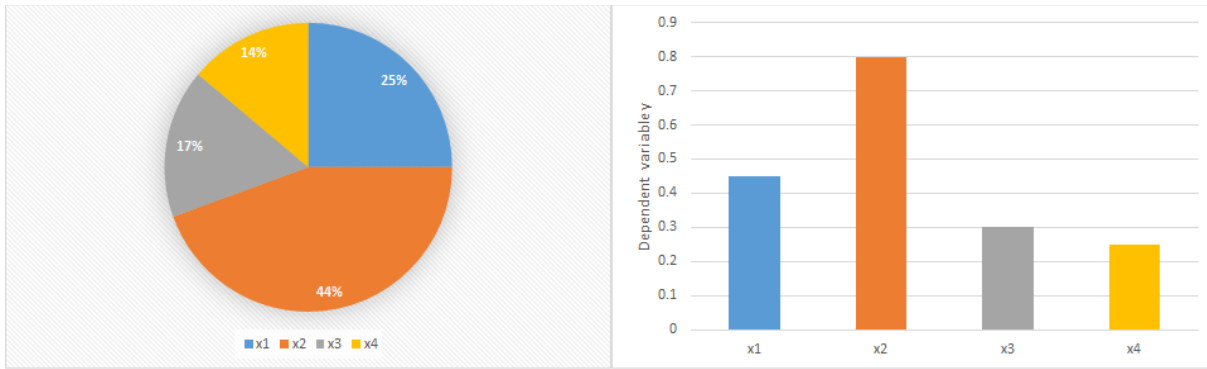


Figure 4-2 Coefficients Diagram [123]

The optimum quantity of hidden layers of an ANN affected by various factors, the most significant influence is by the RD factor for which the data grouping of multi-layer neural network training.

The generated regression model is consisting of 4 factors obtained as follows. Three factors obtained from grouping the learning data of the ANN and one factor represents the quality measure of the architecture of the ANN.

4.2.3. Factors to identify the quantity of hidden units of ANN

As stated in the previous subsections to identify a regression model, we need an experimental data obtained by observations or measurements. In case ANN with the only available data are a set of training examples consist of an input vector and output vector. Based on these data it is relatively easy to identify the quantity of units in the input layer, and the concerned of the output layer of the ANN but we cannot identify the quantity of hidden layers/units.

To identify the quantity of hidden units of the ANN, a regression method must be used based on the experimental data obtained previously using the grouping method. If the ANN has more than a single hidden layer, the quantity of calculated units is divided by the quantity of hidden layers so the quantity of hidden units being identical in each hidden layer. To identify the quantity of hidden units, designer usually use several heuristic methods that take into account the quantity of inputs, the quantity of output and the quantity of examples of learning methods.

From trial results considered independent variables in generating regression formula, which then identifies units in hidden layers:

Table 4-4 Selected factors to identify the hidden layers quantity

Number	INDEPENDENT VARIABLES
1	Number of inputs units
2	Number of obtained clusters
3	The value Deference Distance
4	The quality measure of the neural network

X_1 represents the independent variable obtained by determining the number of features presented in the learning data, X_2 is the second independent variable which represents the quantity of groups obtained by grouping the learning data presented in the previous chapter.

Based on the factors presented above was built an MLR model of the form:

$$Y = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \quad (4.3)$$

The dependent variable Y represents the quantity of units in the hidden layers of an ANN. If the quantity of hidden layers exceeded one layer the quantity of units will be evenly distributed on the quantity of hidden layers and so the quantity of hidden units in each hidden layer will equal.

The meaning of the independent variables x_i is given in Table 4-5.

Table 4-5: The independent variables [123]

independent variable	Meaning
X_1	Number of inputs units
X_2	Number of obtained clusters
X_3	Reference distance
X_4	Quality measure of the neural network

The figure below represents the absolute values of partial correlation coefficients which describe the influence of X_1 , X_2 , X_3 and X_4 on Y based on the proportion of the contribution of independent variables within the regression model.

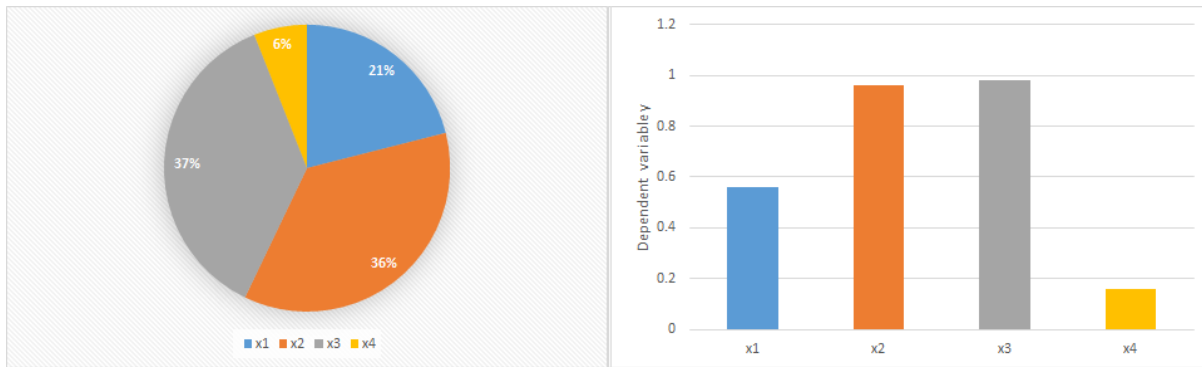


Figure 4-3: Coefficients Diagram [123]

The independent variables x_2 have influence percentage equal to 36% and x_3 have influence percentage equal to 37% which represents the most influential variables. The independent variable x_3 presented by the reference distance has the maximum percentage of influence in the regression model then the independent variable x_2 come next to it which represent the number of clusters multiplied by the Reference Distance. x_1 and x_4 have a minimal influence percentage on the regression model.

The independent variable x_1 has a low influence in the regression model with a percentage equal to 21%.

The quality measure factor x_4 can take only two cases, the first case is when the architecture has connections among the hidden layers, in this case, quality measure factor takes 1 the second case is when for architecture don't have connections this case the quality measure factor take zero. Due to the weak influence of this factor, the decrease in the quantity of connections between hidden layers can cause an increase in the quantity of hidden layers.

It is easily seen in Figure 4-3 the importance of every independent variable to calculate the quantity of hidden units for an ANN.

The quantity of hidden units is influenced by all selected factors, but the RD has the highest influence compared to the quantity of clusters, quantity of inputs and the percentage of grouped items.

4.3. Experimental results using regression techniques

Previously we have built two Multiple Linear Regression models, one for identifying the quantity of hidden layers and the second one is for identifying the quantity of hidden neurons for an MLP. The models used experimental results obtained from the clustering algorithm previously defined.

4.3.1. Experimental result for various quantities of hidden layers

We compare the results obtained using the regression method based on the values relative error for the Landsat Satellite dataset with the results obtained previously by an MLP structure composed of various quantities of hidden layers.

In order to determine the regression model based on the parameters obtained using the clustering method described above for the training dataset Landsat Satellite images the following values were obtained: $x_1 = 2 \cdot 75$, $x_2 = 75$, $x_3 = 90$, $x_4 = 87$. Replacing in equation (4.2) we will obtain

$$Y = (0.04483) * x_1 + (-0.06045) * x_2 + (0.00742) * x_3 + (-0.00695) * x_4$$

$Y = 2.25 \approx 2$ the neural network that analyzes a Landsat Satellite images should contain two hidden layers. Thus, affirm the results identified by the comparison of different neural network architecture to define the best architecture.

Table 4-6: The number of layers of neural networks using regression techniques for all Datasets

Data Set	a_0	b_0	c_0	d_0	Y	The best number of hidden layers
Landsat Satellite	0.04483	-0.06045	0.00742	-0.00695	2.25	2
P-wave	0.1035	-0.2431	-0.0234	0.07171	1.41	2
QRS	0.162	2.694	-0.692	0.094	1.13	1
T-wave	0.0018	-2.642	0.264	0.147	2.60	3
ECG	0.271	0.280	-0.101	-0.038	1.09	1
Sonar	0.839	-6.252	0.100	0.025	1.74	2

The comparison of the results obtained using the proposed regression method for the six selected datasets presented in Table 4-6 with the results obtained by comparison of different neural network architecture to define the best architecture we confirm that the proposed regression method can predict the best quantity of hidden layers for an MLP.

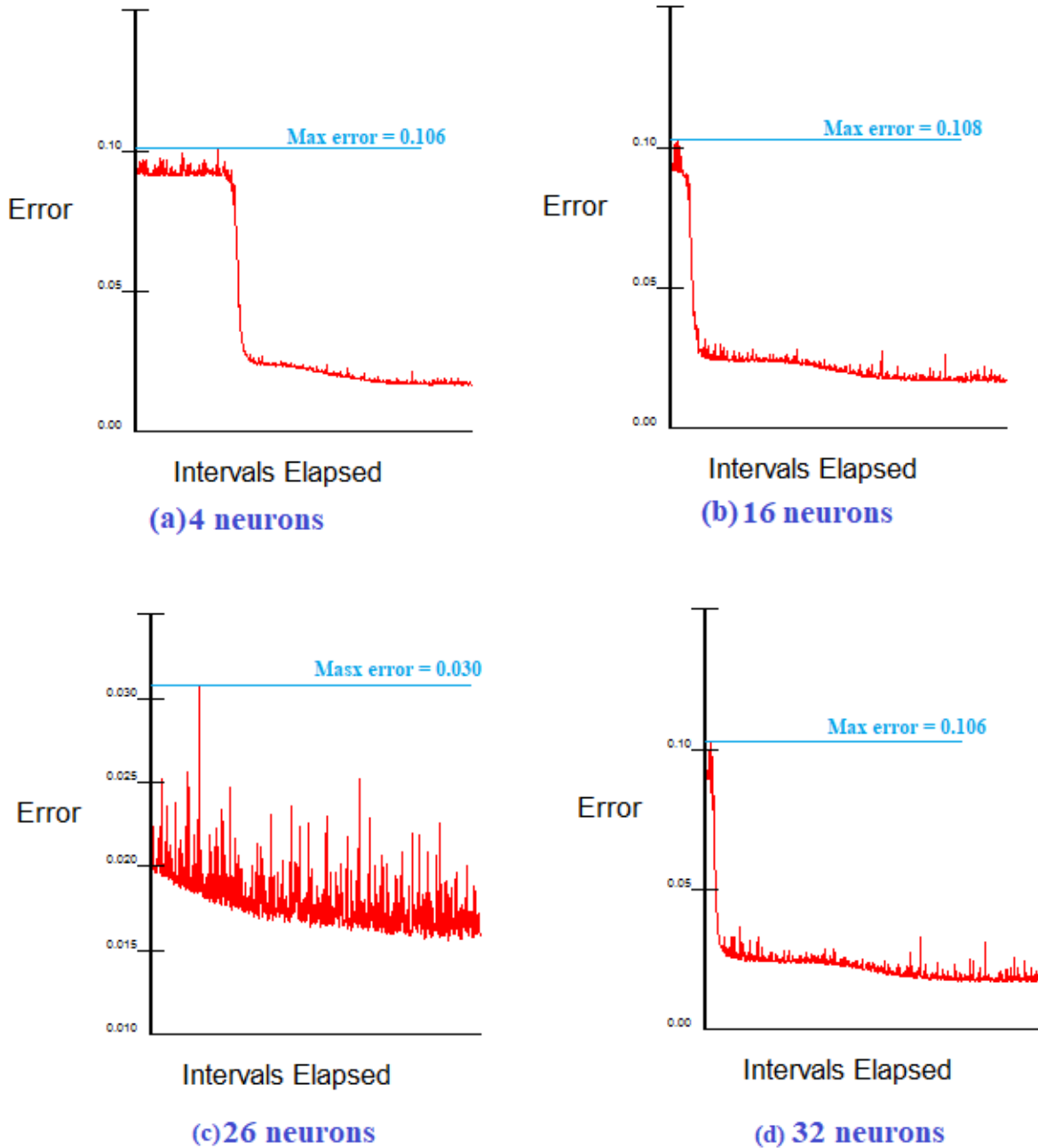
4.3.2. Experimental result for various quantities of hidden units

The following is the results obtained from the analysis of dataset with the same neural network architecture, the only difference between networks is the quantity of hidden units.

(1) *Experimental results for Landsat Satellite*

We will use the results obtained from training the Landsat Satellite images dataset to prove the validity of the projected regression method.

Figure 4-4 presents the obtained charts of errors generated from the training of the Landsat dataset for MLP architectures consist of 2 hidden layers and a various quantity of units.



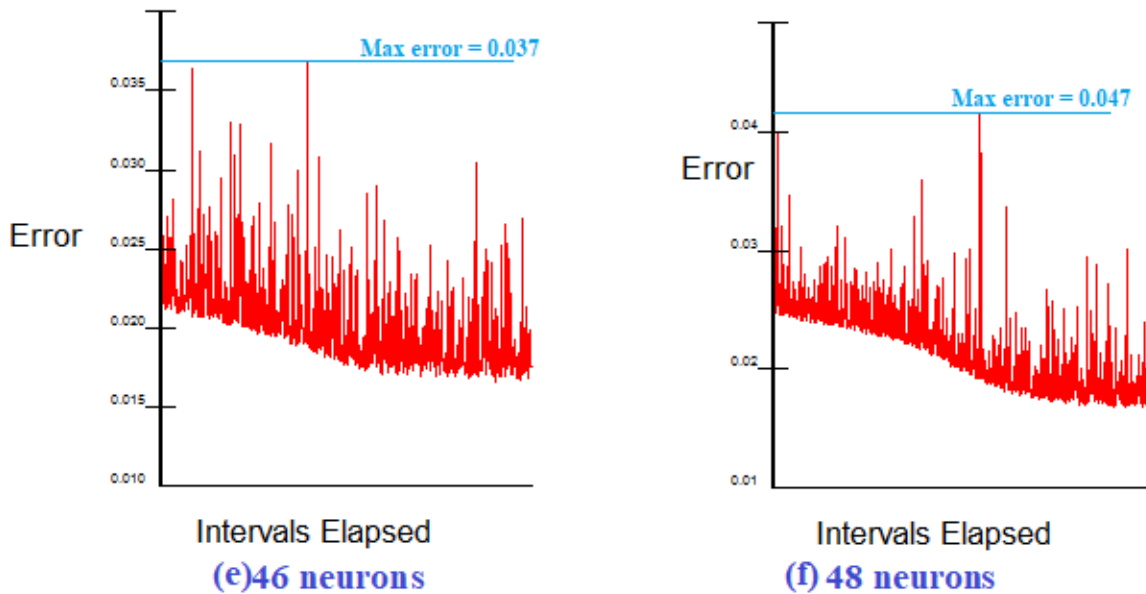


Figure 4-4: Errors indicator for the Landsat training data

It is seen from the figure that the neural network architectures with four units and network with 16 units generate higher values of errors exceed 0.10 and for a structure consists of 26 units has one answer with an error value that doesn't exceed 0.030. For a neural network structure consisted of 46 units and network structure with 48 units, we have error values exceed 0.036.

The experimental results for Landsat Satellite images dataset through a comparison of error values of various structures prove that the ANN architecture containing 26 hidden units achieves the lowest values of error compared of the other architectures.

We compare the results obtained using the regression method based on the values relative error for the Landsat training data with the results obtained previously by an MLP structure composed of various quantity of hidden units.

In order to identify the regression model to calculate the quantity of hidden units based on the parameters obtained using the clustering method described above for the training dataset Landsat Satellite images the following values were obtained: $x_1 = 4$, $x_2 = 2$, $x_3 = 75$, $x_4 = 85$. Replacing in equation (4.3) we will obtain:

$$Y = (5.717) * x_1 + (1.929) * x_2 + (-6.349) * x_3 + (2.257) * x_4$$

$Y = 23.88 \approx 24$ the neural network that analyzes a Landsat Satellite images should contain 24 hidden units. Thus, affirm the results identified by the comparison of different MLP architecture to define the best architecture.

(2) *Experimental results for P-wave*

Figure 4-5 presents a comparison of P-wave errors for MLP architectures consist of two hidden layers and a various quantity of hidden units.

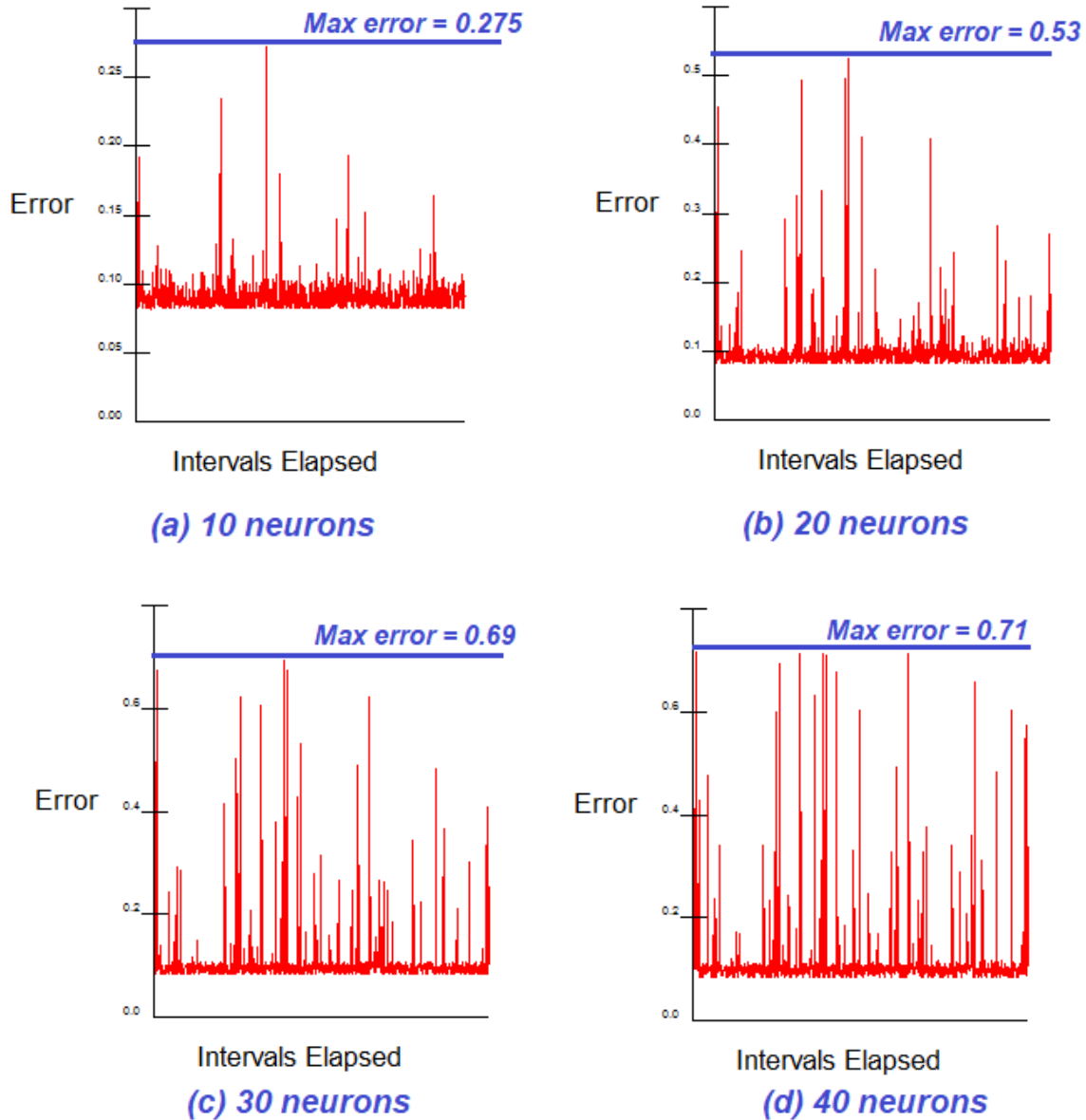


Figure 4-5: Errors indicator for the P-wave signal training data

It is seen from the figure that the neural network architectures with 20, 30 and 40 units generate higher values of errors exceed 0.53 and for a structure consists of 10 units has a larger error equal to 0.275.

The experimental results for P-wave signal dataset through a comparison of error values of various MLP structure prove that the MLP architecture containing 10 hidden units achieves the lowest values of error compared of the other architectures.

We compare the results obtained using the regression method based on the values relative error for the P-wave training data with the results obtained previously by an MLP structure composed of various quantity of hidden units.

In order to identify the regression model to calculate the quantity of hidden units based on the parameters obtained using the clustering method described above for the training dataset P-wave signal the following values were obtained: $x_1 = 2$, $x_2 = 2$, $x_3 = 11.4$, $x_4 = 1$.

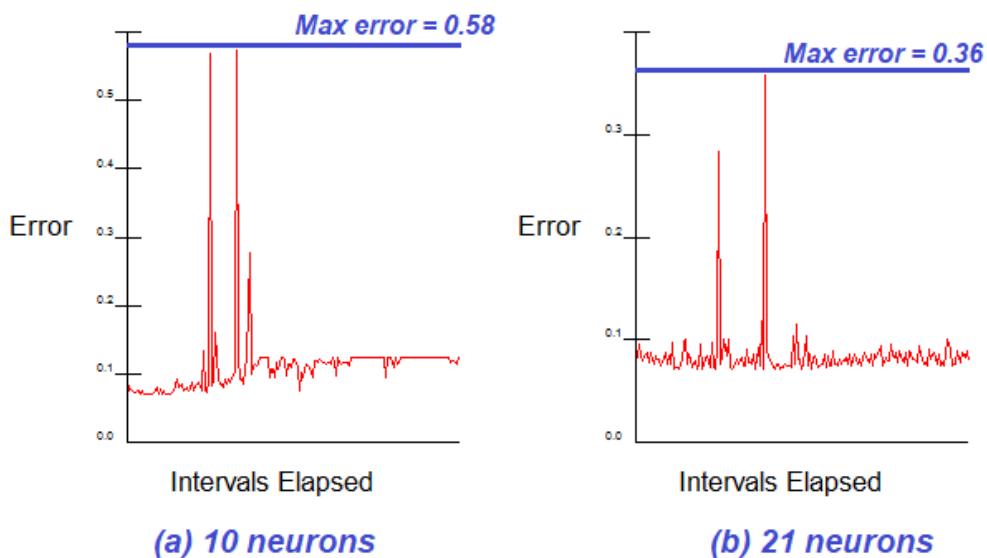
Replacing in equation (4.3) we will obtain:

$$Y = (1.676 * 10^{14}) * x_1 + (0.4668) * x_2 + (-0.127) * x_3 + (3.352 * 10^{14}) * x_4$$

$Y = 10.5 \approx 10$ the neural network that analyzes a P-wave dataset should contain 10 units in the hidden layers. Thus, affirm the results identified by the comparison of different neural network architecture to define the best architecture.

(3) *Experimental results for QRS complex*

Figure 4-6 presents a comparison of QRS for MLP architectures consist of one hidden layer and a various quantity of hidden units.



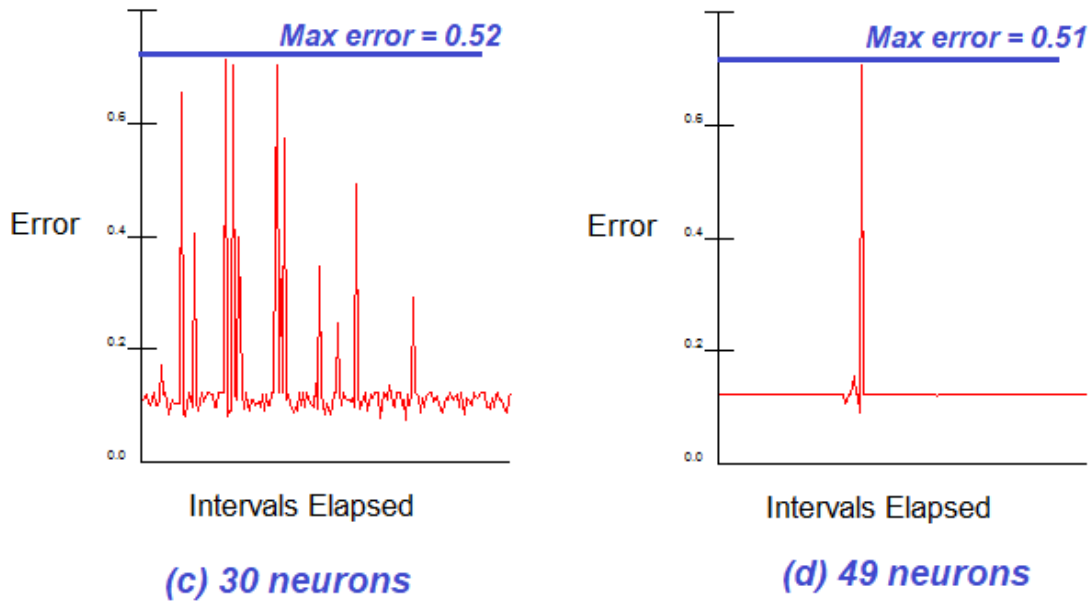


Figure 4-6: Errors indicator for the QRS signal training data

It is seen from the figure that the neural network architectures with 10, 30 and 49 generate higher values of errors exceed 0.51 and for a structure consists of 21 units has one answer with an error value that equal to 0.36.

The experimental results for QRS signal dataset through a comparison of error values of various MLP structure prove that the MLP architecture containing 21 hidden units achieves the lowest values of error compared of the other architectures.

We compare the results obtained using the regression method based on the values relative error for the QRS signal dataset with the results obtained previously by an MLP neural network structure composed of different numbers of units in the hidden layers.

In order to identify the regression model to calculate the quantity of hidden units based on the parameters obtained using the clustering method described above for the training QRS signal the following values were obtained: $x_1 = 2$, $x_2 = 1$, $x_3 = 22$, $x_4 = 1$.

Replacing in equation (4.3) we will obtain:

$$Y = (1.87 * 10^{14}) * x_1 + (0.005474) * x_2 + (-0.0107) * x_3 + (3.74 * 10^{14}) * x_4$$

$Y = 21.25 \approx 21$ the neural network that analyzes a QRS dataset should contain 21 units in the hidden layers. Thus, affirm the results identified by the comparison of different neural network architecture to define the best architecture.

(4) *Experimental results for the T wave signal*

Figure 4-7 presents a comparison of T-wave errors for MLP architectures consist of three hidden layers and a various quantity of hidden units.

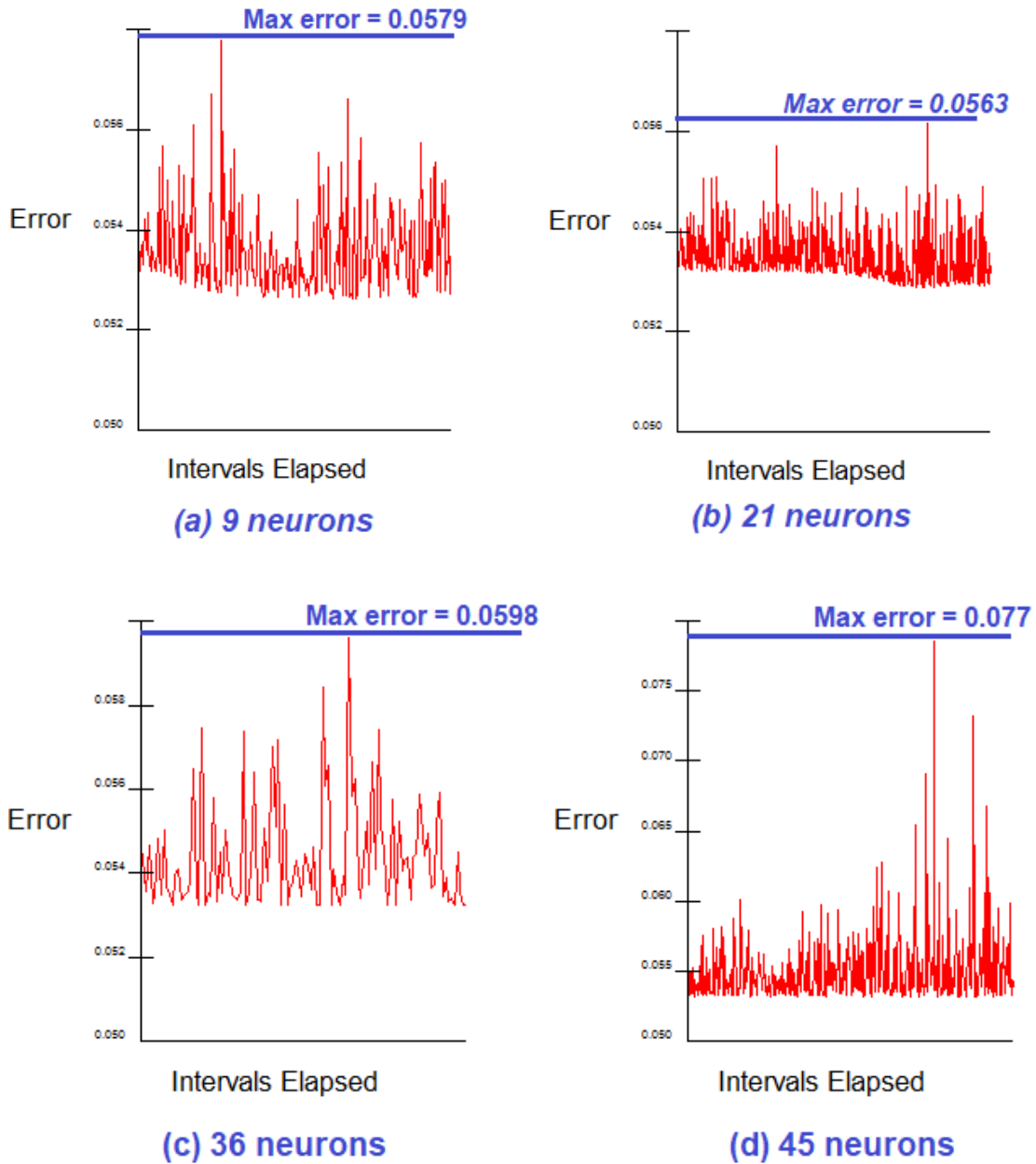


Figure 4-7: Errors indicator for the T wave signal training data

It is seen from the figure that the neural network architectures with 9, 36 and 45 units generate higher values of errors exceed 0.579 and for a structure consists of 21 units has a larger error equal to 0.0563.

The experimental results for T-wave signal dataset through a comparison of error values of various MLP structure prove that the MLP architecture containing 21 hidden units achieves the lowest values of error compared of the other architectures.

We compare the results obtained using the regression method based on the values relative error for the T-wave signal dataset with the results obtained previously by an MLP neural network structure composed of different numbers of units in the hidden layers.

In order to identify the regression model to calculate the quantity of hidden units based on the parameters obtained using the clustering method described above for the training dataset T-wave signal the following values were obtained: $x_1 = 2, x_2 = 1, x_3 = 13, x_4 = 1$.

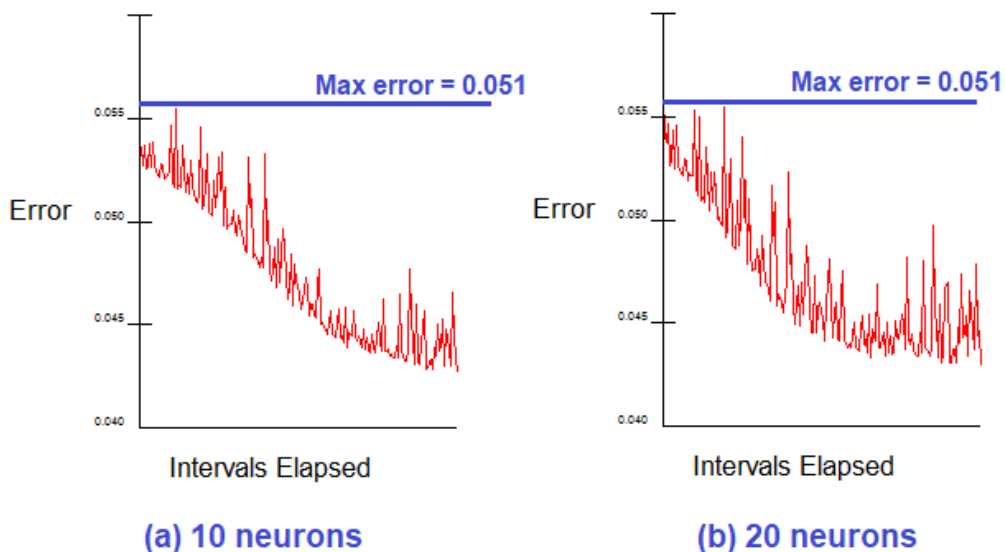
Replacing in equation (4.3) we will obtain:

$$Y = (-3.919 * 10^{14}) * x_1 + (0.379) * x_2 + (-0.0442) * x_3 + (7.839 * 10^{14}) * x_4$$

$Y = 19.75 \approx 20$ the neural network that analyzes a T-wave dataset should contain 20 hidden units. Thus, affirm the results identified by the comparison of different neural network architecture to define the best architecture.

(5) *Experimental results for a diagnostic ECG signal*

Figure 4-8 presents a comparison of ECG errors for MLP architectures consist of one hidden layer and a various quantity of hidden units.



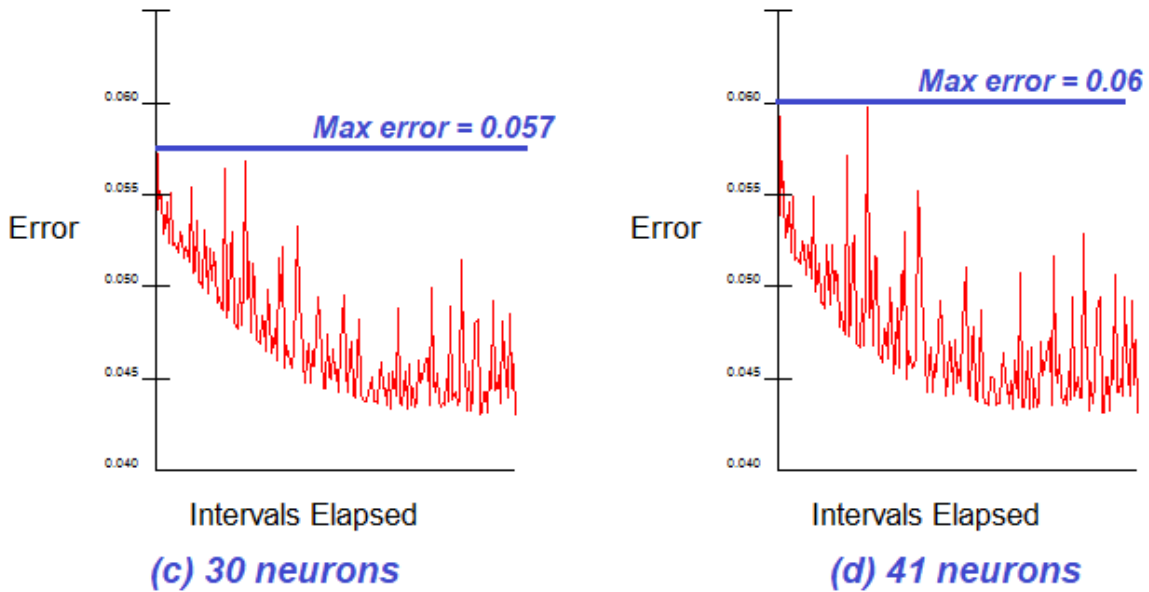


Figure 4-8: Errors indicator for the ECG signal training data

It is seen from the figure that the neural network architectures with four units and network with 30 and 41 units generate higher values of errors exceed 0.057 and for a structure consists of 10 and 20 units has one answer with an error value that doesn't exceed 0.051.

The experimental results for ECG signal dataset through a comparison of error values of various MLP structure prove that the MLP architecture containing 10 and 20 hidden units achieve the lowest values of error compared of the other architectures.

We compare the results obtained using the regression method based on the values relative error for the ECG signal dataset with the results obtained previously by an MLP neural network structure composed of different numbers of units in the hidden layers.

In order to identify the regression model to calculate the quantity of hidden units based on the parameters obtained using the clustering method described above for the training dataset ECG signal the following values were obtained: $x_1 = 6$, $x_2 = 1$, $x_3 = 23$, $x_4 = 1$.

Replacing in equation (4.3) we will obtain:

$$Y = (1.244 * 10^{14}) * x_1 + (0.160) * x_2 + (-0.032) * x_3 + (7.465 * 10^{14}) * x_4$$

$Y = 20.37 \approx 20$ the neural network that analyzes an ECG dataset should contain 20 units in the hidden layers. Thus, affirm the results identified by comparison of different neural network architecture to define the best architecture.

(6) Experimental results for the analysis of a signal from a sonar

Figure 4-9 presents a comparison of Sonar signal errors for MLP architectures consist of two hidden layers and a various quantity of hidden units.

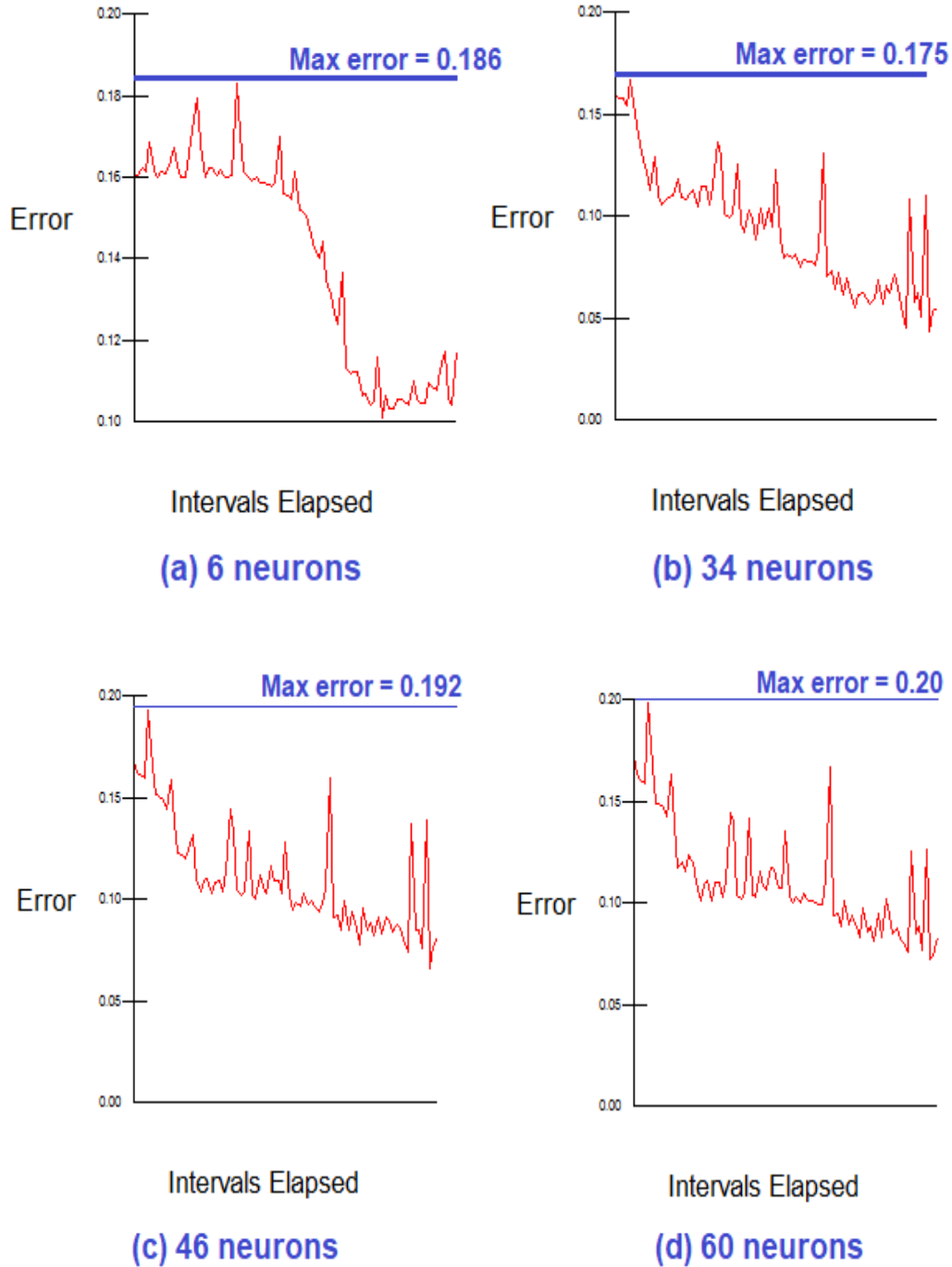


Figure 4-9: Errors indicator for the Sonar signal training data

It is seen from the figure that the neural network architectures with 6, 46 and 60 units generate higher values of errors exceed 0.186 and for a structure consists of 34 units has a larger error equal to 0.175.

The experimental results for Sonar signal dataset through a comparison of error values of various MLP structure prove that the MLP architecture containing 34 hidden units achieves the lowest values of error compared of the other architectures.

We compare the results obtained using the regression method based on the values relative error for the Sonar signal dataset with the results obtained previously by an MLP neural network structure composed of different numbers of units in the hidden layers.

In order to identify the regression model to calculate the quantity of hidden units based on the parameters obtained using the clustering method described above for the training dataset Sonar signal the following values were obtained: $x_1 = 60$, $x_2 = 2$, $x_3 = 2.1$, $x_4 = 1$. Replacing in equation (4.3) we will obtain:

$$Y = (0.261 * 10^{14}) * x_1 + (-0.230) * x_2 + (12.777) * x_3 + (-0.156 * 10^{14}) * x_4$$

$Y = 34.75 \approx 35$ the neural network that analyzes a Sonar dataset should contain 24 units in the hidden layers. Thus, affirm the results identified by the comparison of different neural network architecture to define the best architecture.

(7) *Conclusion for various quantity of hidden units' results*

The results obtained through the experimental test confirm the validity of the projected regression method to identify the quantity of hidden units of MLP based on the analysis of the selected dataset.

4.4. Conclusion

Paying attention to pattern recognition is an integral part when MLP's optimal structure is subject to discussion based on the proposed regression technique.

By grouping the learning data, useful parameters are collected making it easier to define the architecture of the ANN. These parameters are used as the independent variables of the regression function of the anticipated method under study. The reference distance of the independent variable highly impact results versus other variables.

Using the projected regression method to identify the structure of the MLP many factors affects the accuracy of the results. The main factor is RD which has a remarkable effect compared to the other factors. The value of RD must be selected precisely based on the criteria defined in the previous chapter.

Through this study, we conclude that the projected regression method to identify the architecture of the MLP in terms of the quantity of hidden layers/units is viable. The model generated using the projected regression method can be essential in practical applications and in the worst assumptions it can provide the initial quantity of hidden layers/units for an MLP and relying on the information obtained from the training dataset the designer can reduce or increase these numbers.

This study proposes and develops a new design strategy for an MLP neural network where make the design of the structure of an MLP neural network unsupervised and helps to reduce the time allocated for network design.

Chapter V

5. Importance of Training Data Analysis to Improve Generalization Capabilities of Neural Network Architectures

ABSTRACT

This chapter presents a contribution to append generalization capabilities to the proposed clustering method presented in chapter 3 to define the architectures of an ANN. The accuracy of the clustering method used requires precision when the selection of clustering distance measures which necessitate a comparative study. The proposed method depends on the results obtained through the clustering of the learning data. Therefore, the analysis of training dataset is important to improve the generalization capabilities of Neural Network Architectures. Due to the diverse types of datasets, the classical methods do not cover all type of datasets. The proposed method is more flexible with various types of datasets in terms of the type of data, size, and number of features. Many aspects have been taken into consideration to give more generalization capabilities to the proposed method such as distance measures and linkage methods.

5.1.Introduction

Machine learning depends mainly on the analysis of data to extract patterns and converts them into useful information used for unsupervised learning of systems. Machine learning is a branch of artificial intelligence specified in training a machine how to learn. Neural Network is one of the efficient algorithms for machine learning. Artificial Neural Network has evolved very much nowadays to cover many real-world applications. The common problem of neural network architecture where there are no well-developed formal methods to determine the structure of the neural network. Currently used methods are based on trial and error methods, heuristic rules, designer experience and other which are time-consuming and inaccurate methods. Classical methods are supervised and need an amendment to the structure of the network whenever there is an occurrence of a change in the complexity of the considered problem.

This chapter is an extension of the method originally presented in chapter 3, which depend on the results obtained from the clustering of learning data to optimize the neural network architecture. The level of complexity of the considered problem [124] can be evaluated by reveal patterns [125] [126] in training dataset and through it possible to define the structure of the neural

network. The proposed method depends on AHC algorithm. A comparison study is required to determine the suitable similarity measure for this algorithm [127] [128] [129].

This chapter presents a contribution to append generalization capabilities to the neural network architectures. By making a comparison of the results of diverse types of datasets, it was concluded that classical methods do not cover all type of training datasets. The effectiveness of the method varies with the variation of the complexity of the problem that needs to be resolved. The proposed method can adapt to the changes in the complexity of the problem and have more flexibility with different types of datasets.

5.2. Artificial Neural Network Structures and Generalization

There is a solid relationship between the generalization performance of artificial neural networks and their structure. It is shown that the generalization performance of neural networks is affected by the structure of the network. The quantity of hidden layers/units and the Hyper-parameters associated with network structure influence the generalization capability of the neural network. In this chapter, we discuss the generalization capabilities of ANN architectures through the analysis of training data.

5.3. Neural Network Architecture through grouping of training data

The projected method seeks to develop a general formula to identify the structure of a multi-layer ANN match different situations of the problems to be solved. Using pattern recognition methods [130] and depending on a set of criteria [131] [132] the results of grouping of learning data can be a solution to define the number of hidden layers.

By evaluating patterns discovered in data using pattern recognition method based on clustering algorithms, it is possible to determine the complexity of the considered problem.

This method is based on the concept that the complexity of the considered problem affects the network size [133] and thus affects the generalization capabilities [134]. Therefore, the quantity of discovered clusters through grouping can be considered as the optimum quantity of hidden layers, but this is possible only if several criteria are achieved.

The goal of this research is to determine the best clustering condition to define the perfect quantity of clusters. This quantity will be considered the best quantity of hidden layers desired for a specific classification problem. Therefore, the precision is required when the selection of clustering distance measures which necessitate a comparative study [135].

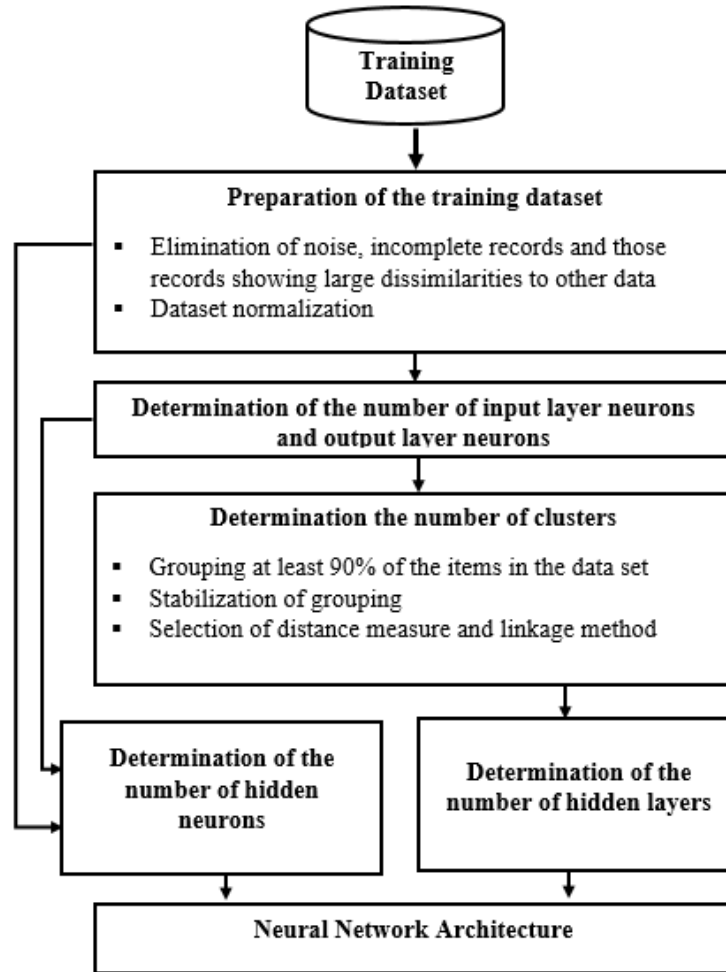


Figure 5-1 framework for the proposed method

The proposed method depends on the analysis of training dataset, the analysis goes through several stages. The first stage is after preparation of the dataset it will be assigned one input neuron for each feature presented in the dataset. If the considered problem is a classification problem, it will be assigned one output neuron for each class or the number of output neurons will be equal to the output features number.

In the second stage, the quantity of hidden layers will be defined. The experiment in this study has shown that by following a set of criteria the quantity of obtained clusters can be considered as the optimum quantity of hidden layers.

The initial criterion is clustering at least 90% of the training items. This percentage of grouped items enough to represent the complexity of the problem. The second criterion is the Reference Distance (RD) level in which we cut the obtained Dendrogram from clustering the training dataset must be in a point that any increase on it the quantity of clusters does not change. For the case of a high level of RD for which all elements in data set clustered into one single cluster and the case of a short distance for which each object represents a cluster are not taken into account. The

distance measures and linkage methods must be accurately selected which require a comparative study to determine the suitable distance measures. By achieving the mentioned criteria, the number of clusters obtained through grouping of learning data will be considered as the best quantity of hidden layers.

The third stage determines the quantity of hidden units. A comparison of several formulas it will be made until the perfect formula will be selected. Due to the lack of well-developed formal methods to identify the structure of the ANN, a comparison of most currently used formulas is necessary. Among the most commonly used formulas, there are Trial and Error, Pruning and constructive algorithms, Genetic search and Heuristic search [136] [137] [138].

In this chapter, the classification accuracy and the error per epoch will be taken as tools to identify the structure of the ANN. In addition, the size of the training data and the training time are taken into consideration.

There are several rules of thumb methods, which are widely used to identify the approximate quantity of hidden units. These methods depend on the theories that the quantity of hidden units should be in the midst of the number of input units and output units. The number of hidden units should be equal to two-thirds of the sum of the quantity of input units plus the quantity of output units. In addition, the quantity of hidden units should be below twice the quantity of input units [139]

Among the rule of thumb methods the Formulas (2.4)-(2.9) previously mentioned in chapter 2. These empirical formulas will be used to identify the quantity of hidden units.

The formula that achieves the best results will be relied upon in the comparative study to determine the suitable distance measures.

5.4.Comparison Study to Identify the Optimum ANN Structure

The projected method requires a perfect selection of distance measures due to the significant impact of the distance measures on the results of grouping of the learning data, therefore the distance measures affect the accuracy of the architecture of Neural Network. Consequently, it is required a comparative study to determine the suitable distance measures. Distance metrics (such as Manhattan and Euclidean) will be accredited for calculation of the distance between observations. Linkage methods (such as average-linkage clustering, complete-linkage clustering, and single-linkage clustering) used to calculate the distance between objects will be applied with the Clustering method.

The proposed method depends on AHC Algorithm. The generated Dendrogram will be cut to generate several clusters based on the value of distance in which the cut is made. This value of distance will be referred to as "Reference Distance" (RD). The main reason that imposes a comparative study to determine the suitable distance measures is the variation of datasets types. Each distance measures have effectiveness with a particular type of data, for this reason, it must define the suitable distance measures corresponding to the type of dataset. For example, interval

variables, nominal variables, or ordinal data, which required different types of distance measures [140] [141] [142].

Several similarity/distance measures are proposed in this chapter for the study. The distance measure Euclidean distance and Manhattan Distance are used for the comparative study to determine the suitable distance measures in addition to that the clustering techniques average linkage, Minimum linkage, and Maximum linkage must be discussed to define the similarity and dissimilarity measures in grouping of the learning data.

Like any data analysis process, the proposed method needs a dataset normalization in order to minimize data redundancy and maximize data integrity [143] [144]. The Normalization methods used are Z-score Normalization method and Standard Normalization method.

The proposed method for the determination of the structure of the neural network required a comparative study to determine the suitable distance measures. The reason is the importance of accuracy of the selection of distance measure for the clustering method used. A set of training datasets will be used for experiments using the previously mentioned distance measure techniques. For each dataset, the most effective technique will be defined. Due to the direct influence of the identified quantity of clusters on the precision of the results of the projected method, the distance measures techniques should be chosen tightly. Not all dataset works well with the same similarity measure. Therefore, a comparison study helps to define the appropriate distance measures for any particular case study which improve the generalization capabilities of the clustering method. The optimum quantity of hidden layers defined using the projected method will be equal to the quantity of the obtained clusters when the previously mentioned criteria are achieved. This makes the comparative study essential.

5.5. Studies for the Validation of Hypothetical Outcomes

In this chapter, the theoretical research is validated using a set of datasets. The Ionosphere dataset [145] will be analyzed in detail to explain how to select the perfect distance measures suitable for clustering the training dataset. The results of the rest of datasets will be compared to extract the efficiency of the projected method with various types of datasets.

The Ionosphere dataset contains a collected radar data. The dataset composed of 351 instances and 35 attributes.

For Data Mining Orange Canvas is used. Orange Canvas is open source machine learning and data visualization.

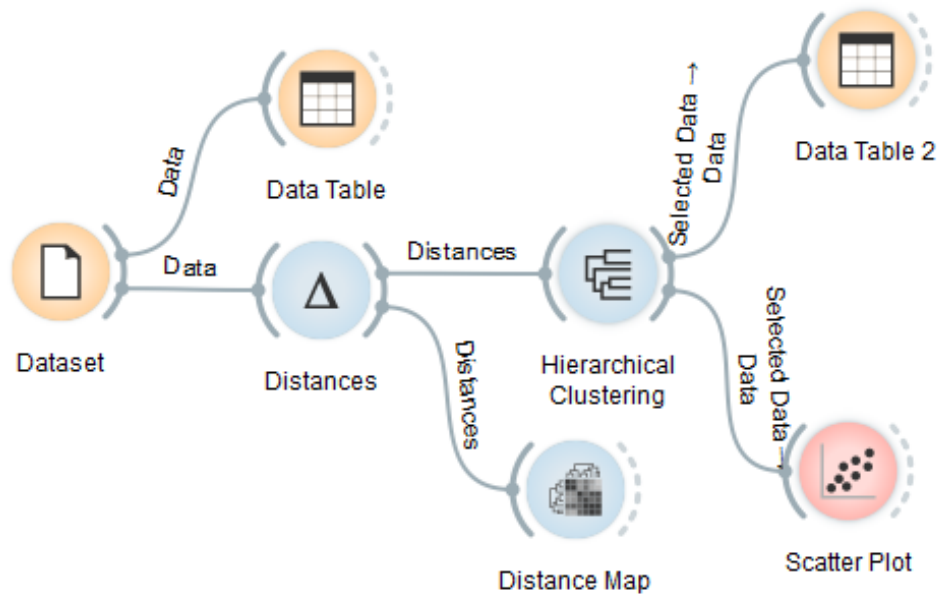


Figure 5-2 Orange Canvas open source machine learning and data visualization.

This research adopted “Weka” to develop a third party platform. Weka contains a collection of visualization tools and algorithms used in this chapter for the analysis of training dataset [146].

5.6. Experimental Results

5.6.1. Comparative Study of Clustering Distance Measures

Different clustering distance measures will be compared with each other to determine the appropriate distance measure to cluster the training dataset. A set of case studies will be tested, and the results will be compared to determine the suitable clustering distance measures, which can help to identify the optimum quantity of clusters for the selected dataset. Based on that we can define the structure of the neural network.

The evaluation of the structure of the neural network is made based on the values of error/epoch and the classification accuracy.

5.6.2. First Case Study

The first case study depends on Complete Linkage for clusters distance, Manhattan distance, and Standard normalization. The Neural Network activation function used is sigmoid.

Figure 5-3 presents the obtained Dendrogram for clustering of training dataset Ionosphere. For the selected value of reference distance $d = 0.695$, it is obtained two clusters based on the first case study and in accordance with the criteria described above concerning at least 90% of items of the dataset are grouped and by increasing the chosen reference distance dose note result a change on the number of clusters obtained.

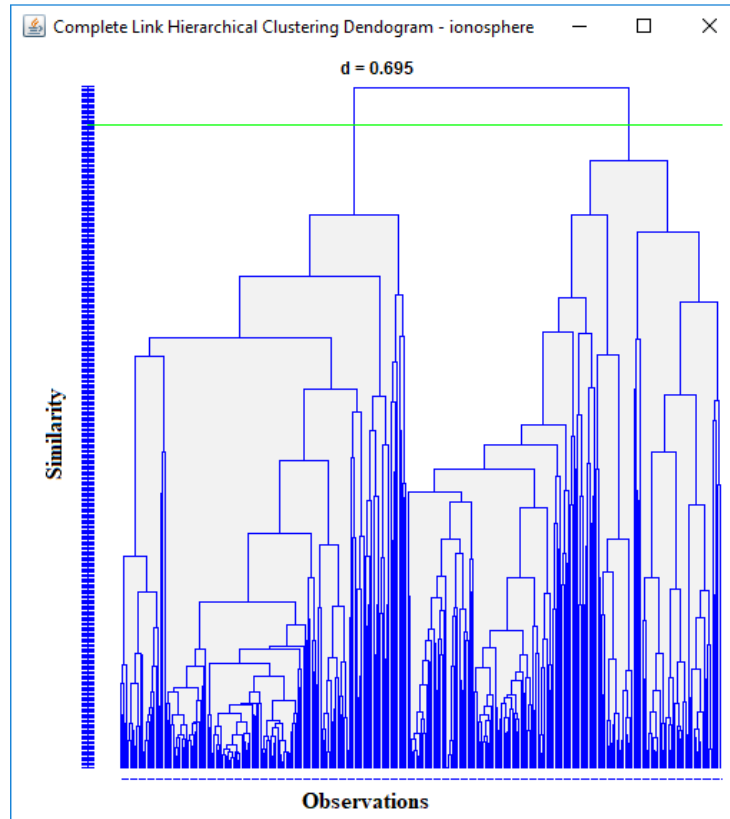


Figure 5-3 Dendrogram for clustering Ionosphere dataset, two clusters appeared

Table 5-1 presents obtained results based on the first case study. The table shows the classification accuracy, indicator of Error/epoch and the Training time for different numbers of hidden neurons with two hidden layers. Through the results obtained from Table 5-1, it is possible to evaluate the performance of the ANN architecture. Using the criteria presented above it has been identified the possible quantity of hidden layers. The projected method agrees with the quantity of hidden layers presented in the table in accordance with the first case study.

Table 5-1 The Value of classification Accuracy and Errors for First Case Study (two hidden layers)

Hidden neurons	Accuracy [%]	Error / Epoch	Training time
3	97.4359	0.0248584	38.37
7	99.1453	0.0085564	26.32
10	99.4302	0.0057253	25.48
15	99.4302	0.005731	16.35
20	99.4302	0.0058149	21.42
25	99.1453	0.0085796	34.4
30	99.4302	0.0057444	22.11
35	99.4302	0.0057372	27.57
40	99.1453	0.0085842	24.96
45	99.1453	0.0085893	22.25
47	99.1453	0.0085904	27.9

Figure 5-4 presents the percentage of classification accuracy for different numbers of hidden neurons with two hidden layers:

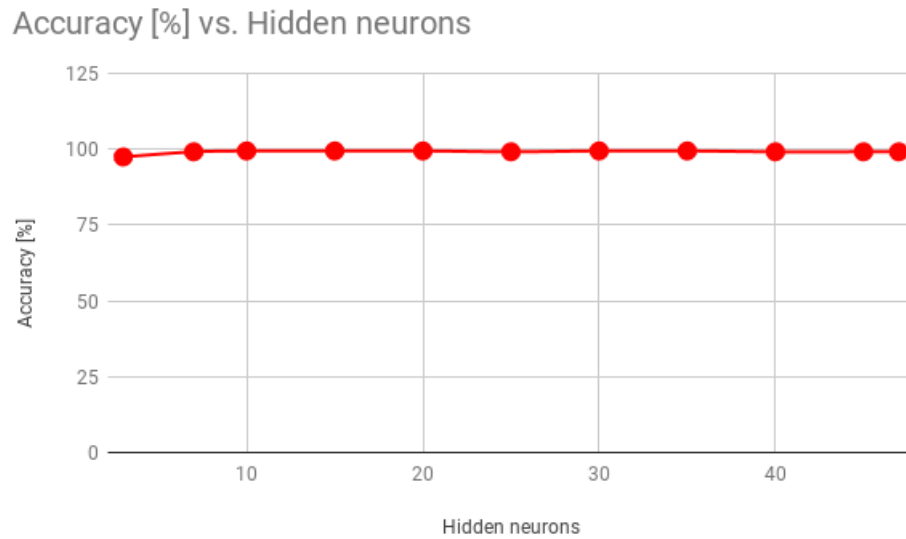


Figure 5-4 Percentage of classification accuracy for different numbers of hidden neurons with two hidden layers.

Figure 5-5 presents error values for various quantity of hidden units with two hidden layers:

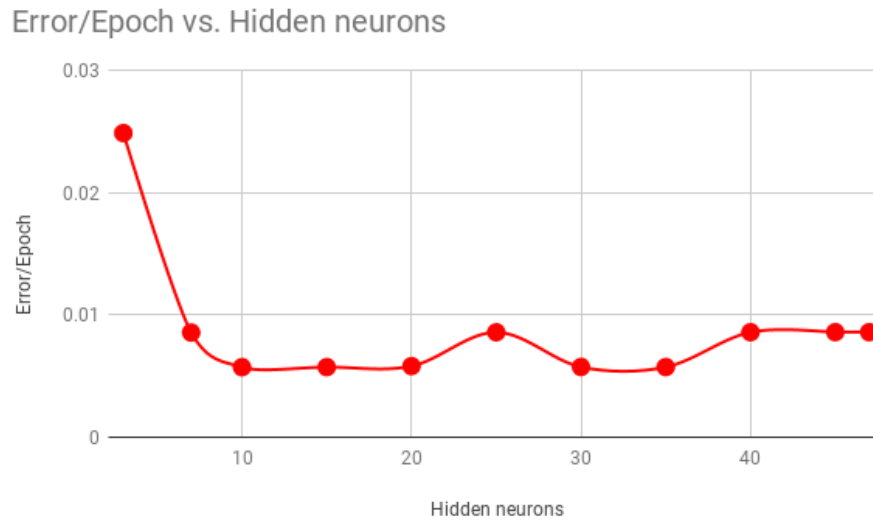


Figure 5-5 Error indicator for various quantity of hidden units with two hidden layers.

Figure 5-6 presents Training time for different numbers of hidden neurons with two hidden layers:

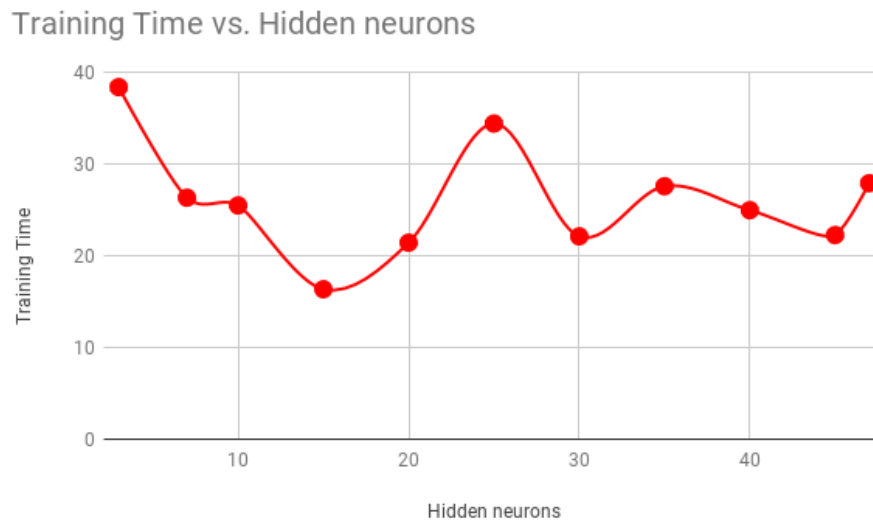


Figure 5-6 Training time for different numbers of hidden neurons with two hidden layers.

As observed from Table 5-1, the highest value of classification accuracy obtained using 2 hidden layers and 10 hidden neurons with a percentage equal to 99.43% and a value of error/epoch equal to 0.0057253, which represent the lower error value.

Figure 5-4, Figure 5-5 and Figure 5-6 present the value of the percentage of classification, error/epoch and training time respectively. The value of error/epoch decreases for a high quantity of hidden units.

The case study represented by Maximum Linkage for clusters distance, Manhattan distance, and Standard normalization impose two layers and with ten hidden units to train Ionosphere Dataset.

5.6.3. Second Case Study

The second case study depends on Complete Linkage for clusters distance, Euclidean distance, and Standard normalization.

Figure 5-7 presents the obtained Dendrogram for clustering of training dataset Ionosphere. For the selected value of reference distance $d = 0.72$, it is obtained one cluster based on the second case study and taking into consideration the criteria described above.

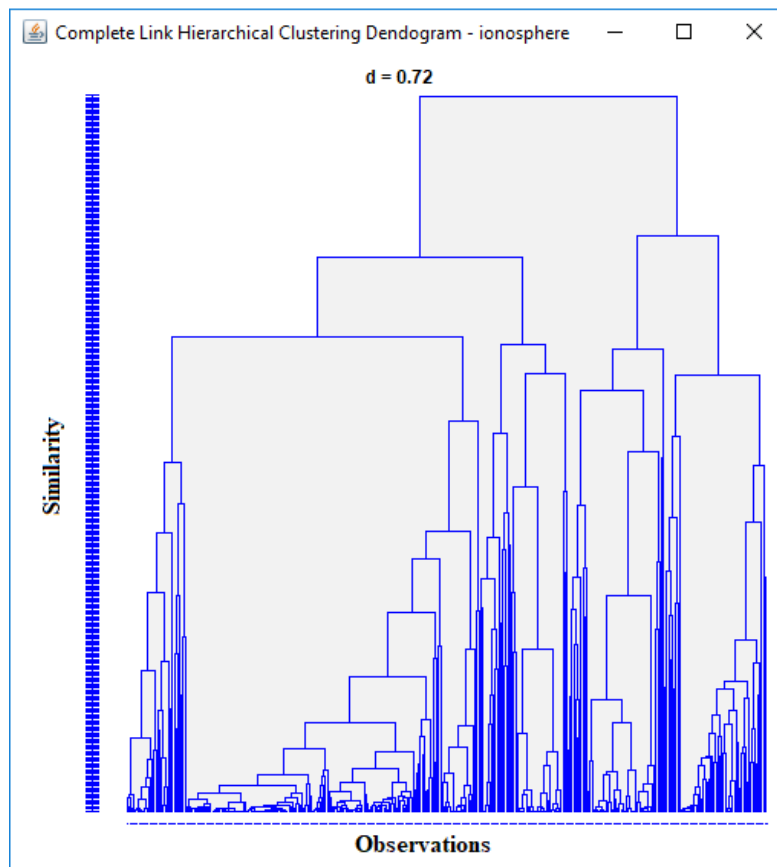


Figure 5-7 Dendrogram for clustering Ionosphere dataset, one cluster appeared

Table 5-2 presents the obtained results based on the second case study. The table shows the values of classification and the values of error/epoch for various quantity of hidden layers taking into consideration the criteria presented above to define the hidden layers quantity.

Table 5-2 The Value of classification Accuracy and Errors for Second Case Study (One hidden layer)

Hidden neurons	Accuracy [%]	Error / Epoch	Training time
3	99.4302	0.0059634	33.01
7	99.7151	0.0029575	28.4
10	99.7151	0.0029863	20.18
15	99.7151	0.0029253	25.49
20	99.7151	0.0018855	20.93
25	99.4302	0.0057776	25.34
30	99.7151	0.0029516	26.51
35	99.7151	0.0028937	37.84
40	99.7151	0.0029467	23.25
45	99.1453	0.0086265	49.23
50	100	0.0007232	31.08

Figure 5-8 presents the percentage of classification accuracy for different numbers of hidden neurons with one hidden layer:

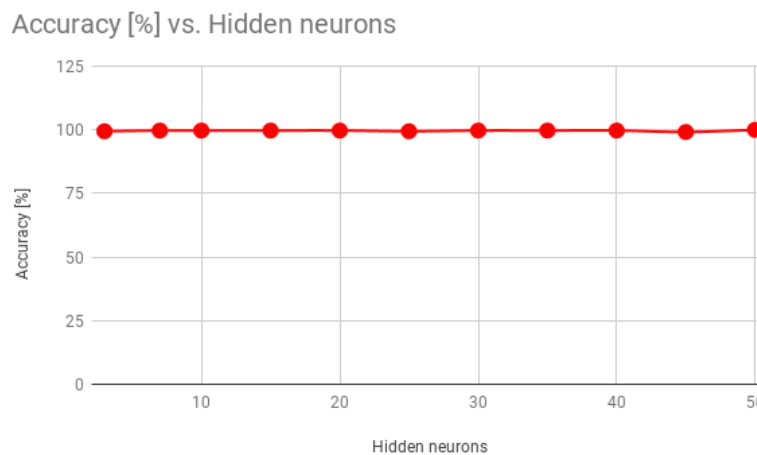


Figure 5-8 Percentage of classification accuracy for different numbers of hidden neurons with one hidden layer.

Figure 5-9 presents error values for various quantity of hidden units with one hidden layer:

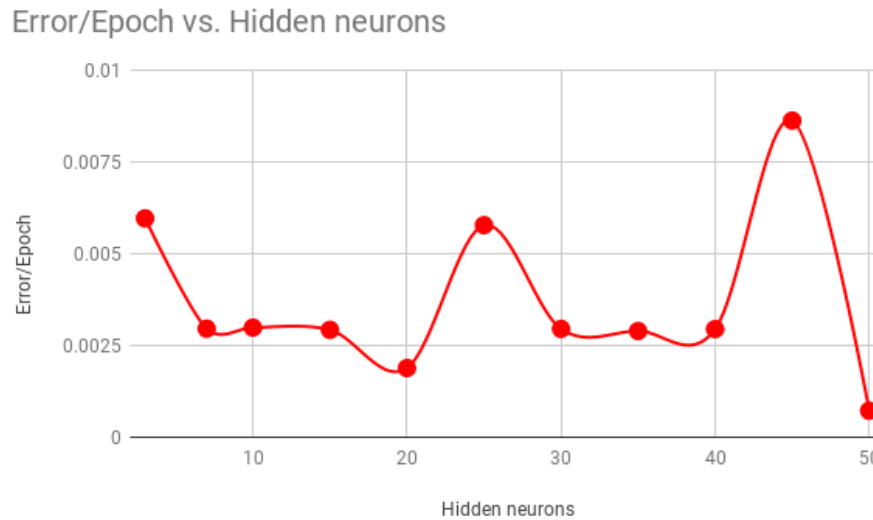


Figure 5-9 Error indicator for various quantity of hidden units with one hidden layer

Figure 5-10 presents Training time for different numbers of hidden neurons with one hidden layer:

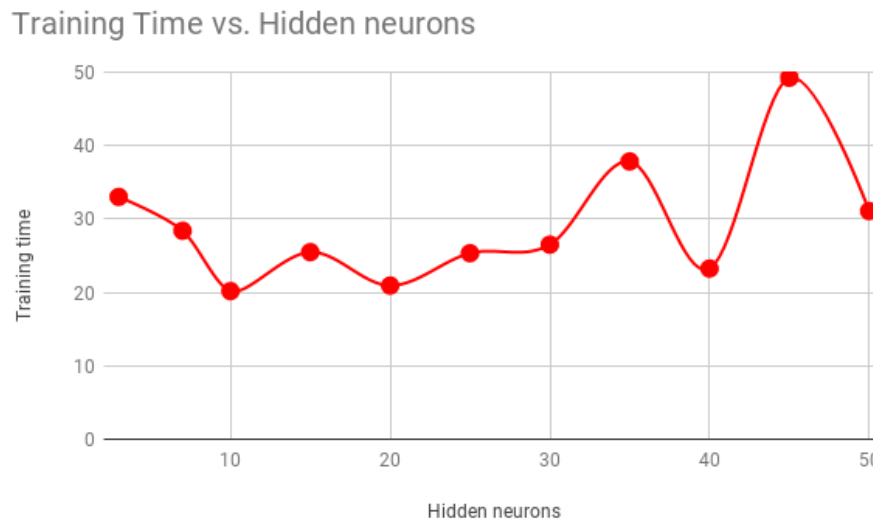


Figure 5-10 Training time for different numbers of hidden neurons with one hidden layer.

As observed from Table 5-2, the highest value of classification accuracy obtained using one hidden layer with a percentage equal to 100% and a value of error equal 0.0007232, which represent the lower error value. The smallest value of error acquired with 50 hidden neurons.

Figure 5-8, Figure 5-9 and Figure 5-10 present the value of the percentage of classification, error/epoch and training time respectively. The least value of error/epoch obtained for 50 hidden neurons.

The case study represented by Complete Linkage for clusters distance, Euclidean distance, and Standard normalization impose 1 hidden layer and 50 hidden neurons to train Ionosphere Dataset.

5.6.4. Third Case Study

The third case study depends on Average Linkage for clusters distance, Manhattan distance, and Standard normalization.

Figure 5-11 presents the obtained Dendrogram for clustering of training dataset Ionosphere. For the selected value of reference distance $d = 0.451$, it is obtained three clusters based on the third case study.

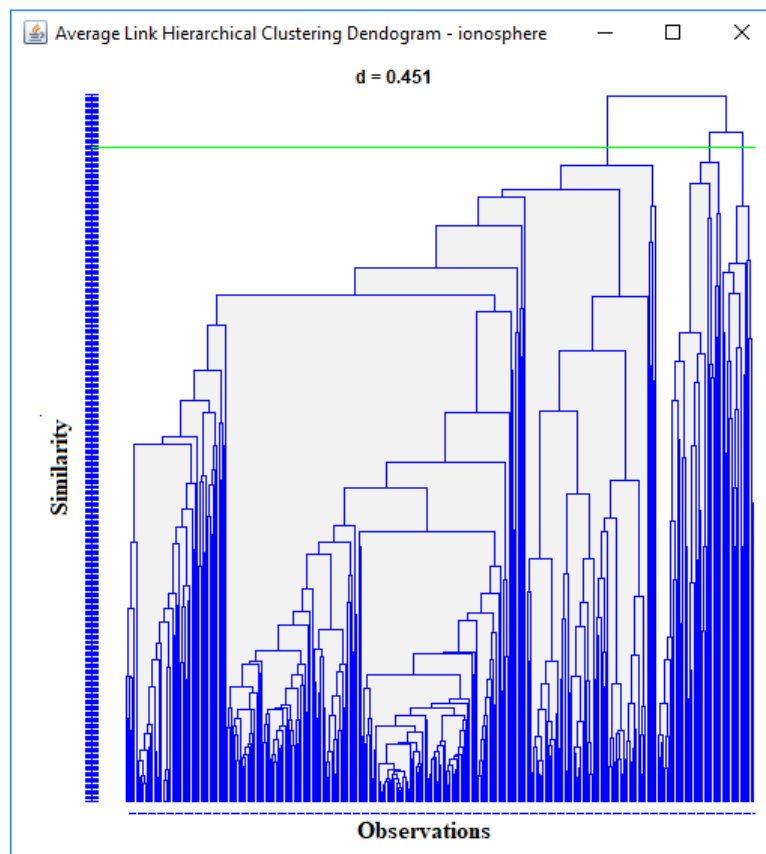


Figure 5-11 Average Linkage (Manhattan) three clusters appeared

Table 5-3 presents the obtained results based on the third case study. The table shows the values of classification and the values of error/epoch for different numbers of hidden layers taking into consideration the criteria presented above to define the hidden layers number.

Table 5-3 The Value of classification Accuracy and Errors for Third Case Study (three hidden layers)

Hidden neurons	Accuracy [%]	Error / Epoch	Training time
3	95.1567	0.00455487	19.29
7	98.8604	0.0112355	21.27
10	99.1453	0.0084975	24.6
15	99.4302	0.0056985	22.31
20	99.4302	0.0057139	20.07
25	99.1453	0.0085636	42.3
30	99.1453	0.0085618	30.19
35	99.1453	0.0085637	19.24
40	99.1453	0.0085645	26.74
45	99.1453	0.0085626	22.9
50	99.1453	0.0085648	26.82

As observed from Table 5-3, the best results obtained is for three hidden layers for the present case study with a percentage of accuracy equal to 99.43%.

Figure 5-12 presents the percentage of classification accuracy for different numbers of hidden neurons with three hidden layers:

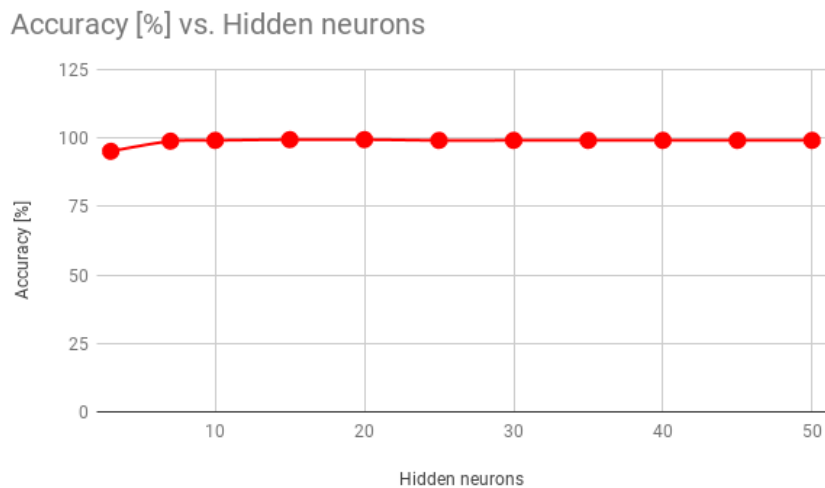


Figure 5-12 Percentage of classification accuracy for different numbers of hidden neurons with three hidden layers.

Figure 5-13 presents error values for various quantity of hidden units with three hidden layers:

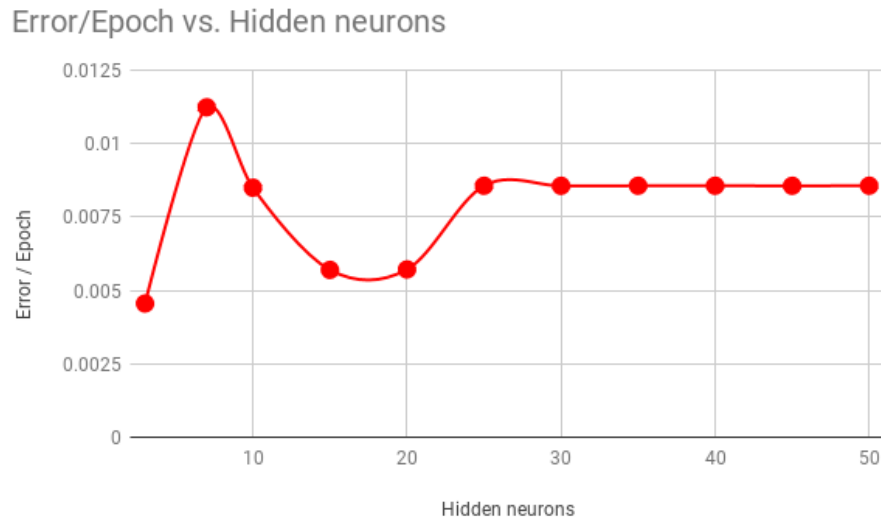


Figure 5-13 Error indicator for various quantity of hidden units with three hidden layers.

Figure 5-14 presents Training time for different numbers of hidden neurons with three hidden layers:

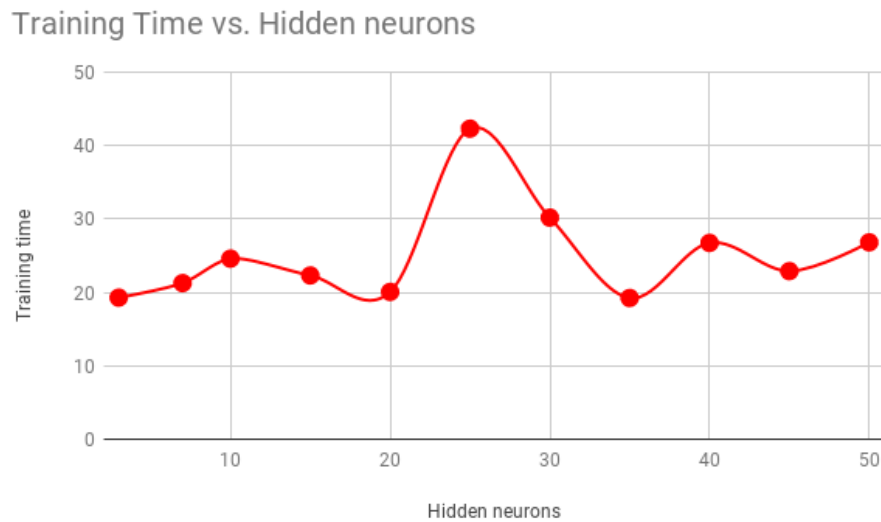


Figure 5-14 Training time for different numbers of hidden neurons with three hidden layers.

5.6.5. Fourth Case Study

The fourth case study depends on Average Linkage for clusters distance, Euclidean distance, and Standard normalization.

Figure 5-15 presents the obtained Dendrogram for clustering of training dataset Ionosphere. For the selected value of reference distance $d = 0.377$, it is obtained four clusters based on the third case study.

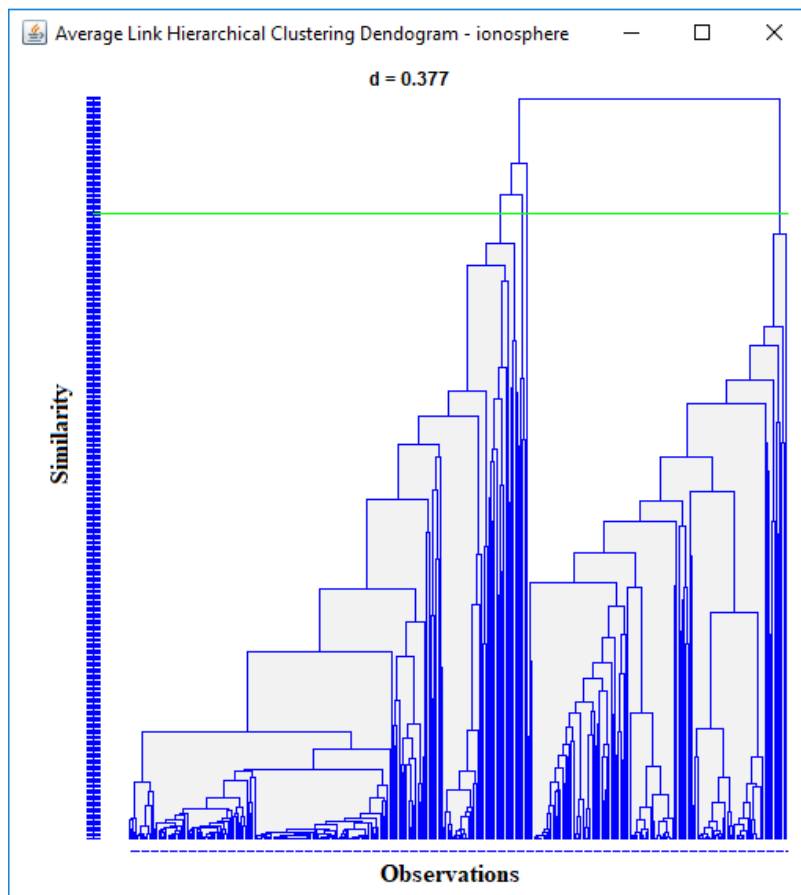


Figure 5-15 Average Linkage (Euclidean).

Table 5-4 presents the obtained results based on the fourth case study. The table shows the values of classification and the values of error/epoch for various quantity of hidden layers taking into consideration the criteria presented above to define the hidden layers number.

Table 5-4 The Value of classification Accuracy and Errors for Fourth Case Study (Four hidden layers)

Hidden neurons	Accuracy [%]	Error / Epoch	Training time
3	64.1026	0.2335309	25.23
7	64.1026	0.2335322	20.79
10	64.1026	0.2335328	19.35
15	64.1026	0.2335327	24.22
20	64.1026	0.2335322	24.32
25	64.1026	0.2322227	18.68
30	64.1026	0.2330977	19.31
35	98.8604	0.0113529	19.56
40	98.8604	0.0114421	41.09
45	99.1453	0.0086075	22.51
50	98.5755	0.0138187	25.95

Figure 5-16 presents the percentage of classification accuracy for different numbers of hidden neurons with four hidden layers:

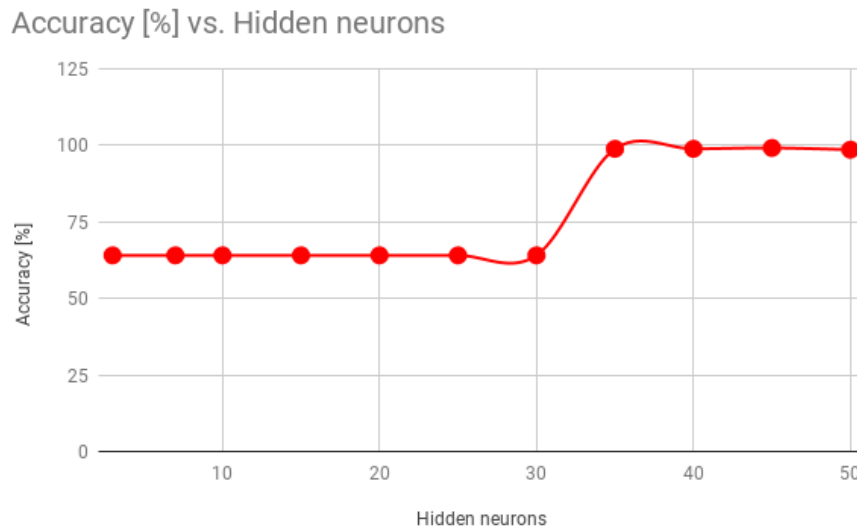


Figure 5-16 Percentage of classification accuracy for different numbers of hidden neurons with four hidden layers.

Figure 5-17 presents error values for various quantity of hidden units with four hidden layers:

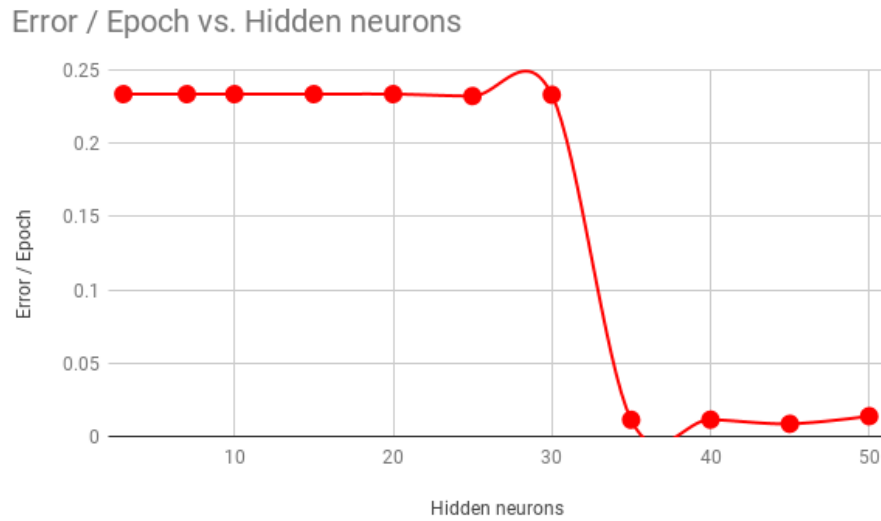


Figure 5-17 Error indicator for various quantity of hidden units with four hidden layers

Figure 5-18 presents Training time for different numbers of hidden neurons with four hidden layers:

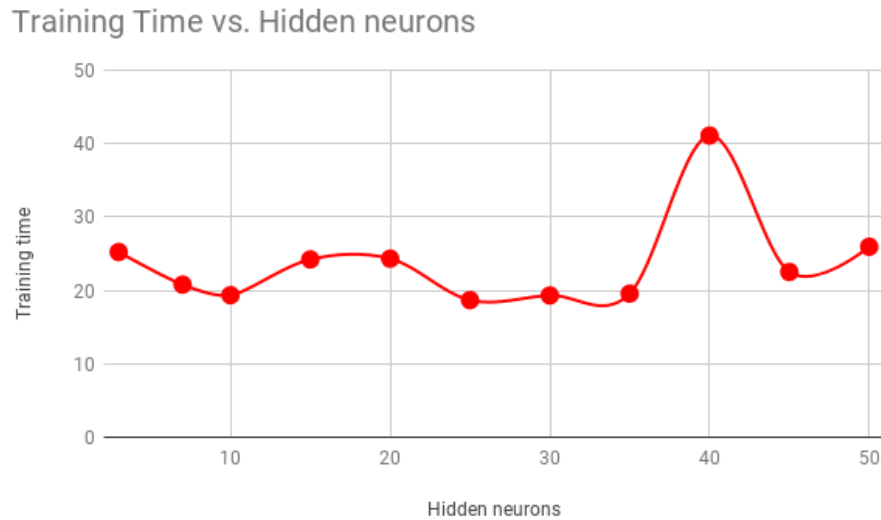


Figure 5-18 Training time for different numbers of hidden neurons with four hidden layers

As observed from Table 5-4, the highest value of classification accuracy obtained using four hidden layers with a percentage equal to 99.14% and a value of error/epoch equal to 0.0086075, which represent the lower error value. For a small number of hidden units, the classification accuracy is low while accuracy increase for a large number of hidden units as well the values of errors decrease.

Figure 5-16, Figure 5-17 and Figure 5-18 present the value of the percentage of classification, error/epoch and training time respectively. For a few numbers of hidden neurons, the classification accuracy is low while accuracy increase for a large number of hidden units. Error/epoch decrease for a large number of hidden units.

The case study represented by Average Linkage for clusters distance, Euclidean distance, and Standard normalization impose 4 hidden layers and 45 hidden neurons to train Ionosphere Dataset.

5.6.6. Analysis of Results

Based on the comparison of obtained results it is clear that the best neural network architecture to train Ionosphere dataset have 1 hidden layer and 50 hidden neurons. The optimal architecture obtained using distance metrics Euclidean distance, Complete Linkage for clusters distance, and Standard normalization. The classification accuracy obtained using the defined architecture is 100% and with a value equal to 0.0007232 for error/epoch.

Percentage of classification accuracy for Ionosphere Data set

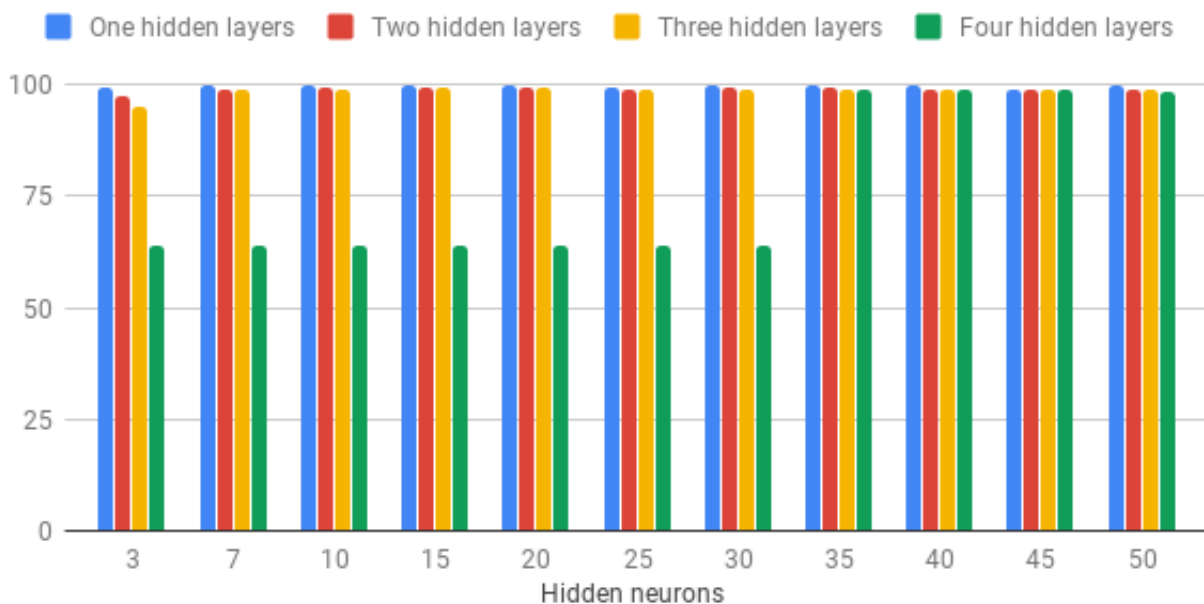


Figure 5-19 Percentage of classification accuracy for Ionosphere Data set.

Ionosphere dataset imposes a specific distance measure based on the type of data. A comparison of distance measures such as clusters distance, type of normalization and type of distance is necessary until we can define the optimal neural network architecture.

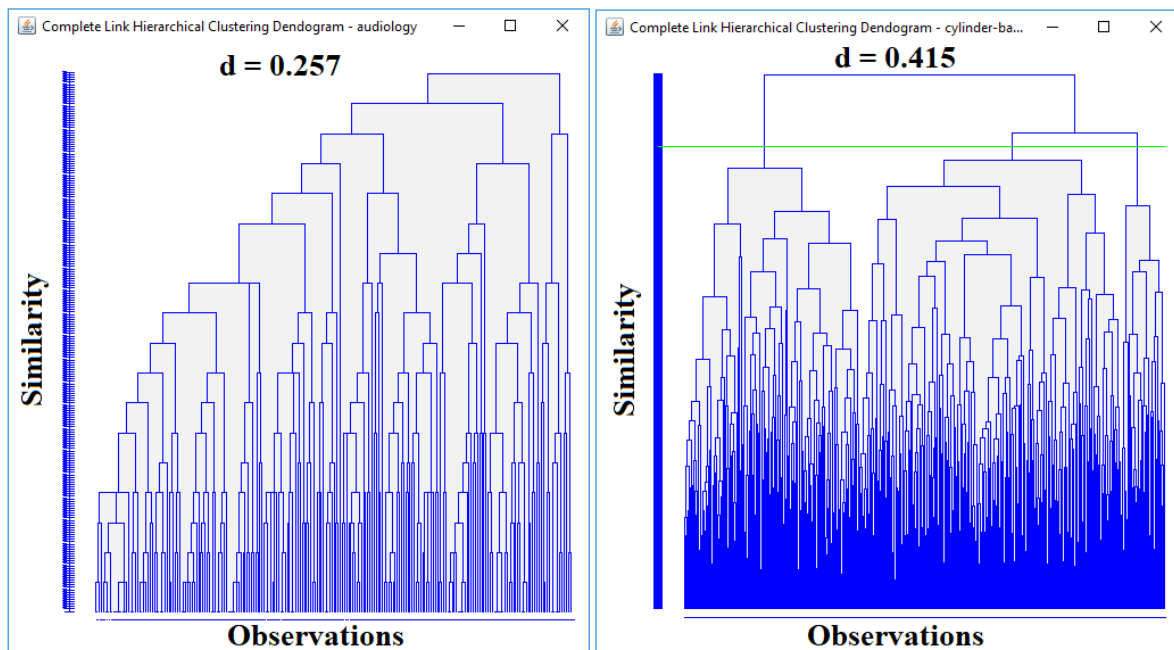
5.6.7. Prove of Generalization Capabilities of the projected Method

The projected method can evaluate the complexity of the considered problem through grouping of the learning data. For this reason, the projected method can adapt to different type of dataset unlike other methods, which gives results only under certain condition and for specific issues. Accordingly, the proposed method acquired more generalization capabilities compared to other methods.

To prove the generalization capabilities of the proposed method we select a set of datasets with different numbers of instances and numbers of features. To cover different levels of complexity we compare the classification accuracy and error/epoch of the considered datasets to results obtained with different numbers of hidden layers until we can confirm the effectiveness of the projected method.

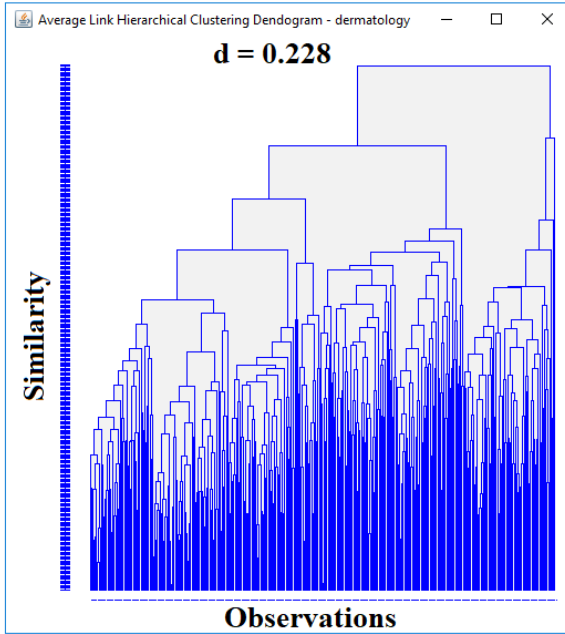
The number of hidden units is obtained based on the formulas (2.4)-(2.9) presented above. The formula, which has the superior value of accuracy compared to the six formulas, will be used in the comparison.

Figure 5-20 presents Dendrogram for datasets and the quantity of clusters corresponding based on the projected method.

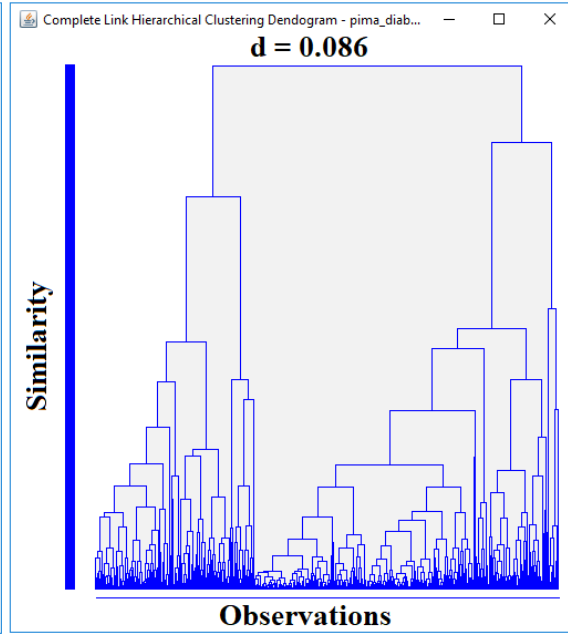


(i) Audiology dataset: one cluster for Standard Normalization, Manhattan Distance, and Complete Linkage

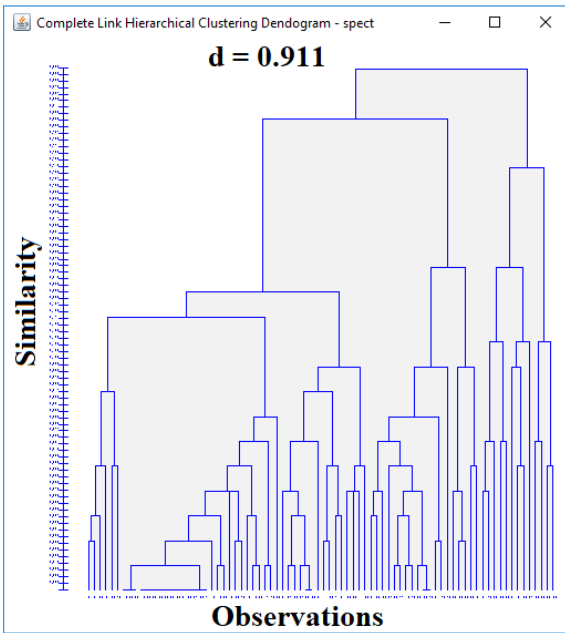
(ii) Cylinder bands dataset: Three clusters for Standard Normalization, Euclidian Distance, and Complete Linkage



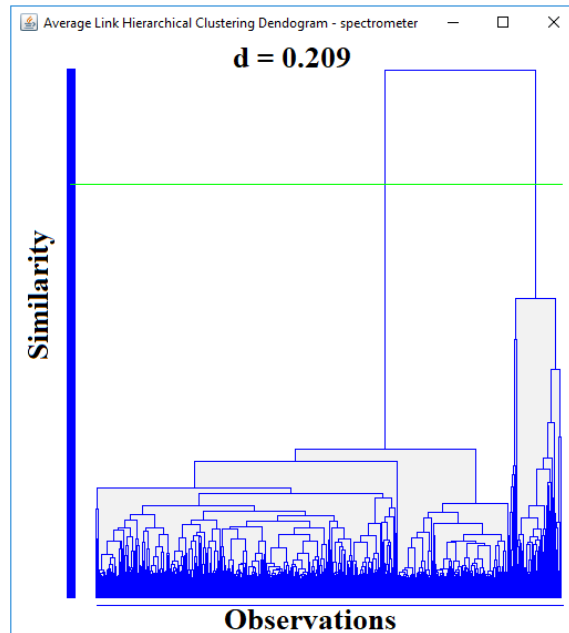
(iii) Dermatology dataset: one cluster for Standard Normalization, Manhattan Distance, and Average Linkage



(iv) Diabetes dataset: one cluster for Standard Normalization, Euclidian Distance, and Complete Linkage



(v) SPECT Heart dataset: one cluster for Z-score Normalization, Manhattan Distance, and Complete Linkage



(vi) Spectrometer dataset: two clusters for Standard Normalization, Euclidian Distance, and Average Linkage

Figure 5-20 Dendrogram for datasets.

Figure 5-20 presents the obtained Dendrogram for clustering of the selected training datasets. The values of reference distance are selected based on the best case study suitable with each dataset and in accordance with the criteria described above concerning at least 90% of items of the dataset are grouped and by increasing the chosen reference distance dose note result a change on the number of clusters obtained.

Table 5-5 shows the obtained quantity of hidden layers based on the projected method.

Table 5-5 Quantity of Hidden Layers based on the projected Method

Datasets	Number of hidden layers
Ionosphere	1
Audiology	1
Cylinder bands	3
Dermatology	1
Diabetes	1
SPECT Heart	1
Spectrometer	2

Table 5-6 presents the obtained results for various quantity of hidden layers and the results of the projected method. The table shows the values of classification accuracy.

Table 5-6 The Values of classification Accuracy

Datasets	1 Hidden layer	2 Hidden layers	3 Hidden layers	4 Hidden layers	Proposed method
Ionosphere	99.7151	99.7151	99.7151	98.2906	99.7151
Audiology	98.2301	87.6106	46.4602	25.2212	98.2301
Cylinder bands	42.2222	65.7407	69.4444	42.2222	69.4444
Dermatology	96.9945	96.7213	93.4426	30.6011	96.9945
Diabetes	78.776	78.3854	77.8646	65.1042	78.776
SPECT Heart	93.75	91.25	80	67.5	93.75
Spectrometer	28.8136	39.7363	19.774	10.3578	39.7363

The quantity of hidden layers defined using the projected method achieve the superior value of accuracy for most of the datasets.

Table 5-7 presents the obtained results for various quantity of hidden layers and the results of the projected method. The table shows the values of errors.

Table 5-7 Errors/Epoch

Datasets	1 Hidden layer	2 Hidden layers	3 Hidden layers	4 Hidden layers	Proposed method
Ionosphere	0.003177	0.0028845	0.0028658	0.0142633	0.003177
Audiology	0.0000181	0.0001573	0.0141475	0.0360344	0.0000181
Cylinder bands	0.2776212	0.247546	0.2475429	0.2475326	0.2475429
Dermatology	0.0000102	0.000033	0.0000371	0.0411695	0.0000102
Diabetes	0.0000254	0.000093	0.0000191	0.1340351	0.0000254
SPECT Heart	0.0000086	0.0000411	0.06288	0.0136035	0.0000086
Spectrometer	0.1863167	0.0002018	0.0000191	0.0002117	0.0002018

Table 5-6 and Table 5-7 show results obtained from various types of datasets with different numbers of instances and numbers of features. The number of hidden layers obtained using the proposed method get the lowest value of error/epoch for most of the datasets. The anticipated method familiarizes with the data set to produce the best outcome no matter how bigger or smaller the dataset is.

The results prove the generalization capabilities of neural network architectures defined using the proposed method.

5.7. Conclusion

Usage of clustering techniques on the training data can define the optimum quantity of hidden layers. The precision is required when the selection of the convenient clustering distance measures which necessitate a comparative study to perform a perfect clustering of training dataset where it affects positively on the accuracy of results.

The comparative study to determine the suitable distance measures gives to the proposed method more performance to handle different types of data which increase generalization capabilities of the neural network architectures.

The proposed method proves the capability to deal with various types of datasets, which enhance the generalization capabilities of the projected method.

The projected method facilitates the ANN structure design and reduces the building time.

Chapter VI

6. Comparison of the anticipated method vs. traditional technique

MLP architecture is an area of concern in the identification of Neural Network. As stated earlier, to date, there exists no verified and accepted general theory explaining. There is no theory explaining the neural network architecture for MLP when the difficulty of the problem is not known. Adopted approaches include “evolutionary methods, exhaustive search, and Growing and Pruning algorithms. Despite their use, most of them consume a lot of time and have major limitations” [147].

The more the number of hidden units was, the capacity of a neural network to solve problems increase but it gives rise to the time consumption. Decreasing the units quantity can cause an improves in the generalization of the neural network, but the neural network becomes weak or unable to meet the required needs [148].

Here is a selection of the most common methods used for designing the structure of ANN:

- Rules of Thumb: There is a lot of rules of thumb used by the designer of neural network structure among the most used rules of thumb the unit's quantity must be in the interval of the quantity of input/output units. Another rule of thumb is that the unit's quantity calculated through formula: $(\text{inputs units} + \text{outputs units}) * (2/3)$. Another rule of thumb is that the unit's quantity must be below the twice of the quantity of input units [149].
- Trial and Error: Does not achieve perfect results only occasionally and by chance, also called exhaustive search [150].
- Exhaustive Search: Seeking to search through all neural network topologies to select the neural network structure with less generalization error. This method takes a long time to determine the optimal structure.
- Growing Algorithms: Seeking to search in all neural network structures to select the neural network structure with less generalization error. This method differs from the exhaustive search in that searching stops when it is noticed that the general error does not exhibit any possibility. In this method, searching stops when no significant change is noticed.
- Pruning Algorithms: after training a large neural network this method tries to evaluate the weights to determine the level of importance of each weight. Weights with low importance must be pruned and then repeat the task until we obtain the best structure. The analysis of weights takes a long time to determine the optimal structure where it represents a disadvantage for this method.

In this chapter, several empirical formulas used to identify the quantity of hidden units of an MLP will be examined and tested on a different training dataset. These classical methods will be compared to the proposed method as well. The formulas below are selected for the comparison:

- Formulas (2.4)-(2.9) previously mentioned.
- Formula (6.1) : Quantity of hidden nodes must be between the input units and the output units' quantities.
- Formula (6.2) : Quantity of hidden nodes must be below the twice of input units.

Table 6-1 shows Properties of learning data set utilized.

Table 6-1 Configuration of the ANN

Dataset	Sonar	ECG	P-wave	QRS	T-wave	Landsat	Glass	Segmentation	Waveform
Input neurons	60	6	2	2	2	4	9	19	21
Output neurons	2	16	16	16	16	7	7	7	3
Training items	208	452	452	452	452	6435	214	2310	5000

Table 6-2 shows the quantity of hidden nodes through the traditional methods:

Table 6-2 Quantity of hidden nodes through the traditional methods

Dataset	Formula (2.4)	Formula (2.5)	Formula (2.6)	Formula (2.7)	Formula (2.8)	Formula (2.9)	Formula (6.1)	Formula (6.2)
Sonar	31	5.75	41	0	6	45	$2 < x < 60$	$x < 120$
ECG	11	3.32	14	2	89	32.26	$6 < x < 16$	$x < 12$
P-wave	9	3	12	2	235	30.26	$2 < x < 16$	$x < 4$
QRS	9	3	12	2	235	30.26	$2 < x < 16$	$x < 4$
T-wave	9	3	12	2	235	30.26	$2 < x < 16$	$x < 4$
Landsat	5	2.24	7	58	1615	85.22	$4 < x < 7$	$x < 8$
Glass	8	2.83	10	1	31	22.63	$9 < x < 7$	$x < 18$
Segmentation	13	3.61	17	8	129	62	$19 < x < 7$	$x < 38$
Waveform	12	3.46	16	20	241	82.71	$21 < x < 3$	$x < 42$

Table 6-3 shows the quantity of neurons generated by the projected method. The obtained quantity of hidden units will be evenly distributed on the hidden layers:

Table 6-3 Quantity of hidden nodes through the projected method

Dataset	Neurons	layers
Sonar	35	2
ECG	20	1
P-wave	10	2
QRS	21	1
T-wave	20	3
Landsat	24	2
Glass	207	2
Segmentation	75	1
Waveform	74	1

6.1. Traditional and proposed method percentage of accuracy comparison

Table 6-4 below shows the percentage of classification accuracy obtained using the classical methods compared to the proposed method.

Table 6-4 Percentage of accuracy indicator in both projected and traditional methods

Dataset	Formula (2.4)	Formula (2.5)	Formula (2.6)	Formula (2.7)	Formula (2.8)	Formula (2.9)	Formula (6.1)	Formula (6.2)	Proposed method
Sonar	81.25	80.76	80.76	74.5192	81.73	81.25	81.7308	81.25	82.2115
ECG	57.9646	59.292	59.292	60.8407	59.292	59.292	57.9646	59.5133	60.8407
P-wave	53.7611	53.9823	53.9823	53.9823	53.9823	54.2035	53.9823	53.9823	54.2035
QRS	59.0708	58.8496	58.8496	59.5133	59.9558	58.8496	59.5133	59.5133	60.177
T-wave	53.9823	54.2035	53.9823	54.2035	56.1947	56.6372	56.6372	54.2035	56.6372
Landsat	76.7535	50.3006	80.5611	76.7535	82.5651	82.3647	84.8703	85.3213	85.6595
Glass	85.0467	72.8972	82.7103	61.6822	97.6636	94.3925	85.0467	82.7103	98.5981
Segmentation	97.5325	95.2381	97.9221	95.8442	97.6623	98.8312	95.7576	98.8312	99.1342
Waveform	95.74	89.28	97.32	97.86	98.22	98.4	96.82	97.86	98.6

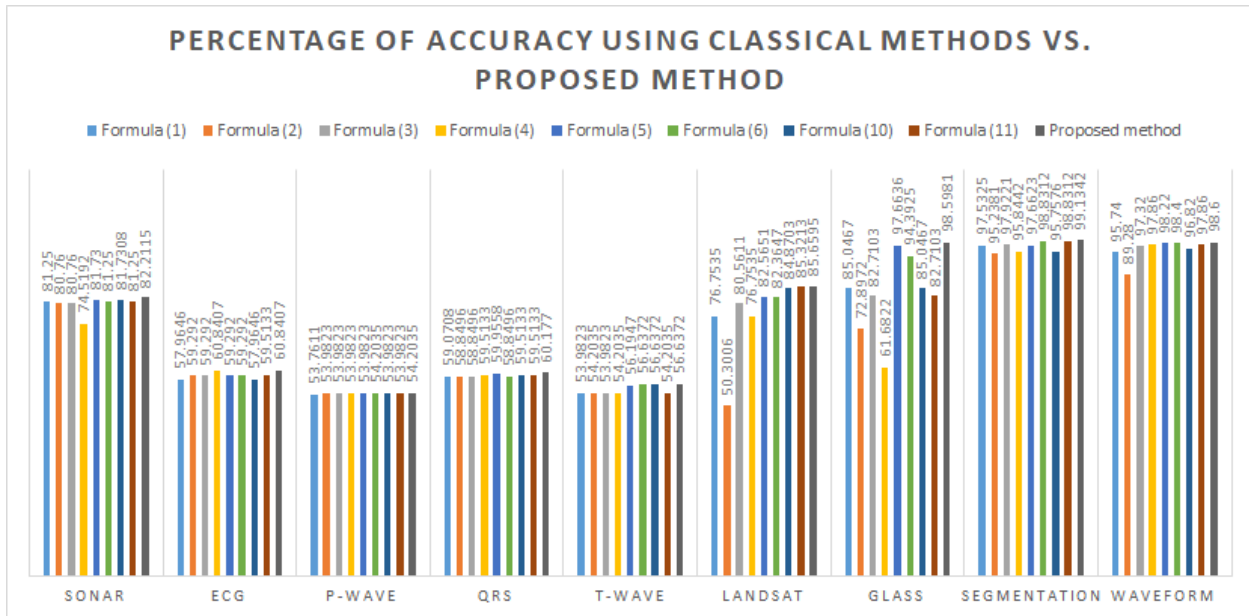


Figure 6-1 Percentage of accuracy indicator in both projected and traditional methods

As observed from Table 6-4 and Figure 6-1, the anticipated method provides the best results in almost all datasets concerning the accuracy proportion. On the other hand, traditional methods, in some cases, produce good results depending on the set database. Case in point; method (2.7) has a high accuracy score for ECG data combination and lower accuracy score for sonar/Landsat sets; method (2.8) indicates a good accuracy level compared to other methods in Glass data grouping. It also scores low accuracy proportion for ECG data combination. Method (2.9) record average score in a most and good performance in “Segmentation, Glass and waveform” datasets.

6.2. Error/epoch indicator in both projected and traditional methods

As for Table 6-5 and Figure 6-2, the projected method records low Error/epoch score. Traditional approaches, in some instances, have good scores when methods “(2.8), (2.9) and (6.1)” are adopted. Such results are mostly being accepted, leaving out other traditional methods. Methods “(2.5) and (2.7) possess a high error/epoch score.

Table 6-5 shows the relationship in error/epoch when either the traditional or proposed method is applied:

Table 6-5 Error/epoch indicator in both projected and traditional methods

Dataset	Formula (2.4)	Formula (2.5)	Formula (2.6)	Formula (2.7)	Formula (2.8)	Formula (2.9)	Formula (6.1)	Formula (6.2)	Proposed method
Sonar	0.0145937	0.0356288	0.0136024	0.046503	0.0134245	0.0145827	0.004859	0.014482	0.004876
ECG	0.0316269	0.0346926	0.0314092	0.0352489	0.0315745	0.030976	0.031865	0.033147	0.035073
P-wave	0.0421722	0.0421824	0.0421695	0.0421824	0.0422299	0.0421805	0.042169	0.042182	0.04209
QRS	0.0351015	0.0357337	0.0350602	0.0361089	0.0354633	0.0351336	0.03524	0.036108	0.035482
T-wave	0.0415934	0.0424934	0.0413225	0.0424079	0.0392774	0.0392845	0.039138	0.042407	0.039138
Landsat	0.0465562	0.0877938	0.0373482	0.0465562	0.0285187	0.0280371	0.028926	0.031118	0.026716
Glass	0.0368909	0.0612921	0.0360473	0.0843718	0.0033422	0.0101347	0.03689	0.036047	0.002673
Segmentation	0.0050226	0.0100253	0.0048926	0.0081031	0.0033338	0.0027053	0.009412	0.003655	0.002399
Waveform	0.0266691	0.0522059	0.0175918	0.014369	0.0117388	0.0106167	0.020263	0.01436	0.009346

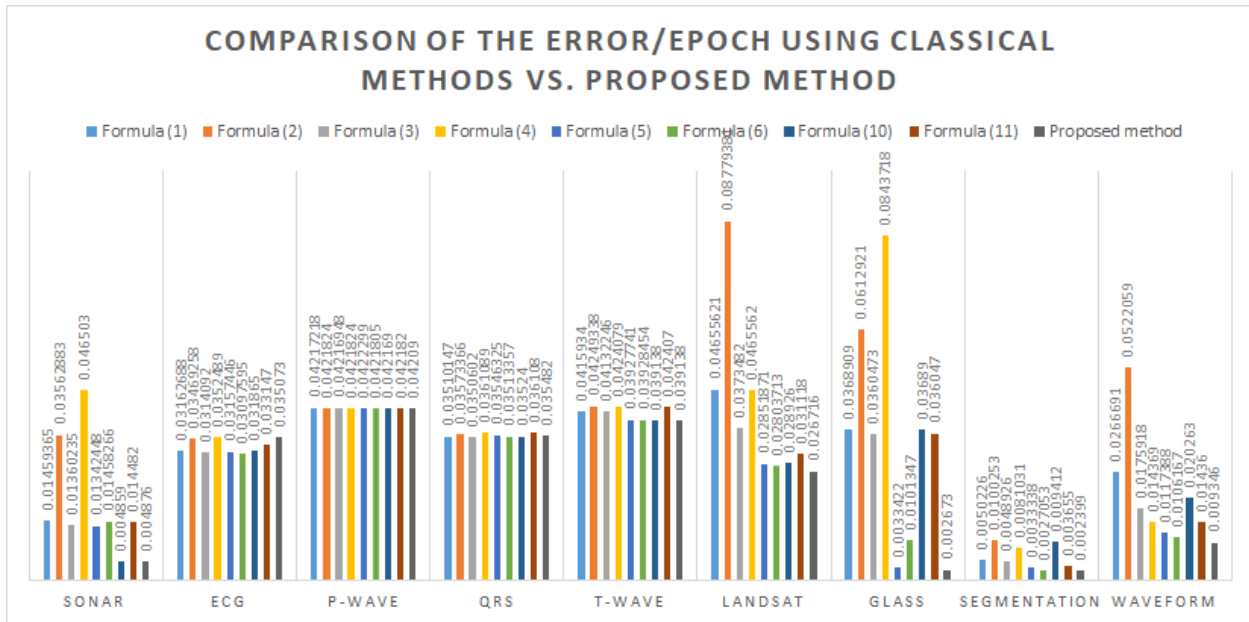


Figure 6-2 Error/epoch indicator in both projected and traditional methods

6.3. Traditional and proposed method training period comparison

Table 6-6 indicates results for data training in either methods:

Table 6-6 Training period indicator in both projected and traditional methods (seconds)

Dataset	Formula (2.4)	Formula (2.5)	Formula (2.6)	Formula (2.7)	Formula (2.8)	Formula (2.9)	Formula (6.1)	Formula (6.2)	Proposed method
Sonar	9.14	7.45	20.57	12.25	3.83	8.93	8.07	25.04	11.76
ECG	4.72	3.85	5.82	5.06	19.16	13.22	2.25	1.47	0.75
P-wave	6.32	4.66	18.38	4.66	132.69	8.1	18.38	4.66	2.77
QRS	4.17	2.89	24.56	3.17	39.35	7.37	1.5	3.17	1.37
T-wave	3.61	2.54	3.84	0.95	136.74	25.97	2.39	0.95	2.39
Landsat	7.77	5.41	8.51	7.77	490.22	35.35	10.96	7.56	74.25
Glass	31.67	36.72	39.9	30.58	108.91	41.14	31.67	39.9	2523.85
Segmentation	36.12	29.02	83.56	24.06	175.64	88.78	31.09	45.41	99.14
Waveform	70.89	41.34	96.12	119.65	1433.66	458.26	88.94	119.65	1062.75

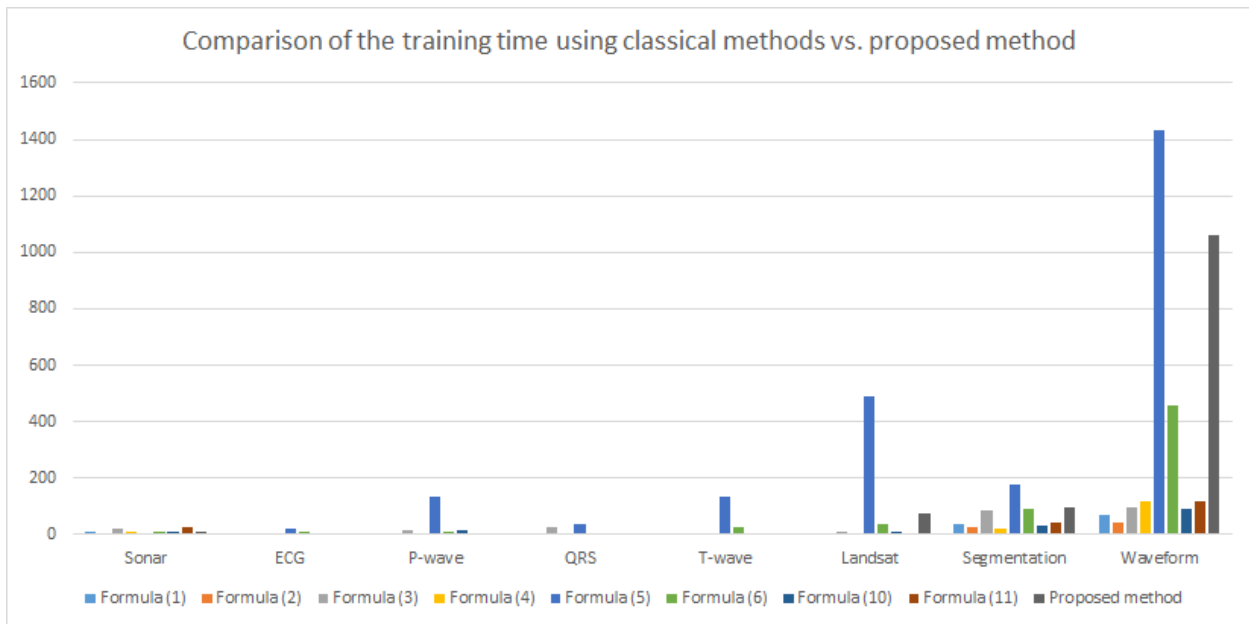


Figure 6-3 Training period indicator in both projected and traditional methods

As observed from

Table 6-6 and Figure 6-3, the projected method achieve good results for QRS, P-wave and ECG.

6.4. Conclusion in the comparisons

From the findings, it is evident that the projected method records a high accuracy score in many datasets than traditional methods. Traditional methods in “(2.4), (2.5), (2.6), (6.1) and (6.2)” has

a good score in minor datasets and with less training. Methods “(2.7), (2.8), (2.9) ” score well in big datasets since they consider the quantity of training items. Method “(2.9) is superior to (2.7) and (2.8)” this can be concluded that square root has some influence on the results and mostly it is positive. Methods “(2.4), (2.5) and (2.6) ” relies on the quantity of input/output neurons hence works best on minor datasets and low performance in bigger datasets with huge problems tackled. On training time, it can be concluded that it is dependent on the magnitude of the dataset and the network’s quantity of neurons, formula notwithstanding. error/epoch results experienced obtained comparable to the accuracy level.

From comparisons between traditional methods and proposed methods shows that the projected method has good score in various datasets. Meaning that it is very flexible with different sets of data as compared to traditional methods. It easily adapts to any set of data, no matter how complex it is or how bigger or smaller it is. In some instances, the data combination can be chosen with a size larger than needed, thereby leading to wrong outcomes when traditional methods are used. This challenge can be overcome by adopting the projected method as it has full focus on the problem complexity tackled regardless of data combination size.

Chapter VII

7. Conclusion

In this thesis, we demonstrated the importance of data mining techniques to identify the architecture of the ANN. We believe that the analysis of the training data of the ANN can lead to the optimal ANN architecture. Patterns discovered in the training dataset can identify the complexity level of the considered problem through it we can define the architecture of the network. In this thesis, we exploit the Data Mining techniques to analyze the training dataset to determine a general method, which adapts with all type of problems considered by the neural network.

7.1. Summary of Contributions

There is no defined theory of explaining MLP neural network composition considering the existing problem's complexity. Renowned approaches are evolutionary algorithms, exhaustive search, Growing and Pruning algorithms and rules of thumb. They have limitations and time consumption.

The ANN is an integral part of data classification. It provides an opportunity for generalizations as per the identified data requiring analysis. Irrespective of neural network design and method of learning adopted, hidden units and layers currently are identified by testing.

Using the proposed method, the design complexity of an ANN structure decrease.

This study proposes and develops a new design strategy for an MLP neural network where make the design of the structure of an MLP neural network unsupervised and helps to reducing the time allocated for network design.

Using the proposed method, the design time and effort to identify the structure of the MLP are reduced. The projected method can make the design simple and easy and possible for a Non-specialist designer.

7.1.1. Clustering techniques to identify the architecture of ANN

The optimal ANN architecture is arrived at using training data clustering outcomes. By adopting the projected clustering procedure, several factors have effects on the neural network's

hidden layers number. Reference distance is the most influential. When it is accurate, the optimum quantity of hidden layers is defined.

Using the projected regression method to identify the architecture of the ANN many factors affects the accuracy of the results. The main factor is Reference Distance which has an important effect compared to the other factors. The value of Reference Distance must be selected precisely based on the criteria defined in the previous chapter.

The comparative study of clustering distance measures lends on the proposed method more effectiveness and accuracy through the perfect selection of the convenient distance measures for grouping technique used to identify the structure of the MLP.

It is evident from this research that Pattern Recognition has a considerable function in the identification of the structure of an ANN.

7.1.2. Linear regression to identify the quantity of hidden units of an ANN

The statistical hypothesis testing proved that the regression model is significant. Hypothesis testing demonstrated that the independent variables identified by hierarchical grouping of learning data have a significant relationship with the dependent variable. Therefore, a relationship between the grouping results and the architecture of the ANN exist. Consequently, the regression model was established.

Through this study, we conclude that the projected regression method to identify the architecture of the ANN in terms of quantity of hidden layers/units is viable.

The model generated using the proposed regression method can be essential in practical applications and in the worst assumptions it can provide the initial quantity of hidden layers/units for an ANN and relying on the information obtained from the training dataset the designer can reduce or increase these numbers.

7.1.3. Generalization Capabilities of ANN Architectures using the projected method

The projected clustering method can define the structure of the neural network with good accuracy results. To improve generalization capabilities of the method it required a precise selection of the convenient clustering distance measures. Therefore, a comparative study of the distance measures used in clustering of the training data is necessary to perform a perfect clustering, which affects positively on the accuracy of the proposed method results.

To increase generalization capabilities of the proposed method in which the performance to handle different types of training data enhance we need a comparative study to determine the suitable distance measures.

The proposed method proved the capability to deal with various types of datasets, which enhance the generalization capabilities of the proposed method.

7.1.4. Contribution of the proposed method

One of the most important contributions of this thesis is developing an algorithm to determine the architecture of the ANN based on the complexity of the considered problem.

In conclusion, the projected method performs better than the traditional method in different sets of data due to its flexibility with different sets of data as compared to classic methods. It easily adapts to even complex sets of data. The projected method is flexible with different sets of data that the traditional methods no matter how complex such a set of data is hence providing good outcomes, no matter the size.

7.2.Future Works

The work presented in this thesis is intended to encourage researchers to use the results of the analysis of training dataset to identify the architecture of the ANN. This method can impose a new insight into the way of building the training dataset. Based on the research carried out in this study, our research results suggest a different way of creating the training dataset. The training dataset must be constructed based on the patterns, which they contain. Many researchers indicate that a large quantity of training data will produce a better generalization. Other researchers indicate that is not always a large dataset will exhibit good generalization [151]. A large training dataset with poor examples cannot give a good generalization and cause a waste of time in the training.

It is important that any future investigations carried out on the training dataset should take into consideration the complexity of the problem to be solved by the neural network, as the patterns extracted using data mining techniques from the training dataset could be the indicator of the performance of the training dataset to increase the generalization ability of network. In this way, we get rid of learning insignificant aspects of training dataset, which cause overfitting of network.

In this way, we can reduce the size of the dataset and increase the learning ability of the network. With this method, we can decrease the training time caused by large datasets.

APPENDICES

A. Learning Neural Network Architectures

Learning neural network architectures is the way to use neural networks to search for the best architecture. Using gradient descent techniques for designing the structure of ANN, we need immense search because of the large number of possible neural building blocks and the immense search space.

Many methods are used to search for the architecture of the neural network among them:

Gradient-based methods: some methods try to use a recurrent neural network to create new architectures and then test them with reinforcement learning. In this paper [62] controller-based methods are used. These methods encode the connectivity and structure of a neural network into a variable-length string and use the recurrent neural network controller to generate new architectures. The “child network” – on the real data will result in an accuracy on a validation set. The validation accuracy is used as a reward signal to train the controller. Which produces better neural networks in the next iterations, as the controller improves in the search over time.

Heuristic search: this paper [152] uses heuristic search to start with a simple neural network structure the progressively add hidden layers and neurons in these layers. This paper [153] relies on which use a recurrent neural network controller to search for architectures.

Genetic search: The Genetic search is an exhaustive search method, which creates different architectures. After that search for the best architecture by trying them one by one. This paper [154] gives an overview of the most prominent methods for evolving ANNs with a special focus on recent advances in the synthesis of learning architectures.

Pruning as network architecture search: pruning of neural network is used as a technique for neural network architecture search. In this paper [155] a number of observations are consistent for multiple network architectures, datasets, and tasks, which imply that:

- 1) Training a large, over-parameterized model is often not necessary to obtain an efficient final model,
- 2) Learned "important" weights of the large model are typically not useful for the small pruned model,
- 3) The pruned architecture itself, rather than a set of inherited "important" weights, is more crucial to the efficiency in the final model, which suggests that in some cases pruning can be useful as an architecture search paradigm.

Many factors affect the ability to learn for a neural network. Among them the training dataset, in order to achieve good learning for a neural network it is necessary to select a relevant dataset from which a neural network learn. Another factor is the architectures of a neural network the determination of this factor should be accurate because of its impact on the ability of learning of a neural network. There are many other factors, which have a significant impact on the ability of learning of a neural network such as the cost function, activation function, Hyper-parameters (Learning rate, momentum factor, regularization parameter...) and so on.

Many problems can be encountered when trying to learn Neural Network architectures represented in:

Training Data: The training dataset must be able to learn a neural network, at the same time avoid the specialization, and improve the generalization of the network. That means dataset can learn neural network for forms presented in the dataset, but the network cannot solve the problem for unseen data, which limit its generalization and ultimately its performance.

Architectures: Currently there is no viable theory to identify the structure of an ANN through the level of complexity of the considered problem. Most often the architecture of an ANN is defined based on the designer experience and exhaustive simulations. Experiments performed for designing the structure of ANN are designed to find a solution for a specific issue, and a good result cannot be obtained only for certain cases. Presently, no available valid general-purpose theories to determine the structure of an MLP neural network.

The cost function, activation function, and Hyper-parameters: The cost function and activation function affect the ability to learn for an ANN. It is not possible to realize a general cost function or a general activation function, which work with all type problems. Each training algorithm required a specific hyper-parameter. To learn Neural Network architectures, it is necessary to tune the hyper-parameters during training. Hyper-parameters can be learning rate, momentum factor, regularization parameter, and so on.

Conclusion: Learning neural network architectures is the way to design a network according to number specifications, which reinforce the ability of learning of a neural network. Learning neural network architectures do not produce as good results as the gradient-based search techniques discussed above.

A.1. Currently used Neural Network Architectures

A large number of Neural Network Architectures was created for different issues each one was designed based on certain specifications to achieve different goals. These specifications are imposed by the problem to be solved by the neural network. In this chapter, we will mention the most efficient architectures.

A.1.1. Perceptron

Perceptions represent the first generation of neural networks. Perceptions is a model consist of one single neuron it presented by Frank Rosenblatt in (1956) [156]. Also called the feed-forward neural network. The use of perceptron is limited but is combined with other neural network architectures to determine new architectures. There are very strong limitations on the ability of learning of a perceptron.

A.1.2. Convolutional Neural Networks

A convolutional neural network (CNN) is a deep neural network is mostly used for analyzing visual imagery. It used back propagation in a feedforward net with many hidden layers, many maps of replicated neurons in each layer, pooling of the outputs of nearby replicated neurons.

A.1.3. Recurrent Neural Networks

Jeffrey Elman (1990) presented recurrent neural networks (RNN) in [157]. Recurrent neural networks are designed to recognize data's sequential characteristics. Recurrent Neural Networks (RNN) are powerful and robust because it is the only neural network, which has an internal memory. Recurrent neural networks can memorize information about the received input that gives it a great ability to predict.

Feed-Forward Neural Networks vs. Recurrent Neural Networks: for Feed-Forward neural network, the information moves from the input layer to the input layer through the hidden layers. Feed-Forward Neural Networks cannot remember anything about what happened in the past, except their training thereby impairing predictability. When the Recurrent Neural Networks makes a decision, it takes into consideration the current input and what it has learned from the inputs it received previously.

Figure 7-1 below illustrates the difference in the information flow between an RNN and a Feed-Forward Neural Network.

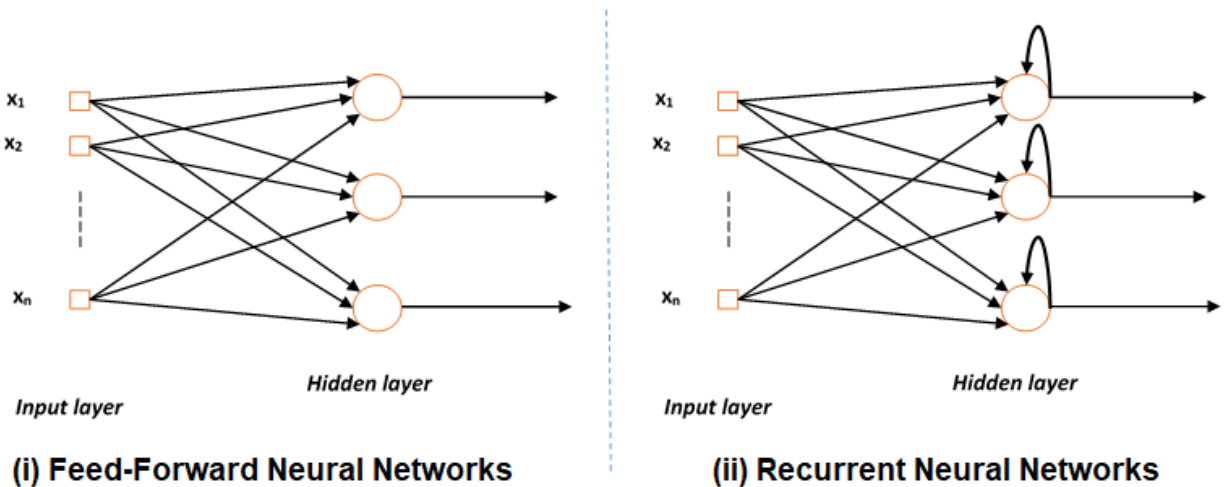


Figure 7-1 the difference in the information flow between an RNN and a Feed-Forward Neural Network

The problem with the Recurrent Neural Networks is the vanishing gradient problem where, depending on the activation functions used, information rapidly gets lost over time.

A.1.4. Long Short-Term Memory

Hochreiter & Schmidhuber (1997) [158] presented Long short-term memory to solve the problem of getting a Recurrent Neural Networks to remember things for a long time by building what known as long-short term memory networks (LSTMs).

A.1.5. Gated Recurrent Unit

Gated recurrent units (GRUs) are a slight variation on LSTMs. In most cases, GRUs function very similarly to LSTMs, with the biggest difference being that GRUs is slightly faster and easier to run.

A.1.6. Hopfield Network

John Hopfield (1982) introduced Hopfield Net in [159]. A Hopfield network (HN) is a network where every neuron is connected to every other neuron. Hopfield Net proved its limitations in its capacity.

A.1.7. Boltzmann Machine

Geoffrey Hinton and Terrence Sejnowski (1986) introduced the Boltzmann Machine in [160]. A Boltzmann Machine is a stochastic recurrent neural network. Boltzmann Machine is a stochastic, generative counterpart of Hopfield nets. Boltzmann Machine neural network capable of learning internal representations.

A.1.8. GoogLeNet

Christian Szegedy (2014) introduced GoogLeNet in [161] to reduce the computational burden of deep neural networks. GoogLeNet reduce the computational burden of deep neural nets and improve performance with the same computational cost.

A.1.9. Inception V3 and V2

Christian et al (2015) introduced Inception V2 in [162] and introduced Inception V3 in [163]. Inception V3 add more detail on the design choices to explains the original GoogLeNet architecture

A.1.10. Inception V4

Christian et al (2015) introduced Inception V4 in [164]. Inception V4 is rather similar to Inception V3 with the integration of ResNet module to the Inception module.

A.2. Comparative Study to Determine the Optimal Activation Function for a Certain Neural Networks

The activation function of a neuron calculates the output of that neuron based on the inputs and bias. The purpose of an activation function is to convert the input signal of a neuron to an output signal, which will be used as an input in the next layer.

$$Y = \sum(\text{weight} \times \text{input}) + \text{bias} \quad (7.1)$$

The activation function can be a Linear Activation Function or a Non-linear Activation Function. For a Linear Activation Function the output in the range of [-infinity, infinity] which is not effective with the complexity of a complex problem to be solved by a neural network. The Non-linear Activation Function is a frequently used activation function. The Non-linear Activation Function gives the neural network model more generalization capability and to adapt to a variety of data.

Sigmoid Activation function: Sigmoid is a widely used activation function. The general formula for Sigmoid Activation function:

$$f(x) = \frac{1}{1+e^{-x}} \quad (7.2)$$

For Sigmoid Activation function output values between 0 and 1, which make it efficient for models, used to predict the probability as an output. Sigmoid is non-linear, continuously differentiable that means, we can find the slope of the sigmoid curve at any two points, monotonic, and has a fixed output range.

The advantage of sigmoid Activation function that is a smooth gradient, preventing “jumps” in output values, and a precise prediction for input X above 2 or below -2, which result a predicted value close to 1 or 0 which enables a clear prediction. The disadvantage of sigmoid Activation function is that is vanishing gradient for very high or very low input values of X, which cause a vanishing gradient problem. This results a deficit of network to learn further or make it slow to reach an accurate prediction. Another problem is that the outputs of the sigmoid Activation function are not zero centered and computationally expensive.

Tanh or Hyperbolic Tangent activation function: Tanh function is similar to a sigmoid function, which can be considered a scaled version of the sigmoid function. The general formula for Tanh activation function:

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \quad (7.3)$$

The output of Tanh function has the range between -1 and 1 making it easier to optimize because is zero centered. Tanh function is easier to model inputs that have strongly negative, neutral, and strongly positive values. Tanh function has the vanishing gradient problem.

ReLU or Rectified Linear Unit activation function: ReLU is widely used activation function. The ReLU have a range between zero and infinity. The general formula for ReLU activation function:

$$f(x) = f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \quad (7.4)$$

The advantage of ReLU activation function is that it does not activate all the neurons at the same time, which make the network sparse making it efficient and easy for computation. It avoids and rectifies the vanishing gradient problem.

The disadvantage of ReLU function is that all negative values become zero, which decreases the ability of the model to fit or train from the data properly because the function doesn't map the negative values appropriately. ReLU function has a problem with the gradients moving towards zero. The gradient is zero for the negative values, which cause the inability to update the weights during backpropagation. This can create dead neurons, which will make it never activate on any data point again. The ReLU function used only within hidden layers.

Leaky ReLU activation function: Leaky ReLU is an improved version of the ReLU function to solve the dying ReLU problem. Leaky ReLU is an increase in the range of the ReLU function, which is [-infinity, infinity]. The general formula for ReLU activation function:

$$f(x) = f(x) = \begin{cases} 0.01x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \quad (7.5)$$

SOFTMAX activation function: The Softmax function is a type of sigmoid function but it is more able to handle more than two classification problems. Softmax function normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class. Softmax function will calculate the probabilities of each target class over all possible target classes. The general formula for Softmax activation function:

$$\sigma(z)_j = \frac{e_j^z}{\sum_{k=1}^k e_j^z} \text{ for } j = 1, \dots, K. \quad (7.6)$$

The ideal use of Softmax activation function is in the output layer for a classification problem in which the neural network seeks to define the class of each input neuron.

Conclusions:

The selection of activation function is critical in the design of neural network architecture. Most researchers use trial and error to define the suitable activation function for a considered problem. Experimental tests based on different activation functions for different problems are necessary to define the optimal activation function for certain neural networks. Some paper

proposes a method to automatically learn different combinations of base activation functions during the training phase to define the optimal activation function for certain neural network [165].

To define the optimal activation function for a specific problem it is necessary to evaluate the function based on a number of properties. Among them, the nonlinearity of function, the range, check if the activation function is continuously differentiable, check if it is monotonic, check if it is a smooth function with a monotonic derivative and check if the activation function approximates identity near the origin.

Depending upon the properties of the proposed problem it is possible to make a decision for the suitable activation function.

Sigmoid function generally works better with classification problems.

Due to the vanishing gradient problem, Sigmoid and Tanh functions are sometimes avoided.

ReLU function is widely used nowadays is a general activation function and is able to avoid the rectifies vanishing gradient problem.

To avoid the problem of dead neurons during training it is better to use leaky ReLU or Maxout function.

A.3. Comparative study of different types of cost function

The optimization algorithm of the neural network repeats a two-phase cycle propagation and weight update. The cost function is used to compare the obtained output with the desired output. The values of error obtained are used to calculate the gradient of the loss function. Then the optimization algorithm updates the weights in an attempt to minimize the loss function.

The cost function is used in machine learning to improve the performance of the model. The cost function is an evaluation tool to evaluate a model until we make certain of the ability of this model to estimate the relationship between an independent variable X and a dependent variable Y.

The cost function used to define the objective function (The objective function indicates how much each variable contributes to the value to be optimized in the problem usually it would be to maximize or to minimize some numerical value). The cost function is usually cross-entropy with L₁ and L₂ regularization for the classification problem [166].

$$E_w = \underbrace{-\sum_{x \in X} \sum_{k=1}^K [t_k^x \log(o_k^x) + (1 - t_k^x) \log(1 - o_k^x)]}_{\text{cross-entropy data loss}} + \underbrace{\lambda_1 \sum_{\omega \in W} |\omega|}_{L_1} + \underbrace{\lambda_2 \sum_{\omega \in W} \omega^2}_{L_2} \quad (7.7)$$

model complexity loss

where W is the weights, X is the training dataset, K is the number of classes and t_k^x is the training example x is of class k , o_k^x is the output of the classification algorithm which depends on the weights. λ_1 and λ_2 weights the regularization and is typically smaller than 0.1. L_1 represents Mean absolute error (MAE) and L_2 represent Mean Square Error (MSE).

The machine-learning model learns by minimizing the cost function. Gradient descent is an efficient optimization algorithm that attempts to find local or global minima of a function. The cost function falls under a number of categories such as classification cost functions, regression cost function and so on.

To determine the suitable cost function a comparative study of different types of functions is necessary. Depending on the type of the considered problem such as Binary classification, Multiclass classification or Regression it is possible to determine the best cost function.

A.3.1. Mean Square Error MSE

Mean Square Error (MSE) calculate the average of the squares of the errors. MSE is widely used regression cost function. Mean Square Error is the sum of squared distances between the observed values and the predicted values.

General formula for mean squared error:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \tilde{y}_i)^2 \quad (7.8)$$

Figure 7-2 represents the plot of the MSE function:

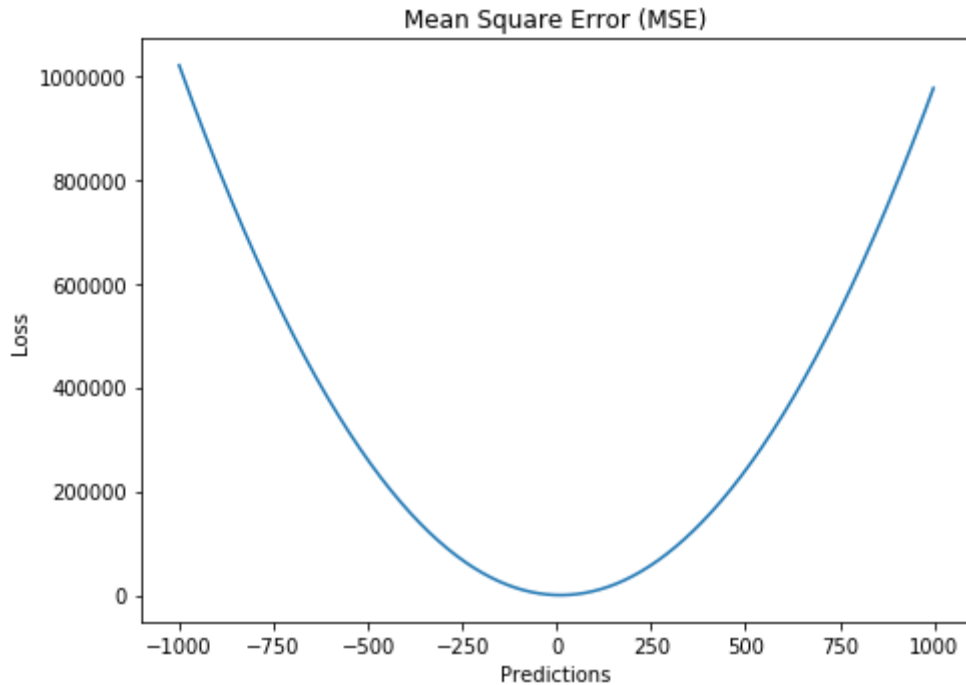


Figure 7-2 the plot of the cost function using MSE function

A.3.2. Mean Absolute Error MAE

Mean absolute error used as a cost function for regression models. Mean absolute error calculate the absolute average of the squares of the errors. MAE is the sum of the absolute of squared distances between the desired values and the predicted values.

General formula for Mean absolute error:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \tilde{y}_i|^2 \quad (7.9)$$

Figure 7-3 represents the plot of the MSE function:

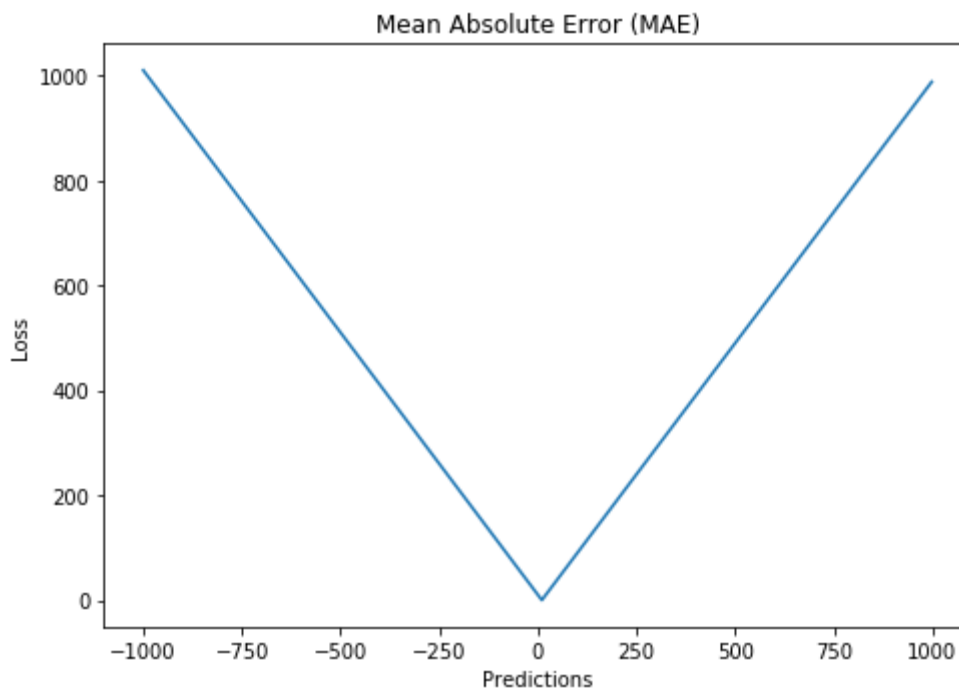


Figure 7-3 the plot of the cost function using MAE function

Experiment result obtained from the comparison between the Mean Square Error and Mean Absolute Error make as getting some conclusions. For a slight variance on the data in which the predictions are close to true values and the error has small variance among observations will make the model with Mean Square Error loss give more weight to outliers than a model with Mean Absolute Error loss.

For data with outliers, the Mean Square Error as a cost function will be adjusted to minimize the outliers, which will reduce its overall performance. Mean Absolute Error loss is useful if the training data corrupted with outliers, which consequently makes Mean Absolute Error more robust to outliers than Mean Square Error.

Using MAE loss have a problem when it used with a neural network represented in gradient, which is the same throughout causing an increase in the gradient making gradient large even for small loss values which aren't good for learning. This requires the use of dynamic learning rate, which decreases as we move closer to the minima.

It is preferable to use MSE as a cost function if the outliers represent important anomalies that should be detected. It is recommended to use MAE as a cost function if the outliers just represent corrupted data.

In the case of outliers, the difference between the incorrectly predicted target value and the original target value is large and the square of this value make it larger, which make L_2 error (MSE) much larger in the case of outliers, compared to L_1 (MAE). This makes us conclude that L_1 loss function is more robust and generally not affected by outliers. While the L_2 loss function is highly sensitive to outliers in the dataset.

A.3.3. Huber Loss

The loss function Huber Loss [167] is less sensitive to outliers in data than the squared error loss. This function is quadratic for small values of error and linear for large values. Depends on a hyperparameter, δ (delta) the Huber Loss function becomes quadratic for a small value of error. Huber Loss function approaches to MAE when δ tends towards zero and approaches to MSE when δ tends towards infinity ∞ .

The formula for Huber Loss [168]:

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \quad (7.10)$$

Figure 7-4 represents the plot of the Huber Loss function with three values of δ (delta) equal to 0.1, 1 and 10:

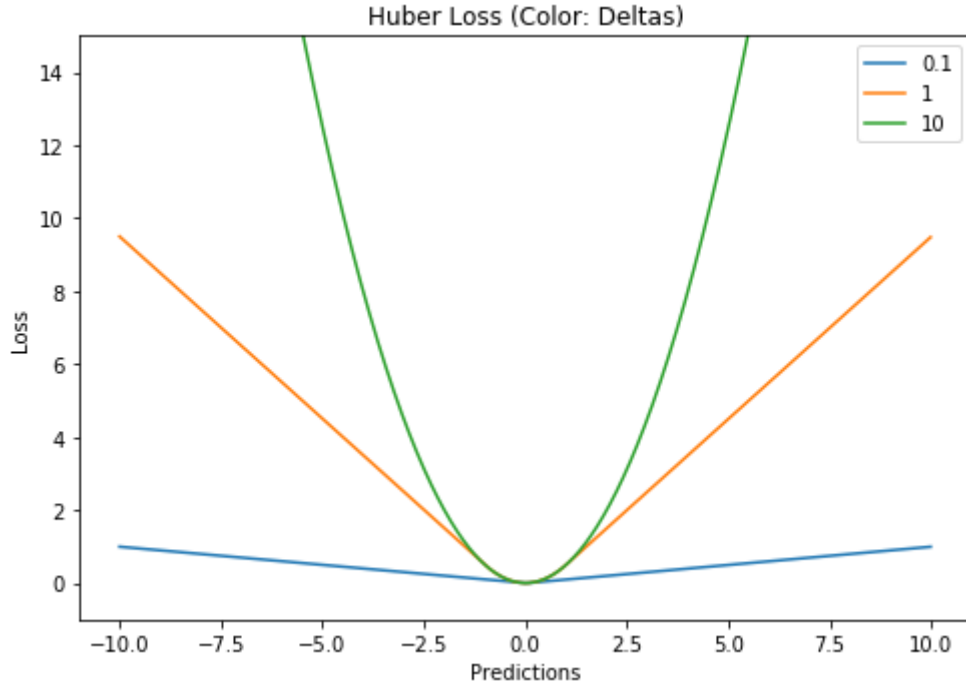


Figure 7-4 the plot of the cost function using Huber Loss function with three values of δ (delta) equal to 0.1, 1 and 10

The value of δ (delta) is selected based on the outliers in the data. Errors larger than δ minimized with MAE, while errors smaller than δ are minimized with MSE.

The problem of Mean absolute error for the training of neural networks is its constantly large gradient, which causes missing minima after training using gradient descent. For Mean Square Error, gradient decreases that makes the loss approaching its minima for more accuracy. While that Huber loss can be a good solution for these circumstances, as it curves around the minima, which decreases the gradient. Huber loss is more robust to outliers than Mean Square Error. Therefore, Huber loss has good features from the Mean Square Error and the good features from Mean absolute error. However, Huber loss imposes a training of hyperparameter delta.

A.3.4. Log-Cosh Loss

Log Hyperbolic Cosine (Log-Cosh) Loss function used as a cost function for regression models, which is smoother than MSE.

The general formula for Log-Cosh:

$$L(y, y^p) = \sum_{i=1}^n \log(\cosh(y_i^p - y_i)) \quad (7.11)$$

Figure 7-5 represents the plot of the Log Hyperbolic Cosine (Log-Cosh) Loss function:

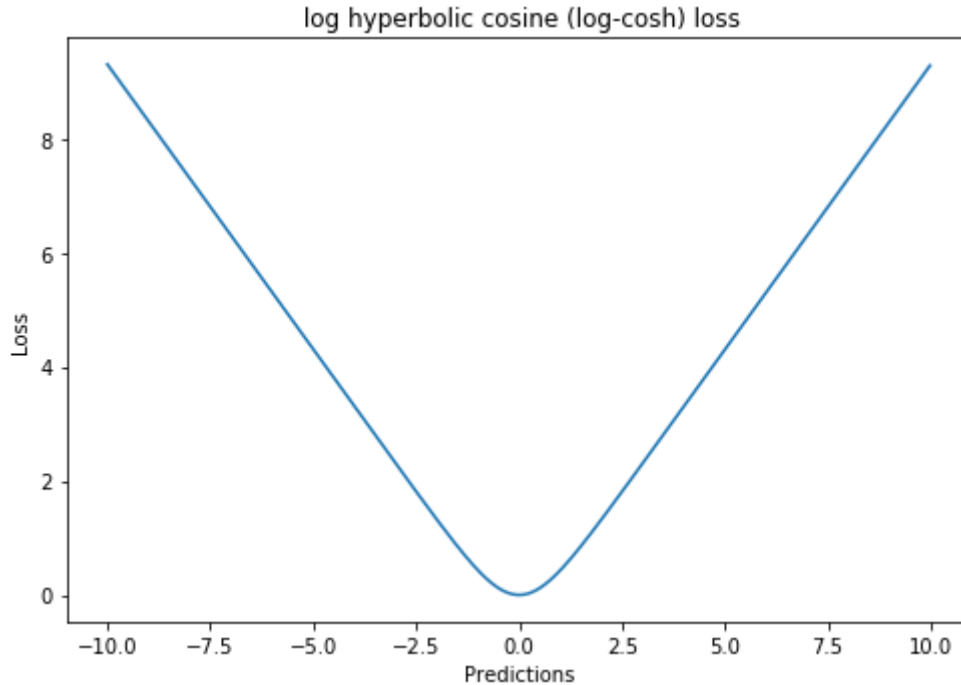


Figure 7-5 the plot of the Log Hyperbolic Cosine (Log-Cosh) Loss function

The Log-Cosh function is approximately equal to $\frac{x^2}{2}$ for a small value of x and to $|x| - \log 2$ for a large value of x , which make as conclude that Log-Cosh function is similar to MSE except that will not be affected by the occasional wildly incorrect prediction. It has all the advantages of Huber loss, and it is twice differentiable (second derivative) everywhere, unlike Huber loss. The second derivative (Hessian) is needed for many machine learning models to find the optimum. The Log-Cosh have the problem of gradient and hessian for very large bad predictions.

A.3.5. Quantile Loss

Quantile Loss function is recommended for the cases where we need to predict an interval instead of only one point. Using Quantile Loss function, we define a quantile value based on the adjustment needed to regulate the positive error or the negative error. Based on the defined quantile value (γ) the Quantile Loss function tries to give different penalties to overestimation and underestimation.

General formula for Quantile Loss function:

$$L_{\gamma}(y, y^p) = \sum_{i=y_i < y_i^p} (\gamma - 1) |y_i - y_i^p| + \sum_{i=y_i > y_i^p} \gamma |y_i - y_i^p| \quad (7.12)$$

where γ is quantile value, which is between 0 and 1.

Figure 7-6 represents the plot of the Quantile Loss function with three values of quantile γ (theta) equal to 0.25, 0.5 and 0.75:

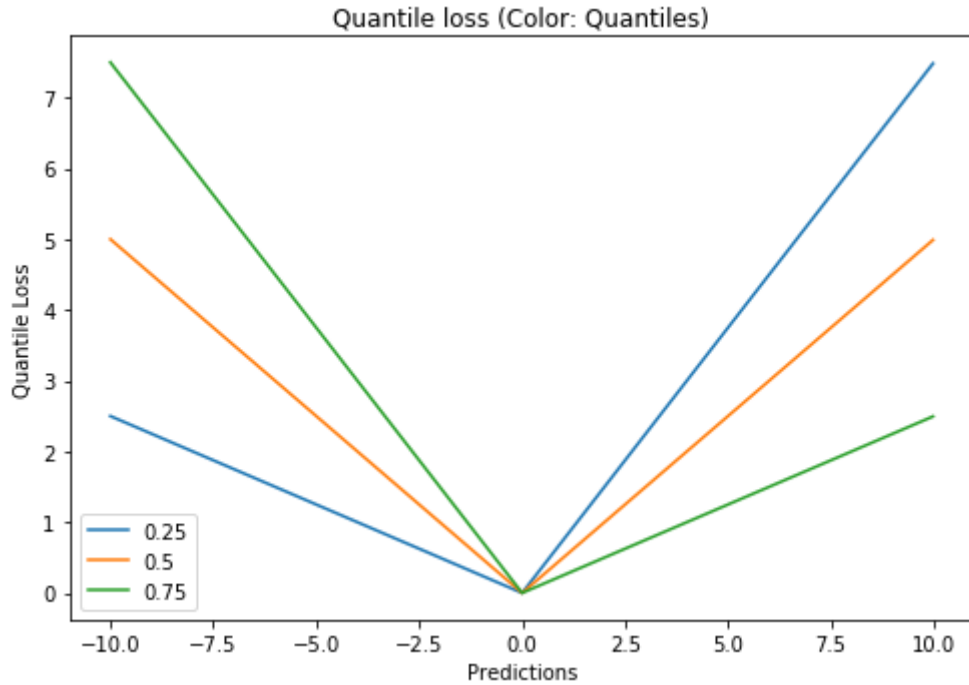


Figure 7-6 the plot of the cost function using Quantile Loss with values of quantile γ equal to 0.25, 0.5 and 0.75

Quantile Loss function is widely used for neural networks to calculate prediction intervals.

Through the study of previous cost functions, we observe a number of conclusions. The predictions using Mean absolute error loss is less affected by the impulsive noise while the predictions using Mean Square Error loss function is slightly biased due to the caused deviations. For a model with Huber loss, the predictions are sensitive to the value of hyper-parameter. The Quantile Loss function has a good estimation of the corresponding confidence levels.

A.4. Hyper-parameters associated to Network structure and Hyper-parameters associated to Training Algorithm

Neural networks depend on many Hyper-parameters, which are related to the structure of the neural network and to the learning algorithm of the network. Hyper-parameters related to the structure of neural network are the variables, which determines the network structure such as the number of hidden layers and neurons, network weight initialization and activation function. Hyper-parameters related to the learning algorithm of the network are the variables, which determine how the network is trained such as Learning Rate, Momentum, Number of epochs, Batch size and so on.

A.4.1. Hyper-parameters related to Network structure

The quantity of hidden layers/units represented Hyper-parameters related to the structure of the neural network which must be tuned to achieve the best accuracy of the network. The classical method is adding layers until the test error does not improve anymore to identify the quantity of hidden layers. The quantity of hidden units is determined by taking into consideration that a large quantity of hidden units can increase accuracy and a few hidden neurons can cause underfitting.

A.4.1.1. Dropout

Dropout is a regularization technique to avoid overfitting and give the more generalizing capability to the neural network. Commonly a dropout percentage is from 20 percent to 50 percent of hidden neurons. A dropout percentage of twenty percent can be a starting point. A low percentage of dropped neurons has minimal effect and a high percentage of dropped hidden neurons cause an under-learning of the network. It is recommended to use dropout on a large network to get good results.

A.4.1.2. Network weight initialization

Neural network weights are initialized with different methods. Weights are associated with the activation function, for this reason, the weights can be distributed based on the activation function used in each layer. Another method, which based on a uniform distribution (random values) of the weights for all neurons.

A.4.1.3. Activation function

Usually, activation functions represent a solution for nonlinear models. Activation functions allow deep learning models to learn nonlinear prediction boundaries. Activation functions are a Hyper-parameter related to the structure of the neural network, which needs to be tuned to achieve good accuracy and increase the generalization capability of the network.

A.4.2. Hyper-parameters related to Training Algorithm

Hyper-parameters related to the learning algorithm of the network, which represents the parameters used to train specific algorithms such as the learning rate, momentum, regularization coefficient and such like. The Hyper-parameters need to be tuned during training for any given neural net architecture.

A.4.2.1. Learning Rate

The learning rate affects the ability and the speed of the network to update his parameters. When the learning rate is selected with a low value, it slows down the learning of network but converges smoothly. For a large learning rate, the learning is done faster but it is possible that the convergence does not happen.

A.4.2.2. Momentum

Neural network momentum is the technique used to improve training speed and accuracy. Momentum helps to prevent oscillations. Generally, the value of momentum is selected from 0.5 to 0.9.

The momentum used to update the weights for the back-propagation algorithm. Formula (7.13) represents the rule of updating weights without using momentum and formula (7.14) represents the rule of updating weights using momentum.

$$\Delta w_{ij} = \left(\eta \times \frac{\partial E}{\partial w_{ij}} \right) \quad (7.13)$$

$$\Delta w_{ij} = \left(\eta \times \frac{\partial E}{\partial w_{ij}} \right) + (\gamma \times \Delta w_{ij}^{t-1}) \quad (7.14)$$

Δw_{ij} is the weight increment.

η is the learning rate.

$\frac{\partial E}{\partial w_{ij}}$ is the weight gradient.

γ momentum factor.

Δw_{ij}^{t-1} is the weight increment for the previous iteration.

A.4.2.3. Number of epochs

The Number of epochs is a hyper-parameter used to define the number of times the necessary for the learning algorithm to work through the entire training dataset to achieve the required accuracy for a neural network.

Usually, the number of epochs is taken with a large number often dozens or thousands or more. The number of epochs must be sufficient to minimize the error of the model.

A.4.2.4. Batch size

The batch size is a hyper-parameter used to select a number of samples from the dataset (training examples) that will be propagated through the network before updating the internal model parameters (utilized in one iteration).

The training dataset can be divided into one or more batches depending on the learning algorithm. There are three types of the batch size:

Batch gradient descent is the case when all the training dataset samples represent one batch in this case the learning algorithm is called batch gradient descent.

Stochastic gradient descent is the case when each sample of dataset represents a batch in this case the learning algorithm is called stochastic gradient descent. The network updates the internal model parameters after each sample.

Mini-batch gradient descent is the case when the batch size is more than one sample and less than the number of all the samples presented in the training dataset in this case the learning algorithm is called mini-batch gradient descent [169]. The most used batch size is 32, 64 and 128 samples.

A.4.3. Neural network parameters optimization

The hyper-parameter optimization or tuning is the selection of the optimal hyper-parameters suitable with a learning algorithm to achieve the best accuracy possible. The hyper-parameters mentioned above must be tuned so that the model can optimally solve the machine-learning problem. Hyper-parameter optimization helps to form a model based on precisely selected hyper-parameters that minimizes a predefined loss function for a specified training dataset [170].

A.4.4. Methods used to find out Hyper-parameters

There are different strategies and tools to handle the searching problem of Hyper-parameters. Methods used to find out Hyper-parameters try to determine the best configuration of hyper-parameters, which will lead to obtaining the results on the validation and test data.

The search process is too expensive because of the large space of possible hyper-parameter values.

There are four methods for searching in the hyper-parameter space for the optimum values:

A.4.4.1. Manual Search

The Manual search method called as Trial & Error or Grad Student Descent is the most widely used method. This method is completely manually. This method follows through all the steps of the learning process and iterates sequentially on the possible hyper-parameter values.

A.4.4.2. Grid Search

The grid search method is the simplest method for hyper-parameter optimization or tuning. Using this method, we build a model for all possible combination based on the possible hyper-parameter values then evaluating each model, and selecting the architecture, which has the best results.

A grid search algorithm depends on several performance metrics, usually measured using cross-validation on the training set [171] or evaluated on a validation set.

A.4.4.3. Random Search

The random search method is different from the Grid search method at one point, which is the random search provide a statistical distribution for each hyper-parameter. Random Search picks the point randomly from the configuration space.

Due to the inequality of the importance of hyper-parameters makes the use of random search more motivated than the Grid search method.

A.4.4.4. Bayesian Optimization

The Bayesian optimization method builds a surrogate model that tries to predict the considered metrics from different possible hyper-parameter values. This method to use the information gained from any given experiment to decide how to adjust the hyper-parameters for the next experiment.

Bayesian optimization proves that it is better compared to grid search and random search, due to the ability to information gained from previous experiments.

A.5. Hardware for Deep Learning

Deep Learning algorithms are widely used for categorizing objects in images and speech recognition, captioning images, understanding visual scenes, summarizing videos, translate the language, paint, even produce images, speech, sounds and so on. Deep learning algorithm such as ResNet [172], Xception, DenseNet [173] when used to solve for classification of huge datasets

into a large number of categories that requires doing a huge number of operations and a lot of power. Consider the difficulties it is better to focus on the side of efficiency hardware assigned to such Deep learning algorithms, which need more optimized microchips and hardware for Deep Learning solutions. Several hardware components assigned to deep learning algorithms must be with high efficiency among them the Graphic processors GPU, field-programmable logic devices FPGA, custom microchips, application-specific integrated circuits ASICs, or systems on a chip SoC, digital signal processors DSP.

A.5.1. GPUs

The graphics-processing unit (GPU) designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPU become increasingly needed due to deep learning. In some researches [174] [175] [176] it has been proven that the training deep learning neural networks using GPU faster than CPUs.

Currently, the GPUs considered as the norm in training Deep Learning systems. GPUs are capable to train large batches of images at once in just a few milliseconds.

A.5.2. FPGA

The field-programmable gate array FPGA designed to be configured based on the requirements of the customer or a designer.

FPGAs is used for hardware acceleration to accelerate certain parts of an algorithm. FPGA was used as AI accelerator, which is a class of microprocessor specially designed to speed up artificial intelligence applications, especially artificial neural networks [177], machine vision and machine learning.

A.5.3. ASIC

Application-specific integrated circuit ASIC [178] customized for a particular use, rather than intended for general-purpose use. ASIC architectures are qualified to support certain types of machine learning algorithms such as a deep CNN where the use of techniques to compress the model to improve hardware performance. ASIC outperform CPU, GPU, and FPGA in terms of energy efficiency and computation speed.

A.5.4. Conclusion

A considerable number of computer hardware platforms for machine learning algorithms was developed to improve energy efficiency and computation speed. The most important hardware platforms are GPU, FPGA and ASIC.

The graphics-processing unit GPU is characterized by its fast computation speed and compatibility with various algorithms.

The field-programmable gate array FPGA outperform GPU in terms of energy efficiency when computing Machine Learning algorithm at the expense of low speed.

Application-specific integrated circuit ASIC architectures are qualified to support certain types of machine learning algorithms to improve hardware performance.

ASIC outperform CPU, GPU, and FPGA in terms of energy efficiency and computation speed. Nevertheless, the selection of the most suitable computation hardware platform remains to depend on the specification of the application.

8. References

- [1] A. Itamar, D. C. Rose and T. P. Karnowski, "Deep machine learning-a new frontier in artificial intelligence research," *IEEE computational intelligence magazine*, vol. 5, no. 4, pp. 13-18, 2010.
- [2] Y. Zhao, J. Chen and H. Vincent Poor, "Efficient neural network architecture for topology identification in smart grid," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016.
- [3] C. Bishop, *Pattern recognition and machine learning*, Springer, 2006.
- [4] J. Bongard, "Biologically Inspired Computing," *INSPEC Accession Number: 10712660*, vol. 42, no. 4, 2009.
- [5] J. Hirel, P. Gaussier and M. Quoy, "Biologically inspired neural networks for spatio-temporal planning in robotic navigation tasks," in *IEEE International Conference on Robotics and Biomimetics*, 2011 .
- [6] H. Guanshan, "Neural Network Applications in Sensor Fusion for a Mobile Robot Motion," in *WASE International Conference on Information Engineering*, 2010.
- [7] R. Lovassy, L. T. Kóczy and L. Gál, "Fuzzy neural networks stability in terms of the number of hidden layers," in *IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2011.
- [8] K. Shibata and Y. Ikeda, "Effect of number of hidden neurons on learning in large-scale layered neural networks," in *ICCAS-SICE*, 2009 .
- [9] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, second ed., 2005.
- [10] Z. Haoyang, M. D. Edwards, G. Liu and D. K. Gifford, "Convolutional neural network architectures for predicting DNA–protein binding," *Bioinformatics*, vol. 32, no. 12, pp. i121-i127, 2016.
- [11] W. Ziyu, T. Schaul, M. Hessel, H. V. Hasselt, M. Lanctot and N. D. Freitas., "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.
- [12] C. Alfredo, A. Paszke and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

- [13] A. Ansari and A. Abu Bakar, "A Comparative Study of Three Artificial Intelligence Techniques: Genetic Algorithm, Neural Network, and Fuzzy Logic, on Scheduling Problem," in *4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, 2014 .
- [14] C. Guada, D. Gómez and J. Tinguaro Rodríguez, "Fuzzy Image Segmentation Based on the Hierarchical Divide and Link Clustering Algorithm," in *10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, 2015 .
- [15] Grady, Leo, Schwartz and L. Eric , "Isoperimetric Graph Partitioning for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, p. 469–475, 2006.
- [16] X. Chen, X. Liu, M. J. F. Gales and P. C. Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015 .
- [17] I. H. Witten, E. Frank and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (Morgan Kaufmann Series in Data Management Systems) 3rd Edition*.
- [18] P. B. Z. a. Q. V. L. Ramachandran, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.
- [19] R. Prajit, B. Zoph and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941* 7, 2017.
- [20] A. Ghosh, H. Kumar and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *hirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [21] S. O. Haykin, *Neural Networks and Learning Machines*, 3rd ed., 2008.
- [22] I. Yeo, S.-g. Gi, B.-g. Lee and M. Chu , "Stochastic implementation of the activation function for artificial neural networks," in *Biomedical Circuits and Systems Conference (BioCAS)*, 2016.
- [23] S. Raschka, "Python Machine Learning".
- [24] J. A. C. Mandic Danilo P, "On the choice of parameters of the cost function in nested modular RNN's.," *IEEE Transactions on Neural Networks* , vol. 11, no. 2, pp. 315-322, 2000.

- [25] J.-Y. Wu , "An Evolutionary Multi-layer Perceptron Neural Network for Solving Unconstrained Global Optimization Problems," in *IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016.
- [26] K. Chen, S. Yang and C. Batur , "Effect of multi-hidden-layer structure on performance of BP neural network: Probe," in *Eighth International Conference on Natural Computation (ICNC)*, 2012.
- [27] P. N. J. McCullagh, *Generalized linear models.*, 2nd edn, London: Chapman and Hall, 1989.
- [28] S. E. D, "Feedback stabilization using two-hidden-layer nets," *IEEE Transactions on neural networks*, vol. 3, no. 6, pp. 981-990, 1992.
- [29] M. S. H. W. Hornik Kurt, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359-366, 1989.
- [30] H. Kurt, "Some new results on neural network approximation," *Neural networks*, vol. 6, no. 8, pp. 1069-1072, 1993.
- [31] B. C. M, "Neural networks for pattern recognition," *Oxford university press*, p. 130, 1995.
- [32] R. B. D, "Pattern recognition and neural networks," *Cambridge university press*, pp. 173-180, 2007.
- [33] E. B. R. D. Geman Stuart, "Neural networks and the bias/variance dilemma.," *Neural computation*, vol. 4, no. 1, pp. 1-58, 1992.
- [34] A. Blum, *Neural networks in C++: an object-oriented framework for building connectionist systems*, John Wiley & Sons, Inc., 1992.
- [35] S. Kevin, *Applying neural networks: a practical guide*, Morgan Kaufmann, 1996.
- [36] G. L. Berry Michael, *Data mining techniques: For marketing, sales and marketing support*, NY: John Wiley & Sons, 1997.
- [37] H. G. Boger Zvi, "Knowledge extraction from artificial neural network models," in *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Orlando, 1997.
- [38] "Irvine Machine Learning Dataset and Benchmark," Center for Machine Learning and Intelligent Systems , [Online]. Available: <http://archive.ics.uci.edu/ml/datasets.html>. [Accessed May 2016].
- [39] I. T. Nabney, *Algorithms for Pattern Recognition*, London: Springer, 2004.

- [40] S. Bunjongjit, A. Ngaopitakkul, C. Pothisarn and C. Jettanasen, "Improvement to reduce training time of back-propagation neural networks for discrimination between external short circuit and internal winding fault," in *International Conference on Information Science, Electronics and Electrical Engineering*, 2014 .
- [41] A. Sasaki, K. Kinoshita and S. Kishida, "Effect of number of input layer units on performance of neural network systems for detection of abnormal areas from X-ray images of chest," in *IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS)*, 2011 .
- [42] C. A. Doukim, J. A. Dargham and A. Chekima , "Finding the number of hidden neurons for an MLP neural network using coarse to fine search technique," in *10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA)*, 2010.
- [43] M. Tomáš, A. Deoras, D. Povey, L. Burget and J. Černocký, "Strategies for training large scale neural network language models," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, 2011.
- [44] P. Adam, A. Chaurasia, S. Kim and C. Eugenio, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147* , 2016.
- [45] "NeuroShell 2," Ward Systems Group, Inc , [Online]. Available: <http://www.wardsystems.com/neuroshell2.asp>.
- [46] J. S. D. S. A. S. LeCun Yann, "Optimal brain damage," 1990.
- [47] D. G. S. G. J. W. Hassibi Babak, "Optimal brain surgeon and general network pruning," in *IEEE international conference on neural networks*, 1993.
- [48] J. P. J. T. W. D. Han Song, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015.
- [49] K. E. D, "A simple procedure for pruning back-propagation trained neural networks," *IEEE transactions on neural networks* , vol. 1, no. 2, pp. 239-242, 1990.
- [50] S. G. Vinod V. V, "Growing nonuniform feedforward networks for continuous mappings," *Neurocomputing*, vol. 10, no. 1, pp. 55-69, 1996.
- [51] G. M. M. MacLeod Christopher, "Incremental evolution in ANNs: Neural nets which grow," *Artificial Intelligence Review*, vol. 16, no. 3, pp. 201-224, 2001.
- [52] C. L. Fahlman Scott E, "The cascade-correlation learning architecture," *In Advances in neural information processing systems*, pp. 524-532, 1990.

- [53] G. S. I, "A connectionist learning algorithm with provable generalization and scaling bounds," *Neural Networks*, vol. 3, no. 2, pp. 191-201, 1990.
- [54] H. S. José, "Meiosis networks," *Advances in neural information processing systems*, pp. 533-541, 1990.
- [55] S. M. Bodenhausen Ulrich, "Automatically structured neural networks for handwritten character and word recognition," in *International Conference on Artificial Neural Networks*, London, 1993.
- [56] H. J. Henry, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.
- [57] R. M. Stanley Kenneth O, "Evolving neural networks through augmenting topologies," *volutionary computation*, vol. 10, no. 2, pp. 99-127, 2002.
- [58] D. B. D. J. G. Stanley Kenneth O., "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185-212, 2009.
- [59] J. L. a. K. O. S. Risi Sebastian, "Evolving the placement and density of neurons in the hyperneat substrate," in *12th annual conference on Genetic and evolutionary computation*, 2010.
- [60] G. O. N. N. R. R. Baker B, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.
- [61] W. R. J., "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 4-3, pp. 229-256, 1992.
- [62] Q. V. L. Zoph Barret, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
- [63] J. V. Saxena Shreyas, "Convolutional neural fabrics," in *Advances in Neural Information Processing Systems* , 2016.
- [64] B. S. L. L. E. L. Khaw John FC, "Optimal design of neural networks using the Taguchi method," *Neurocomputing* , vol. 7, no. 3, pp. 225-245, 1995.
- [65] D. P. R. H. Packianather MS, "Optimizing the parameters of multilayered feedforward neural networks through Taguchi design of experiments," *Quality and reliability engineering international*, vol. 16, no. 6, pp. 461-473, 2000.
- [66] T. J. Sukthomya W, "The optimisation of neural network parameters using Taguchi's design of experiments approach: an application in manufacturing process modelling," *Neural Computing & Applications*, vol. 14, no. 4, pp. 337-344, 2005.

- [67] G. Fred, "Artificial intelligence, heuristic frameworks and tabu search," *Managerial and Decision Economics*, vol. 11, no. 5, pp. 365-375, 1990.
- [68] Q. J. Ross, "Learning decision tree classifiers," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 71-72, 1996.
- [69] N. P. G, "Decision trees for predictive modeling," *SAS Institute Inc*, vol. 4, 1999.
- [70] S. I. Sulaiman, T. K. Abdul Rahman and I. Musirin , "Optimizing one-hidden layer neural network design using Evolutionary Programming," in *5th International Colloquium on Signal Processing & Its Applications CSPA*, 2009.
- [71] C.-Y. Lin, Y.-C. Chen, C.-Y. Wang, C.-Y. Huang and C.-T. Hsu , "Minimization of Number of Neurons in Voronoi Diagram-Based Artificial Neural Networks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 4, 2016.
- [72] E. Bashier and M. Tayeb, "Faults Detection in Power Systems Using Artificial Neural," *American Journal of Engineering Research (AJER)*, vol. 02, no. 06, pp. 69-75, 2013.
- [73] Z. Chiyuan, S. Bengio, M. Hardt, B. Recht and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.
- [74] H. Pan, D. Liang, J. Tang, N. Wang and W. Li, "Shape recognition and retrieval based on edit distance and dynamic programming," *Tsinghua Science and Technology*, vol. 14, no. 6, Dec 2009.
- [75] D. Stathakis, "How many hidden layers and nodes?," *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133-2147, 2009.
- [76] I. Ozan and E. Alpaydın, "Continuously constructive deep neural networks," *arXiv preprint arXiv:1804.02491*, 2018.
- [77] Y. Kim and Y. S. Kim., "Optimizing neural network to develop loitering detection scheme for intelligent video surveillance systems," *International Journal of Artificial Intelligence*, vol. 15, no. 2, pp. 30-39, 2017.
- [78] S. Javad, P. Moallem and H. Koofigar, "Training echo state neural network using harmony search algorithm," *Int. J. Artif. Intell*, vol. 15, no. 1, pp. 163-179, 2017.
- [79] G. Alejandro, A. Luviano-Juárez, I. Chairez, A. Poznyak and T. Poznyak, "Projectional dynamic neural network identifier for chaotic systems: Application to Chua's circuit," *Int. J. Artif. Intell* , vol. 6, no. 11, pp. 1-18, 2011.

- [80] D. Ian and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 59-70. Springer, Berlin, Heidelberg, 2005.
- [81] B. Pavel., "A survey of clustering data mining techniques," in *Grouping multidimensional data*, pp. 25-71. Springer, Berlin, Heidelberg, 2006.
- [82] T. W. Liao, "Clustering of time series data—a survey," *Pattern recognition* , vol. 38, no. 11, pp. 1857-1874, 2005.
- [83] Y. Tamura and S. Miyamoto, "Two-stage clustering using one-pass K-medoids and medoid-based agglomerative hierarchical algorithms," in *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, pp. 484-488. IEEE, 2014.
- [84] H. Geoffrey, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. V, N. P, K. B and S. T., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine* , vol. 29, 2012.
- [85] D. Gopikrishna, P. Wang, D. Rangaprakash and B. Wilamowski, "Fully connected cascade artificial neural network architecture for attention deficit hyperactivity disorder classification from functional magnetic resonance imaging data," *IEEE transactions on cybernetics* , vol. 45, no. 12, pp. 2668-2679, 2015.
- [86] G. Karypis, E.-H. Han and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer IEEE Journals & Magazines*, vol. 32, no. 8, Aug 1999.
- [87] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview, II," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 6, 2017.
- [88] H. A. Dalbough and N. M. Norwawi, "Improvement on Agglomerative Hierarchical Clustering Algorithm Based on Tree Data Structure with Bidirectional Approach," in *Intelligent Systems, Modelling and Simulation (ISMS), Third International Conference on Intelligent Systems Modelling and Simulation*, 2012.
- [89] C. C. Aggarwal and C. K. Reddy, "Data Clustering: Algorithms and Applications," *Chapman and Hall/CRC*, 2013.
- [90] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, Jul 1989.
- [91] P.-N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, 2006.

- [92] M. L. Tej and S. Holban, "Determining optimal neural network architecture using regression methods," in *2018 International Conference on Development and Application Systems (DAS)*, 2018.
- [93] "Connectionist Bench (Sonar, Mines vs. Rocks) Data Set," UC Irvine Machine Learning Repository. [Online].
- [94] "Arrhythmia Data Set," UC Irvine Machine Learning Repository. [Online].
- [95] "Statlog (Landsat Satellite) Data Set," UC Irvine Machine Learning Repository.. [Online].
- [96] "Glass Identification Data Set," Repository, UC Irvine Machine Learning, [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Glass+Identification>. [Accessed 2017].
- [97] F. Agostinelli, M. Hoffman, P. Sadowski and P. Baldi , "Learning Activation Functions to Improve Deep Neural Networks," 21 Dec 2014 .
- [98] R. d. T. R. H. a. J. D. D. Martin, "Optimal tuning of a networked linear controller using a multi-objective genetic algorithm and Its application to one complex electromechanical process," , " *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 10, pp. 3405-3414, 2009.
- [99] R.-C. D. E. M. P. M.-B. R. a. S. P. R.-E. Precup, "Adaptive GSA-Based optimal tuning of PI controlled servo systems with reduced process parametric sensitivity, robust stability and controller robustness," *IEEE Transactions on Cybernetics*, vol. 44, no. 11, pp. 1997-2009, 2014.
- [100] H. N. J. S. P. M. a. F. V. O. Castillo, "A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot," *Applied Soft Computing*, vol. 28, pp. 150-159, 2015.
- [101] F. D. a. H. M. H. S. B. Ghosn, "A parallel genetic algorithm for the open-shop scheduling problem using deterministic and random moves," *International Journal of Artificial Intelligence*, vol. 14, no. 1, pp. 130-144, 2016.
- [102] T. Kavzoglu, "Determining Optimum Structure for Artificial Neural Networks," in *in Proceedings of the 25 th Annual Technical Conference and Exhibition of the Remote Sensing Society*, 1999.
- [103] P. M. A. a. A. R. L. Tatnall, "Introduction Neural networks in remote sensing," *International Journal of Remote Sensing*, *International Journal of Remote Sensing*, vol. 18, no. 4, p. 699–709, 1997.

- [104] S. A. a. a. T. Y. W. A. S. Shirخورshidi, "A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data," *PLOS ONE*, vol. 10, no. 12, 2015.
- [105] S. A. R. a. V. B. B. I. A. Pestunov, "Hierarchical clustering algorithms for segmentation of multispectral images," *Optoelectronics, Instrumentation and Data Processing*, vol. 51, no. 4, p. 329–338, 2015.
- [106] L. S. a. L. M. E. BS, *Cluster Analysis* 4th ed, London: Arnold, 2001.
- [107] J. M. a. A. Jain, "A Self-Organizing Network for Hyperellipsoidal Clustering (HEC)," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, p. 16–29, 1996.
- [108] M. N. M. a. P. J. F. A. K. Jain, "Data Clustering: a Review," *ACM Computing Surveys* 31, vol. 3, 1999.
- [109] E. F. Codd, "Derivability, redundancy, and consistency of relations stored in large data banks," IBM Research Report, 1969.
- [110] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, 1970.
- [111] M. L. Tej and S. Holban, "Comparative Study of Clustering Distance Measures to Determine Neural Network Architectures," in *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2018.
- [112] D. U. o. Waikato, "Weka Waikato Environment for Knowledge Analysis," [Online]. Available: www.cs.waikato.ac.nz/~ml/weka/.
- [113] T. G. M. G. M. a. A. E. H. Smolinski, in *Computational Intelligence in Biomedicine and Bioinformatics: Current Trends and Applications*, Berlin, Springer, 2008.
- [114] N. Imran, R. Togneri and M. Bennamoun, "Linear regression for face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 11, pp. 2106-2112, 2010.
- [115] P. Francesc and Y. Vidal., "Wind turbine fault detection through principal component analysis and statistical hypothesis testing," *Energies*, vol. 9, no. 1, p. 3, 2016.
- [116] C. Patricia, S. G. West and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences*, Psychology Press, 2014.
- [117] Elliott and W. A. A. C. & Woodward, *Statistical analysis quick reference guidebook* Thousand Oaks, CA: SAGE Publications Ltd, 2007.

- [118] W. Wenbao, T. A. Morrison, J. A. Geller, R. S. Yoon and W. Macaulay, "Predicting short-term outcome of primary total hip arthroplasty: a prospective multivariate regression analysis of 12 independent factors," *The Journal of arthroplasty*, vol. 25, no. 6, pp. 858-864, 2010.
- [119] S. Chatterjee and A. S. Hadi., Regression analysis by example, John Wiley & Sons, 2015.
- [120] K. Alexander, I. Sutskever and G. E. Hinton, "Parallelizing neural networks during training.," *U.S. Patent 9,811,775, issued November 7, 2017.*
- [121] L. Xu and M.-Y. Chow, "Power distribution systems fault cause identification using logistic regression and artificial neural network," in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, 2005.
- [122] J.-N. Hwang, S.-R. Lay and M. Maechler, "Regression modeling in back-propagation and projection pursuit learning," vol. 5, no. 3, 1994.
- [123] M. L. Tej and S. Holban, "Determining optimal neural network architecture using regression methods," in *2018 International Conference on Development and Application Systems (DAS)*, 2018.
- [124] G. Umut and M. A. v. Gerven, "Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream," *Journal of Neuroscience*, vol. 35, no. 27 , pp. 10005-10014, 2015.
- [125] A. Muyeed, M. T. Imtiaz and R. Khan, "Movie recommendation system using clustering and pattern recognition network," in *In Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*, 2018.
- [126] R. S. J and P. Norvig, Artificial Intelligence: a Modern Approach, Pearson Education Limited, 2016.
- [127] A. Madhavi, B. R. Surampudi and A. Negi, "A survey of distance/similarity measures for categorical data," in *In Neural Networks (IJCNN), 2014 International Joint Conference*, 2014.
- [128] d. S. T. RL and L. E. Zárate, "Categorical data clustering: What similarity measure to recommend?," *Expert Systems with Applications* , vol. 42, no. 3, pp. 1247-1260, 2015.
- [129] E. S. Anitha and J. Akilandeswari, "Survey on clustering algorithm and similarity measure for categorical data," *ICTACT journal on soft computing* , vol. 4, no. 2, 2014.

- [130] C. C. e. Hau, *Handbook of Pattern Recognition and Computer Vision*, World Scientific, 2015.
- [131] B. Battista, S. R. Bulò, I. Pillai, M. Mura, E. Z. Mequanint, M. Pelillo and F. Roli, "Poisoning complete-linkage hierarchical clustering," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 42-52. Springer, Berlin, Heidelberg, 2014.
- [132] L. Peter, B. Zhang and S. Horvath, "Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R," *Bioinformatics* , vol. 24, no. 5, pp. 719-720, 2007.
- [133] d. O. a. A, M. P. Tcheou and s. Lovisolo, "Artificial Neural Networks For Dictionary Selection in Adaptive Greedy Decomposition Algorithms With Reduced Complexity," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8. IEEE, 2018.
- [134] W. C. S, D. L. Marino, K. Amarasinghe and M. Manic, "Generalization of Deep Learning for Cyber-Physical System Security: A Survey," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 745-751. IEEE, 2018.
- [135] J. Ma, X. Jiang and M. Gong, "Two-phase clustering algorithm with density exploring distance measure," *CAAI Transactions on Intelligence Technology* , vol. 3, no. 1, pp. 59-64, 2018.
- [136] H. Guang-Bin, P. Saratchandran and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation.," *IEEE Transactions on Neural Networks* , vol. 16, no. 1, pp. 57-67, 2005.
- [137] S. Karsoliya, "Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture," *International Journal of Engineering Trends and Technology* , vol. 3, no. 6, pp. 714-717, 2012.
- [138] A. M. Gethsiyal and T. Kathirvalavakumar, "Pruning algorithms of neural networks—a comparative study," *Central European Journal of Computer Science* , vol. 3, no. 3, pp. 105-115, 2013.
- [139] H. J, "Introduction to neural networks with Java," in *Heaton Research, Inc.*, 2008.
- [140] C. Seung-Seok, S.-H. Cha and C. C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43-48, 2010.

- [141] D. Nanjie, Z. Gao and K. Niu, "A Novel Data Dependent Similarity Measure Algorithm Based on Attribute Selection.," in *Big Data and Smart Computing (BigComp)*, 2018 IEEE International Conference on, pp. 603-606. IEEE, 2018 .
- [142] H. Roy and P. Beling, "Unsupervised Hierarchical Clustering of Build Orders in a Real-Time Strategy Game," *The Computer Games Journal* , vol. 7, no. 1, pp. 5-26, 2018.
- [143] W. Guan-De and S.-L. Lo, "Effects of data normalization and inherent-factor on decision of optimal coagulant dosage in water treatment by artificial neural network," *Expert Systems with Applications* , vol. 37, no. 7, pp. 4974-4983, 2010.
- [144] S. D. F, "A general regression neural network," *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568-576, 1991.
- [145] V. Sigillito, "Ionosphere Data Set," UCI Machine Learning Repository, [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Ionosphere>.
- [146] H. Mark, E. Frank, G. Holme, B. P. P. Reutemann and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter* , vol. 11, no. 1, pp. 10-18, 2009.
- [147] T. A. J, P. M, D. W. S, M. G. S and E. M. R, "On Predicting the Optimal Number of Hidden Nodes," in , " 2015 International Conference on Computational Science and Computational Intelligence (CSCI), 2015.
- [148] Vujicic, Tijana, M. Tripo, L. Jelena, A. Balota and S. Zoran, "Comparative Analysis of Methods for Determining Number of Hidden Neurons in Artificial Neural Network," in *Central European Conference on Information and Intelligent Systems*, p. 219. Faculty of Organization and Informatics Varazdin, 2016.
- [149] Berry, J. Michael and L. Gordon, *Data mining techniques: for marketing, sales, and customer support*, John Wiley & Sons, Inc., 1997.
- [150] L. Jiaqiang, H. C and J. Dwen, "Emission modeling of diesel engine fueled with biodiesel based on back propagation neural network," in *2010 3rd International Conference on Computer Science and Information Technology*, 2010.
- [151] E. L. Richards, "Generalization in neural networks," *Experiments in speech recognition*, 1992.
- [152] L. Chenxi, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

- [153] Z. Barret, V. Vasudevan, J. Shlens and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.
- [154] F. Dario, P. Dürr and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47-62, 2008.
- [155] L. Zhuang, M. Sun, T. Z. G. Huang and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.
- [156] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, 1958.
- [157] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179-211, 1990.
- [158] S. Hochreiter and J. Schmidhuber., "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1735-1780 1997.
- [159] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proceedings of the national academy of sciences*, 1982.
- [160] G. E. Hinton and T. J. Sejnowski., "Learning and relearning in Boltzmann machines," *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1, 1986.
- [161] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions". Computing Research Repository," *arXiv:1409.4842. Bibcode:2014arXiv1409.4842S*, 2014.
- [162] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.
- [163] S. Christian, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [164] S. Christian, S. Ioffe, V. Vanhoucke and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *In Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [165] A. R. Manessi Franco, "Learning combinations of activation functions," in *2018 24th International Conference on Pattern Recognition (ICPR). IEEE*, 2018.
- [166] G. E. H. Nowlan Steven J., "Simplifying neural networks by soft weight-sharing," *Neural computation*, vol. 4, no. 4, pp. 473-493, 1992.

- [167] H. P. J., "Robust Estimation of a Location Parameter," *Annals of Statistics*, vol. 53, no. 1, p. 73–101, 1964.
- [168] R. T. J. F. Trevor Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition, Springer, 2009, p. 349.
- [169] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.
- [170] M. Claesen and B. D. Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127*, 2015.
- [171] C.-W. Hsu, C.-C. Chang and C.-J. Lin, "A practical guide to support vector classification.," *Technical Report, National Taiwan University*, 2010.
- [172] H. Kaiming, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [173] H. Gao, Z. Liu, L. V. D. Maaten and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.
- [174] P. Thomas, H. Jin, J. Yang, Z. Lin and T. Huang, "Gpu asynchronous stochastic gradient descent to speed up neural network training," *arXiv preprint arXiv:1312.6186*, 2013.
- [175] A. Cotter, S. Nathan and K. Joseph, "A GPU-tailored approach for training kernelized SVMs," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM*, 2011.
- [176] S. Daniel, K. Kofler and S. Podlipnig, "Performance and scalability of GPU-based convolutional neural networks," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing. IEEE*, 2010.
- [177] M. Sparsh, "A survey of FPGA-based accelerators for convolutional neural networks," *Neural computing and applications*, pp. 1-31, 2018.
- [178] C. A. X. Ming and E. Culurciello, "Hardware accelerators for recurrent neural networks on FPGA," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.
- [179] T. Oohori, H. Naganuma and K. Watanabe, "A New Backpropagation Learning Algorithm for Layered Neural Networks with Nondifferentiable Units," *Neural Computation*, vol. 19, no. 5, 2007.

- [180] R. Berger, S. Dubuisson and C. Gonzales, "Fast multiple histogram computation using Kruskal's algorithm," in *19th IEEE International Conference on Image Processing*, 2012 .
- [181] S. John Peter and S. P. Victor, "Cluster validity with minimum spanning tree based clustering," *Journal of Theoretical and Applied Information Technology* , 2010.
- [182] D. Greco, R. Tagliaferri and A. Serra, "Impact of different metrics on multi-view clustering," in *International Joint Conference on Neural Networks (IJCNN)*, 2015 .
- [183] J. R. Rabuñal and J. Dorado, *Artificial Neural Networks in Real-Life Applications*, London, United Kindom: Idea Group Inc, 2006.