T.C.

ALINBAŞ ÜNİVERSİTESİ

Electrical and Computer Engineering

DIAGNOSIS ON LUNG CANCER USING ARTIFICIAL NEURAL NETWORK

Mohammed Khalaf Abdullah

Master Thesis

Asst. Prof. Dr. Sefer Kurnaz

İstanbul 2019

DIAGNOSIS ON LUNG CANCER USING ARTIFICIAL NEURAL NETWORK

by

Mohammed Khalaf Abdullah

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

DEDICATION

To the one who made me learn how to succeed, to the one who I miss him in the middle of difficult times, to the one who the life decided to take him before I have enough from his caring, to my father may he rest in peace. And to the one who my word racing to describe her caring, to the one who tried her best to teach me, support me, and did the best to make me to what I am now, to the who I swim in the sea of tenderness that she has to cure my wounds, To my mom who I ask Allah to give the longest healthy life. To my brother and sisters. To everyone who at least taught me at least a letter …

I give you this thesis in a form of appreciation and I may ask Allah to accept this work.

ACKNOWLEDGEMENTS

I wish to express my acknowledgements to my supervisor, Asst. Prof. Dr. Sefer Kurnaz who was abundantly helpful and offered invaluable support with his sincerity and belief in me.

ABSTRACT

DIAGNOSIS ON LUNG CANCER USING ARTIFICIAL NEURAL NETWORK

Abdullah, Mohammed Khalaf

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst Prof. Dr. Sefer kurnaz

Date: March 2019

Pages: 51

In today's world, there is a dire need for the appropriate use of technology to diagnose and treat patients by analyzing medical data. Moreover, data mining is one of the predominant researches in computer science field. It is a process of extracting valuable information from a huge amount of dataset. There are many approaches are adopted such as classification, image, regression analysis and association rule mining. In this paper we're representing a concentrated study on classification technique to analyze the performance of data mining where several classification algorithms were used based on classification accuracy, error rate, average error rate, and standard Deviation. The main goal of this study is to make an open view on determining the best procedure for data mining in medical analysis based on performance analysis on lung cancer dataset. There are some common classification techniques such as linear regression, Decision Trees, Gradient Boosting Machine (GBM), Support vector machines (SVM), and Custom ensemble. These approaches are used for the performance of medical classification. In this study, because of the dynamics that ANN has, we're going to use ANN for the classification of breast cancer. Which is probably will make better accuracy, less error, better sensitivity and specificity.

Keywords: Artificial neural network, Backpropagation, Genetic Algorithm, Mean square error, optimization, classification.

ÖZET

YAPAY SİNİR AĞINI KULLANARAK AKCİĞER KANSERİNİN TEŞHİSİ

Abdullah, Mohammed Khalaf

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Danışman: Asst Prof. Dr. Sefer kurnaz

Tarih: March 2019

Sayfalar: 51

Günümüz dünyasında, tıbbi verileri analiz ederek hastaları teşhis etmek ve tedavi etmek için teknolojinin uygun kullanımına ciddi bir ihtiyaç vardır. Ayrıca, veri madenciliği bilgisayar bilimleri alanındaki en önemli araştırmalardan biridir. Büyük miktarda veri setinden değerli bilgileri çıkarma işlemidir. Sınıflandırma, imaj, regresyon analizi ve dernek kuralı madenciliği gibi birçok yaklaşım benimsenmiştir. Bu yazıda, sınıflandırma doğruluğu, hata oranı, ortalama hata oranı ve standart Sapma'ya dayanan çeşitli sınıflandırma algoritmalarının kullanıldığı veri madenciliğinin performansını analiz etmek için sınıflandırma tekniği üzerine yoğunlaşmış bir çalışmayı temsil ediyoruz. Bu çalışmanın temel amacı, akciğer kanseri veri setindeki performans analizine dayanarak tıbbi analizde veri madenciliği için en iyi prosedürün belirlenmesi konusunda açık bir görüş vermektir. Doğrusal regresyon, Karar Ağaçları, Degrade Yükseltme Makinesi (GBM), Destek vektör makineleri (SVM) ve Özel topluluk gibi bazı yaygın sınıflandırma teknikleri vardır. Bu yaklaşımlar tıbbi sınıflandırma performansı için kullanılır. Bu çalışmada, ANN'nin sahip olduğu dinamikler nedeniyle, meme kanserinin sınıflandırılmasında ANN kullanacağız. Muhtemelen daha iyi doğruluk, daha az hata, daha iyi hassasiyet ve özgüllük sağlayacaktır.

Anahtar kelimeleri: Yapay sinir ağı, Backpropagation, genetik algoritma, Ortalama kare hatası, optimazisyon, sınıflandırma.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION

The neural system is arranged into hidden layers, input and output of the Artificial Neural Networks (ANNs). The neurons are joined together by a series of synaptic weights. An ANN is a powerful tool for identifying patterns, predictions and regressions in a variety of problems. The ANN continually changes its synaptic values during the learning process until sufficient acquired knowledge (until a certain number of iterations have been achieved or the error value of the target has been achieved). After completion of the learning or training stage, the ability of the ANN to generalize the problem with samples other than those employed during the training stage must be assessed. Finally, during training and testing, it is expected that the ANN will be able to accurately classify the patterns of a particular problem. In recent years several classic ANN algorithms have been suggested and developed. However, many of them can stay trapped in nondesirable solutions; that is, they will be far from the optimum or the best solution. Moreover, most of these algorithms cannot explore multimodal and noncontinuous surfaces.

Therefore, other kinds of techniques, such as bioinspired algorithms (BIAs), are necessary for training an ANN. As BIAs are strong optimization tools and can resolve very complicated optimization problems, they are well accepted by the Artificial Intelligence Community. For a given problem, BIAs can browse large multimodal and continuous search areas and find the optimum value for the best solution. BIAs is based on nature's behavior described as swarm intelligence. The term [1] defines this concept as being owned by unintelligent agents with limited individual capacities, but intelligent collective behavior.

There are several studies which use evolutionary and bio - inspired algorithms as a basic way of training ANN [2]. Metaheuristic methods for training neural networks are based on local search, population methods, and others such as cooperative coevolutionary models [3]. An excellent work in which the authors show a comprehensive literature review of evolving ANN algorithms [2]. The majority of the research reports however focus on the development of synaptic weight, parameters [4] or the evolution of the numbers of the neurons for hidden layers, but the designer determines the number of hidden layers previously. Moreover, the researchers do not involve the evolution of transfer functions, which are an important element of an ANN that determines the output of each neuron. In [5], for instance, the authors suggested a combining of the Ant Colony

Optimization (ACO) methodology to identify ANN and PSO for weight adjustment. Additional studies such as the [6] modify the Simulated Annealing (SA) PSO for the acquisition of a set of synaptic weight and threshold ANNs. In [7], the authors use Evolutionary Programming to obtain the architecture and weight to solve problems of classification and forecasting. Another example is [8] where Genetic Programming is used to obtain graphs that represent different topologies. In [9], the Differential Evolution (DE) algorithm was applied to design an ANN to solve a weather forecasting problem. In [10], the authors use a PSO algorithm to modify the synaptic weights to model the relationship between daily rains and runoffs in Malaya. In [11], the authors compare the back-propagation method versus basic PSO to adjust only the synaptic weights of an ANN for solving classification problems. In [12], the set of weights are evolved using the Differential Evolution and basic PSO. The three principal elements of an ANN were simultaneously developed in other works such as [13]: architecture, transfers and synaptic weights. In [14] the authors solved the same problem with a differential Evolution (DE) algorithm and suggested a new model with a PSO (NMPSO) algorithm for the authors. The author has also used [15] to develop the design of an ANN with two different fitness functions by using an Artificial Bee Colony (ABC) algorithm.

Therefore in this research work, we proposed a technique that uses PSO for ANN training to improve the training and testing performance of existing ANN on the diabetes dataset.

The chapters in this study focus are:

1. Chapter 1: The first chapter focuses on this thesis, which explains the idea and the purpose of this thesis. That makes a complete review of how this work can be done.
2. Chapter 2: The second chapter is the literature review of references to this thesis. Their results and the algorithms used to classify the dataset by other researchers are thoroughly reviewed.
3. Chapter 3: The third chapter focuses on the methodology for this thesis where the collection of data and the general look of ANN, GA and BP are explained. In addition, review a clear understanding of the characteristics of these algorithms and the ways they can be developed. It also gives a full idea of how GA and BP train an ANN and how these algorithms are being developed during their training. The advantages and disadvantages for ANN training are also presented by both BP and GA.

4. Chapter 4: This chapter provides a clear picture of the results of this study, which shows both GA and BP performance in ANN training and compares the results.
5. Chapter 5: The last chapter is the results and the recommendation that gives an overview of the results and summarizes the results from our experiments.

1.1 PURPOSE AND THESIS IMPORTANCE

This research aims to present a new optimization algorithm called GA, which will improve ANN's training performance. The fitness functions take into consideration the classification error, mean square error, validation error, architectural reduction or a combination. This research also explores the behaviours, using different parameter values, of three bioinspired algorithms. The best parameter values for these algorithms are determined to achieve optimum results during the experimentation phase. In addition, a number of statistically valid experiments are generated for each selected classification problem using the best configuration. Furthermore, the results of the proposed methodology are discussed with regard to the connection number, the neuron numbers and the transfer functions selected for each ANN.

1.2 METAHEURISTIC ALGORITHMS

A metaheuristic method in computer science and mathematical optimisation is designed to identify, create or select heuristic (partially searched algorithms) that can provide a good and adequate solution to the optimisation problem, especially if the information is incomplete, or imperfect, or if the calculator is limited. For sampling metaheuristics, a number of solutions are excessive. Metaheuristics can make few assumptions about the problem of optimization, and can therefore be used in a number of problems.

Metaheuristics cannot ensure that some types of problems are found to offer a global optimal solution in comparison with optimizing algorithms and iterative methods. Many metaheuristics are used to optimize the stochastic to make the solution dependent upon the set of random variables. When combined, metaheuristics often find solutions with less computational effort by looking for a wide array of feasible solutions than optimization Algorithms. Therefore, they are useful

solutions to optimization problems. There have been published a number of books and papers on this topic.

The majority of the metaheuristic literature is experimental, describing empirical findings from experiments with computer algorithms. However, formal theoretical results, often regarding convergence and the possibility of finding the global optimum are also available. Many metaheuristic methods have been published with novelty and practical efficiency. Although there was high quality research in the field, many publications were of bad quality; vagueness, inadequate design, poor experiments and ignorance of previous literature are some of the shortcomings.

1.2.1 Genetic Algorithm (GA)

In the late 1960sç, the concept of GA was first introduced to the evolutionary theory by Holland and its colleagues. In nature, unsuitable and weak species face extinction by natural selection compared to their environment. The strongest species have more opportunities for reproductive transmission of their genes into future generations; this process takes time and is gradual. The species with the right gene combination will be more dominant over the long term. At times, random changes in the genes may take place during this slow process of evolution. These unintended variations offer additional benefits through diversification to the natural selection process. New species evolve from old species, unsuccessful modifications, and natural combinations automatically eliminate in this never-ending challenge of survival. In GA terminology, an individual or chromosome solution vector is referred to. These chromosomes consist of discrete units known as genes. One or more chromosome elements are controlled in each gene. The initial Holland GA implementation had binary numbers as genes. The latest deployments provided more representation and encoding of genes. A chromosome is usually the only solution in the space of the solution. GA works with the population, a set of chromosomes. The population is usually randomly initialized. As the algorithm runs, people with a higher fitness are progressively found ; each candidate has a number of properties (genotype) that can be mutated and therefore better solutions are found.

2. LITERETURE REVIEW

The Artificial Neural Networks (ANNs) are the neural system organized into hidden layers, input and output. The neurons are joined by a set of synaptic weights. An ANN is a powerful tool used to identify models, predictions and regressions in a variety of problems. The ANN continually changes its synaptic values during the learning process until sufficient acquired knowledge (until a certain number of iterations have been achieved or the error value of the target has been achieved). After the training stage, the capacity of ANN to generalize the problem must be evaluated with samples different from those used during the training stage. The problem should be generalized.

Several classic algorithms to train an ANN have been proposed and developed in the last years. However, many of them can stay trapped in nondesirable solutions; that is, they will be far from the optimum or the best solution. Moreover, most of these algorithms cannot explore multimodal and noncontinuous surfaces. Therefore, other kinds of techniques, such as bioinspired algorithms (BIAs), are necessary for training an ANN.

As BIAs are powerful optimizing instruments and can solve very complex optimizing issues, they have a positive support by artificial intelligence. BIAs can explore large multimodal and non - constant search areas for the best solution near the optimal value for a specific problem. BIAs are based on the behavior of nature described as the swarm. In [1] this concept is defined as owned by smart and limited personal agents with smart collective behavior.

There are several works that use evolutionary and bioinspired algorithms to train ANN as another fundamental form of learning [2]. Metaheuristic methods for training neural networks are based on local search, population methods, and others such as cooperative coevolutionary models [3].

The authors show a comprehensive literature review of evolutionary algorithms used for the development of ANN [2]. Most of the research reports are, however, focused exclusively on the development of synaptic weights, parameters [4] or evolution of the numbers of neurons in hidden layers. The designer determines the number of hidden layers. Moreover, the researches do not involve the evolution of transfer functions, which are an important element of an ANN that determines the output of each neuron.

For example, in [6] the authors suggested a method combining ANN and Particle Swarm Optimization (PSO) for a specific architecture (connections) to adjust the weight of the synaptical. Other research, such as [6], has carried out a modified PSO mixture with Simulated Annealing to obtain a set of ANN and ANN weights. [7] In order to find out the architecture and weights, authors use evolutionary programming to solve classification problems and forecasting problems. Another example is [8] where Genetic Programming is used to obtain graphs that represent different topologies. In [9], the Differential Evolution (DE) algorithm was applied to design an ANN to solve a weather forecasting problem. In [10], the authors have been adjusting the synaptic weight by the PSO algorithm to model the daily relations in Malaysia between precipitation and rain. In [11], the authors compare the back-propagation method versus basic PSO to adjust only the synaptic weights of an ANN for solving classification problems. In [12], the set of weights are evolved using the Differential Evolution and basic PSO.

The three main elements of the ANN have evolved simultaneously in other works such as[13]: architecture, transfer functions and synaptic weights. In [14], the authors solved the same problem using the Differential Evolution (DE) algorithm. They proposed a new PSO model. Another example is [15] in which authors develop an ANN design with two different fitness functions using an Artificial Bee Colony (ABC).

This research has significant contributions in comparison with these last three works. First and foremost, there are eight fitness functions to address three common problems arising from the ANN design: accuracy, overriding and reducing ANN. In this respect, fitness functions take classification error into account, mean square error, validation error, architecture reduction and a combination of problems that arise while designing the ANN. This research further examines the behaviour, using different parameter values, of the three bioinspired algorithms. The values of the best parameter for these algorithms will be determined to achieve the best results during the experimentation phase. Furthermore, the best configuration for each selected classification problem is used to create a set of statistically valid tests. In addition, the results obtained by the proposed methodology are presented and discussed regarding the connection number, the number of the neuron and the transfer functions selected in the ANN. Another contribution of this research is related to a new metric that allows comparing efficiently the results provided by an ANN generated with the proposed methodology. This metric takes into account the recognition rate

obtained during training and testing stages where testing accuracy is more weighted in comparison to training accuracy. Finally, the results achieved by the three bioinspired algorithms are compared against those achieved with two classic learning algorithms. The selection of these three biological - inspiring algorithms is based on the metaphor for a basic PSO technique, because NMPSO is a relatively new algorithm (in 2009). It is important to compare its performance to other algorithms that are motivated by the same.

In general, the problem to be resolved can be defined as a set of input patterns, and a set of the desired patterns, and an ANN is found represented by the function to be determined by so that the maximum number of neurons is minimized and defined. It is important to note that three fields (architecture, synaptic weight and transfer functions) form part of the search space.

This research offers a comprehensive study of how the application of bioinspired algorithms can be automated by an ANN, in particular by applying Basic Particle Swarm Optimization (PSO), Second-Generation PSO (SGPSO) and New PSO Model (NMPSO). In order to design the ANNs that are most exact in a particular problem, the proposed methodology evolves at the same time the architecture, the synaptic weights and the type of transfer functions. Moreover, a comparison of the Particle Swarm algorithm performance versus classic learning methods (back-propagation and Levenberg-Marquardt) is presented. In addition, in this research is presented a new way to select the maximum number of neurons (MNN). The accuracy of the proposed methodology is tested solving some real and synthetic pattern recognition problems. In this paper, we show the results obtained with ten classification problems of different complexities.

3. METHODOLOGY

3.1 DATA RESOURCE

Hong and Young used these information to show that even in unprotected environments, the power of the optimally discriminating aircraft. The KNN method applied in the resulting plane with 77 percent precision. These results are however very partial (see Aeberhard for the second reference above, or e-mail to stefan' @'coral.cs.jcu.edu.au). The outcome of Aeberhard et al. is:

RDA : 62.5%, KNN 53.1%, Opt. Disc. Plane 59.4%

Three types of lung cancers were described in the data. The authors don't tell you which variables or where the data was originally used.

Notes:

- For the fifth attribute, the original data 4 values were -1. Having changed these values? (Unknown). (Unknown). Available in German.

- For the 39 attributes, the original data 1 value was 4. Has this value been modified? (Unknown). (Unknown).

Information of Lung Cancer Data Database are described below:

- Title: Lung Cancer Data
- Sources:
    - Original owners: Data was published in :
        - Hong, Z.Q. and Yang, J.Y. "Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane", Pattern Recognition, Vol. 24, No. 4, pp. 317-324, 1991.
    - Donor of database: Stefan Aeberhard, stefan@coral.cs.jcu.edu.au
    - Date received: May, 1992
- Number of Instances: 32
- Number of Attributes: 57 (1 class attribute, 56 predictive)
- For Each Attribute: (all numeric-valued)

- ❖ attribute 1 is the class label
- ❖ All predictive attributes are nominal, taking on integer values 0-3

- Missing Attribute Values: Attributes 5 and 39 (*)
- Class Distribution: 3 classes,
  - 9 observations
  - 13 "
  - 10 "

3.2 DATA PREPROCESSING

The dataset had some missing values which was fix using the tool "IBM SPSS statics 24" and using the algorithm Estimation-Maximization (EM) and then processed using z-score for artificial neural network training.

Expectation – maximization algorithm

Statistics have the effect of finding maximum or maximal post-map (MAP) estimates in statistical models, which depend on unnoticed parameters in the model, with expectation–maximization (EM) algorithms being an iterative method. The EM iteration alternates between a (E) step that produces a function for expectation of the evaluated log-like feature using the current estimate for parameters and a (M) step that calculates parameters maximizing the expected log-like feature in step E. These parameter estimates are then used to determine the latent variables distribution in the next step E.

Z-Score

In the middle of a dataset, z is the default number. Technically, however, it is a measured amount of standard differences in the population or above. A z-point is a standard distribution curve parameter, as well. The range of Z scores varies from-3 standard deviations to-3 standard deviations (to go far left with the normal distribution curve) (to the right of the normal distribution)-the mean μ is known, and if the z scoring is to be carried out, the standard population deviation¨is known.

Z scores are a way to use "normal" populations to compare test results. There are thousands of results and units to be achieved for the test results or surveys. These results, however, often seem insignificant. For example, it might be a good piece of information to find someone weighs 150 pounds but to search for a comprehensive basis of information if some weights are compared to the average, especially if some weights are recorded in the kilogram. You can see a z score where the average person weighs.

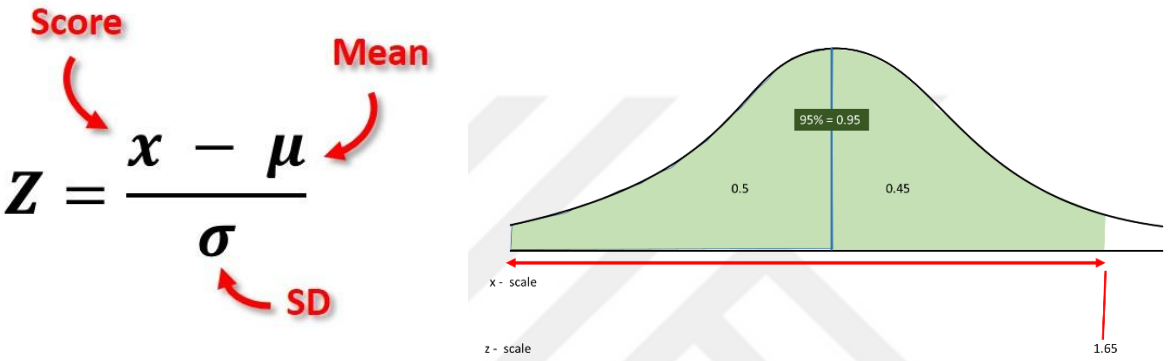

Figure 3.1: Z-score Formula

Z-Score formula

If you have several examples, if you want to describe the default deviation (standard error), you will use the format z - score. If you have several samples.

$$Data_{new} = \frac{Data_{old} - mean}{standard\ division} \tag{3.1}$$

3.3 ARTIFICIAL NEURAL NETWORKS

The neuron structure in the human brain inspires the Artificial Neural Network. The brain learns from experiences that go beyond the range of computers and thus adapts. This modeling also allows solutions to be less technologically developed in order to reduce human involvement.

The implementation of neural networks in computing gives us a key computational advance. Computers, like complex math and face recognition, do well. However, simple patterns are hard for computers. Computers cannot analyze, generalize and transform past patterns into future actions. The advanced neural network study allows people to understand the mechanism of thoughts, for example.

The study focusses on the' brain storage data as patterns method [17]. For individual faces to analyze and recognize, some of these patterns are very difficult. This process saves information as patterns, analyzes patterns and fixes a new computer field for problems. Training and creation of these networks is involved in the neural network to solve specific problems. We can perform different techniques such as learning, behaving, forgetting and reacting, etc. through our neural network.

"All aspects of this processor are known: The human brain is still mysterious to operate accurately. In particular, the most important element in the human brain is a certain type of cell which does not appear to regenerate as opposed to the whole body. Since the only portion of the body not slowly replaced by this type of cell, we are supposed to be able to remember, think and put past experiences into practice in all our actions. These cells are all known as neurons". The neural network as the human brain's neural network offers power through its complex components, its control mechanisms and its subsystems. Genetics programming and learning are also involved. In electro-chemical ways, neurons transmit the information between them. Depending on the classification process used, these neurons are classified into different categories. Current systems remain incompatible with the human brain. "The most basic elements of this complex and powerful organization can only replace these artificial neural networks. Neural computing has never succeeded the developer who is trying to solve problems with the human brain. Neural computing was never substituted to human brain by the development company which attempts to solve problems. It's a new way of solving problems and machinery.

Let's see the overview of human neurons. The basic element of the neural network is neuron. The biological neurons are input and subsequently subject to various non - light operations and then produce final output. Four nerve cells are typical: dendrites, somas, axons and synapses. It accepts inputs as the task of dendrites. The input is processed by Soma. Axon-transforms the processed inputs and synapses— contacts neurons. The biological neurons in structure are not too straightforward but complex.

Biology improves the understanding of neurons. By understanding the biological brain, network designers can enhance their systems further. Neural artificial networks (ANN) are computer tools which have been widely accepted in many different disciplines to model complicated problems in

the real world. ANNs could be defined as structures consisting of strongly linked adaptive elements called neurons that provide massive computations for the representation of knowledge.

"The principal features of the ANN are the ability to handle inaccurate and inadequate information, the ability to manage inaccurate information, robustness, failure and failure tolerances, fault and failure tolerance."

Artificial models possess following characteristics:

1. Nonlinearity makes it more suitable for data
2. Precise prediction for uncertain data and measurement errors due to noise insensitivity.
3. High parallelism means tolerance of hardware failure and quick processing.
4. Learn and adapt to the changing environment, allowing the system to update its internal architecture.
5. The model can be applied to unlearned data by generalization.

The main goal of ANN - based computing is to develop mathematical algorithms that enable artificial neural networks to study the processing of information and to gather human brain knowledge. Patterns based on ANN may provide virtually accurate solutions to specifically formulated problems and processes that only experimental and field observations understand. “In a number of applications from modeling, classification, design recognition and multi - variable data analysis, microbiological ANNs have been applied. The digital image processing interests are derived from two main applications: improvement for human - the reading of photo information; and processing of image information for storing, transferring and representing autonome sensor machine; and defining a two - dynamic function as the image, f(x, y).

“Digital image processing is the processing of digital images via digital computers. A numerous elements with a specific position and value comprises a digital picture. They are called picture elements, elements, pixels and skins. An extensive and diversified application for digital image processing”.

3.3.1 General Properties of Artificial Neural Networks

1. **Nonlinear I/O mapping**

   The analysis of high-dimensional data has an increasing importance due to the growth of sensor resolution and computer memory capacity. Typical examples are images, speech signals and multi-sensor data. In the analysis of these signals they are represented in their entirety or part by part in a sample space. As an example, a single data point may represent a 16x16 (sub) image. A collection of these images constitutes a cloud of points in a 256-dimensional space. In most multi-sensor data sets there is a large dependency between the sensors or between the sensor elements. This is certainly true for nearby image pixels. But also more fundamentally, it is not to be expected that any physical experiment contains hundreds of degrees of freedom that are of significant importance. Consequently, multi-dimensional data sets may be represented by lower dimensional descriptions. There are various reasons why such representations may be of interest. They may reveal the structure of the data or the problem, they may be used for relating individual data points to each other (e.g. finding the most similar one in a database) or they may be used for retrieving missing data values.

2. **Generalization ability**

   Their capacity to generalize is one of the major advantages of neural networks. This means that a trained network can classify data from the same class that it never saw. Developers in real-life applications usually only have a small portion of all possible neural net generation patterns. The data set should be divided into three parts to achieve the best generalization:

   The training is for the training of neural networks. During training, this data set error is minimized.

   The validation set is used to determine the performance of neural networks in untrained patterns. A test set to finally monitor a neural net's overall performance. In the minimum validation error, learning should be stopped. The net generalizes best at this stage. If learning is not stopped, overtrainings occur, and net performance decreases over all data,

even if the error is still smaller in the training data. The network must finally be checked for the third dataset, the test set, after the completion of the learning phase. Each n training cycle, ANNS performs one validation cycle.

3. **Fault-tolerance (graceful degradation)**

   Fault tolerance is the feature that permits the system, when certain components of it fail (or one or more inside failures), to continue to operate properly. The decline in operating quality, as opposed to an unnatural system, is proportional to the gravity to which the failure is causing a total failure if it does not. Fault tolerance is especially sought in high availability or life-critical systems. When parts of a system break up, features are known as graceful degradation.

   A defect-tolerant design allows a system to operate at a lower level than to fail completely if some of the system fails. When a system fails. The term is used most often to describe computer systems designed to continue to operate more or less fully in the event of a partial failure, with reduced output or with an increased response time. In other words, the entire system is not stopped due to hardware or software problems. A motor car is designed to continue to drive in the presence of damage caused by fatigue, corrosion, manufacturing defects or impacts, when a tire has been punctured or when structural integrity is maintained. Another example is an automotive vehicle.

   The tolerance of failures can be achieved by anticipating special conditions and building a system to deal with them within the scope of an individual system, and, in general, by trying to stabilize itself so as to converge the system in a faultless state. Nevertheless some kind of reproduction is better used when the consequences of an insufficient system are catastrophic or the costs of making it reliable enough are very high. In each case, the system must be able to reverse it in safe mode, if a system failure is so catastrophic. The recovery is like a reversal, but if people are present, it can be a human action.

4. **Biological analogy**

   Analogy is a cognitive process where information or significance is translated to a different objective or linguistic expression from one particular topic–analogy or source. A less narrowly defined inference or argument from one specific to another is in comparison to

the analogy of deduction, induction and abduction where at least one premise or conclusion is general. The word analogy could also refer to the relation of the source and the goal itself, often but not necessarily similar, as in the concept of biology. The atom model of Rutherford analogized the atom with the solar system. Analogy plays a major role in problem solving, including decision-making, discussion, perception, generality, remembrance, creativeness, invention, prevision, emotions, explanation, conceptualisation and communication. The analogy was argued as "the cognitive nucleus." In particular, the analogical language includes, although not metonymous, exemplification, comparatives, metaphors, like, alligories and parable. It is not methonymous, but it does not contain the identification of places, objects or persons. As if it were, phrases like, etc. and so on, and the same term also depends on the analogous understanding of the message by the receiver. Analogy is important not only in ordinary language and common sense in the fields of science, philosophy, law and humanity. Analogy is closely linked to concepts of association, comparative approach, correspondence, mathematical and morphological equality, homomorphism and iconicity. The concept of conceptual metaphor in the cognitive linguistics can be the same as the concept of analogy. Analogy also provides a basis for all comparative arguments and experiments whose findings are conveyed to objects not examined.

3.3.2 Structure of ANN

The neural network consists of three groups, layers and phases. The phases input, covered and output.

- The activity or input unit is the raw data that the network receives.
- The cached phase is based on the data entered and the weights of the connection between the input and the cached units.
- The phase of the output depends on the activity of the cloaks and weights between the cloaked and output units.

On the basis of the layer activity, different network types exist. A simple network type is called where the hidden units can build their own input representation. Whenever each hidden unit is active, the masses of the hidden and input units can be adjusted to allow for a hide unit to choose

what it is. Other architectures such as single and multilayer are also available. In a single layer each layer is connected to the other. Overall, the network of the single layer includes only entries and outputs. The inputs are supplied by a number of weights to outputs. In several layers all units have inputs, hidden and outputs in different layers.

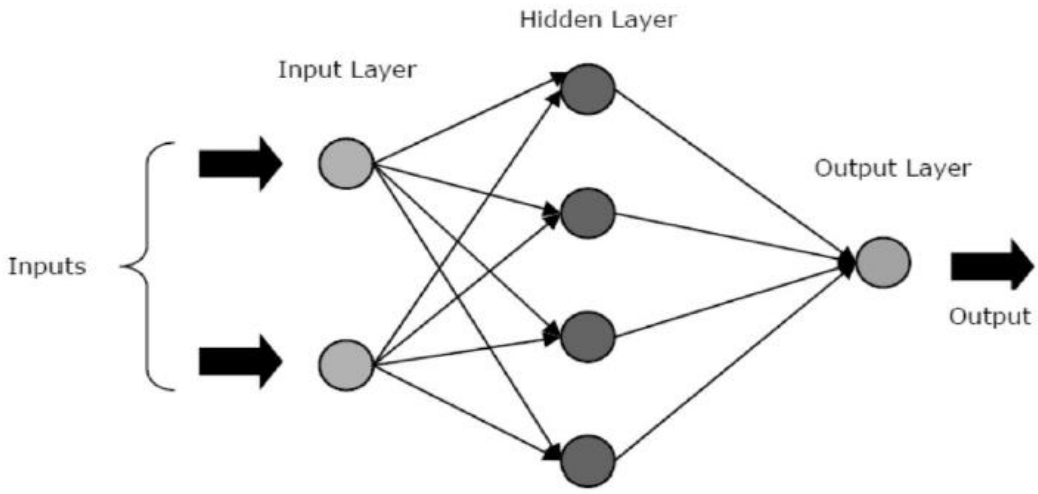

Figure 3.2: General neural network architecture [2]

3.3.3 Elements of Artificial Neural Networks

3.3.3.1 Inputs

The data, signal, feature, or outside environment measurements are received from this layer. These inputs are generally normalized within the limits of the activation functions. The values of samples or patterns. This standardization gives greater numerical accuracy for the network's mathematical operations.

3.3.3.2 Weights

The links between neurons are assigned to the number "weights" or "parameters" in artificial neural networks. These weights change as new data are fed into the neural net. So "learns the neural net.

The weights in an artificial neural network are an estimation of the combined multiple processes in biological neurons. Myelination is important, but not important. Myelination. Weights can be

positive or negative in artificial neural networks. Weight: weight is comparable to an increased combination of dendrites between neurons, numbers of synapses between dendrites, density of postsynaptic terminal neurotransmitter receptors, as well as increased formation and fusion of neurotransmitters of the vesicles and of the pre-synaptic terminals.

Positive weights: The positive weights of the synapses releasing excitation neurotransmitters (i.e. glutamate) are analogous to the pre - synaptic terminals. The receiving cell will increase its likelihood of activation.

Negative weights: Negative weights are similar to those of the neurotransmitter synapse (i.e. GABA). The receiving cell is less likely to fire a potential action.

Myelination: Myelination increases the distance between the action potential and the axon. If the axon is not myelinated, the membrane voltage potential decreases far closer to the cell body. A garden hose is analogous. If the axon does not myelinize, the garden pants are leaking, and a lower water pressure (which cares for waves of water pressure, the potential action) leads to the end of the pants.

3.3.3.3 Additive Function

An additive function is an arithmetic f(n) function in a positive integer that the product function is the sum of its functions when a and b are copressed. F(a) + f(b) holds a completely additive function even when they arenot co-prime to all positive integral parts a or b if f(b) holds f(a)+ f(b). In this sense also, analogy with fully multiplicative functions is used with complete additive. F(1)=0 if f is a full additive feature. Each fully additive function is an additive, but not the other way around.

3.3.3.4 Activation Function

The threshold or transfer feature is also known as activation functions. The activating functions have been used to transform neuron activation levels into output signals. Numerous activation functions are available in the neural network. Identity function, step function, part linear function and sigmoid function are various function types.

a. Identity activation function:

The activation function of identity is also referred to as "liner activation." The Network Activation function can be shown easily to fit a line regression model of the form if the ID is used in the network $Y_i = B_0 + B_1 + \cdots + B_k B_k$ where $x_1, x_2, \ldots\ldots\ldots, x_k$ are the k network inputs, Y_i is the i-th network output $B_1, B_2, \ldots\ldots\ldots, B_k$ are the coefficients in the regression equation. Consequently, a neural network with identity activation used in all its sensors is uncommon to be found.

b. Sigmoid activation function:

Nonlinearity in the model is used in the artificial neural network sigmoid functions. The result of a linear combination of its input signals is calculated by a network neuroelement using a signmoid function. The sigmoid function makes an interface between the product and itself easier and more popular in the Neural Network.

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{3.2}$$

Sigmoid function results are generally used in learning algorithms. The Sigmoid graph is shaped as' S'. This function is defined as an expanding function which is commonly used for neural artificial network development. Sigmoid is a function that strictly increases, and shows a balance between linear and nonlinear functions.

One - polar – is the sigmoid function.

c. Step function:

This is a unipolar threshold, known as.

$$\varphi(v) = \begin{cases} 1 \; if \; v \geq 0 \\ 0 \; if \; v > 0 \end{cases} \tag{3.3}$$

The neuron K output with a threshold is

$$y(k) = \begin{cases} 1 \; if \; v_k \geq 0 \\ 0 \; if \; v_k > 0 \end{cases} \tag{3.4}$$

v_k *is the induced local field of the neuron*

$$v_k = \sum_{j=1}^{m} W_{kj} X_j + b_k \tag{3.5}$$

When the neuron output is 1 when the local neuron field induced is not - negative, the neuron output is 0.

d. Piece Wise Linear Function

It can be defined as a unipolar function

$$\varphi(v) = \begin{cases} 1, & v \geq +1/2 \\ v, +\frac{1}{2} > & v > -1/2 \\ 0, & v \leq -1/2 \end{cases} \tag{3.6}$$

When it is expected that the amplification factor is within the linear area

1. The specific circumstances of linear functions are
    - If the linear operating area is maintained without saturation, a linear combiner is produced.
    - If the linear region's amplification factor is infinitely large, it reduces to a threshold feature.

e. Learning Rules in neural network

There are many different types of study rules in the neural network, usually divided into 3 categories.

- Supervised Learning
- Unsupervised Learning

a. Supervised Learning

Training sets are available for supervised learning. This type of rule includes a set of examples with proper network behavior. The inputs are given as a training in controlled learning and the

expected results are achieved. Parameters in this type of study are adjusted step by step by error signal; parameters are adjusted step by step by step by error signal.

A number of examples (trainings set) together with correct network conduct are provided for the learning rule.

$$\{x1, d1\}, \{x2, d2\}, \dots\dots\dots, \{xn, dn\} \tag{3.7}$$

The network input is xn in this case and dn is the required destination input. The output is generated by input. In order to make network outputs more exact, the Study rule is employed to change network biases and weights.

We commit ourselves with supervised learning to give the system the desired answer (d) when the entry is implemented. The distance between the real response and the desired response is used to correct the network parameter externally. For example, the error can be used to change weighing in the study of input patterns or circumstances where the answer to the error is recognised. For the learning mode, the training set, several input and output patterns are needed.

b. Unsupervised learning

In unexpected learning, self-organized learning is also known. Objective output is not available in uncontrolled learning. In that case, only network input changes weights and biases. For pattern reorganization, unattended study grouping is used. The answer required is not known in unattended learning, therefore explicit error information cannot be utilized for improving network behaviour. Information of this type is not available to correct the wrong answers so that learning has to be done based on observations of marginalized or unknown responses to the data.

The algorithms in unchecked learning use redundant row data, which have no etiquette for class membership or associations. In order to identify its parameters in this way, the network needs to detect any existing patterns, properties, regulations, etc. Unattended study means learning without the teacher because it is not necessary for the teacher to participate, but the teacher must set objectives. Feedback on neural networks is important as well. Feedback is called progressive learning, which for uncontrolled learning is very important.

3.3.3.5 Outputs

This layer contains also neurons that generate and display the final neural network outputs at the previous layers. Thus the artificial neural networks ' principal architectures can be divided, taking into consideration the connection and the structure of the neuronal disposition. The following are possible:

(i) Single-layer feedforward network,

(ii) Multilayer feedforward networks,

(iii) Recurrent networks and

(iv) Mesh networks.

3.3.4 Classification of Artificial Neural Networks

3.3.4.1 Artificial Neural Networks According to Constructions

A. Single Layer Feed Forward Network

A neural network with a source node projecting the neural output level, but not one single feed or an acyclic network. The single layer refers to the calculation node output layer in one network layer Figure 3.2.

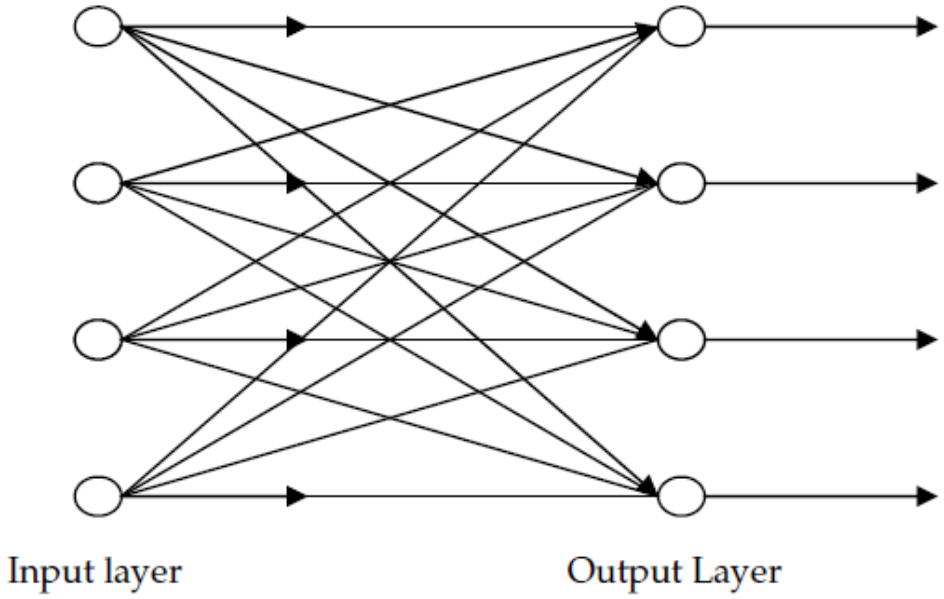

Figure 3.3: Single Layer Feed Forward Network [2]

B. Multilayer Feed Forward Network

The system consists of at least one cache layer known as clad neurons or clad unit hubs. The ability of clad neurons is to work between external information and system output and to get separate insights on a higher demand. In the system input layer, the source hubs provide the information flag for the neurons in the second layer. The third layer of inputs are the second layer of output signals, and so on. The general reaction of the system to the actuation design provided by sources in the main layer of the info is neuron output motions in the system yield level.

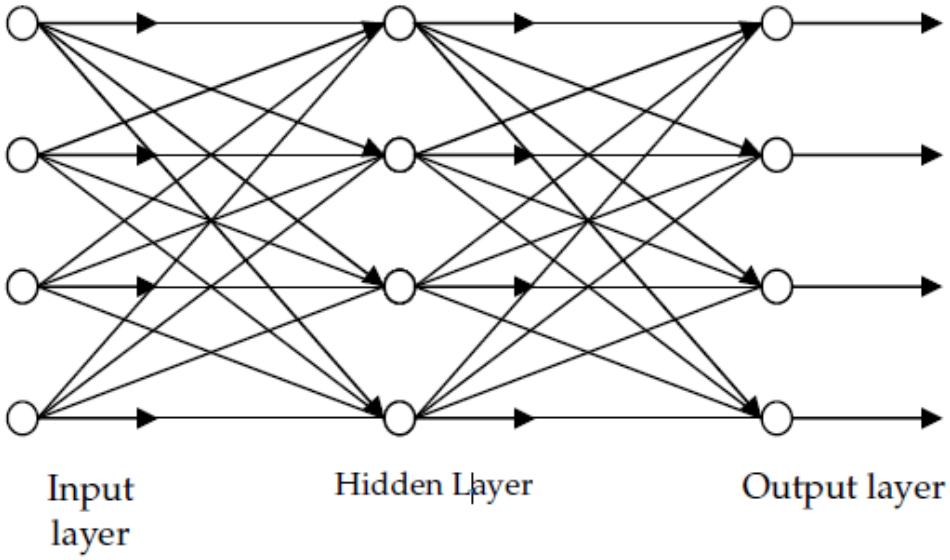

Figure 3.4: Multilayer Feed Forward Network [2]

Short feeding network characterization:

1. In general, activation is provided via ' hidden layers ' from input to output but many other architectures exist.
2. Static input - output mappings are implemented by mathematics.
3. Most popular algorithm for backpropagation supervised training:
4. Has proved useful in many practical applications as approximations of nonlinear functions and as a classification model.

C. **Recurrent Network**

The repetitive system in the figure is known as a forward neural system of something like the input circle, which includes at least one shrouded layer. 2.3.2. The first time. Critics could be your own critique, i.e. if your own information returns to neuronal yield. The use of unit deferring components, which leads to a unique conduct, has sometimes been criticised, since the neural system has non-direct units.

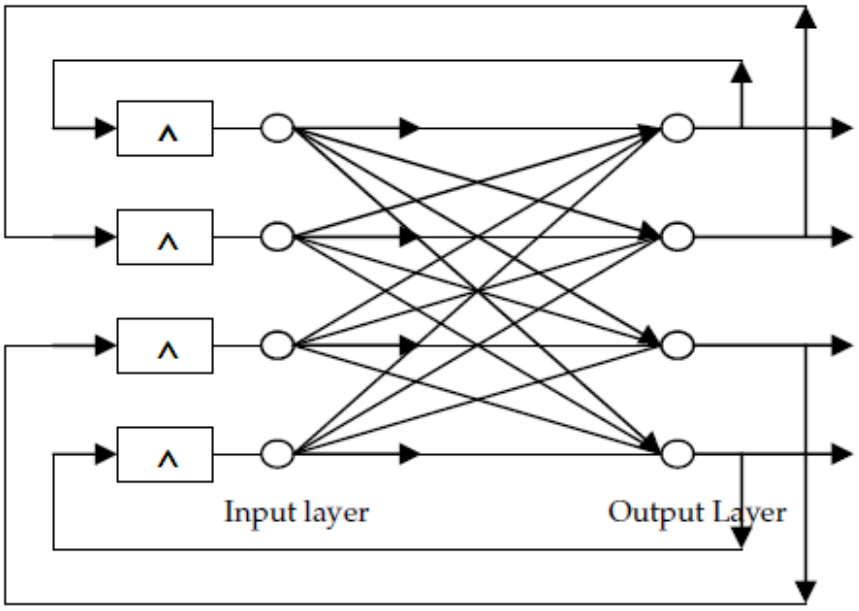

Figure 3.5: Recurrent Network [2]

Other network types are available ; Delta - bar - delta, hop field, quantifying vectors, counterpropagation, probabilist, hamming, memory boltzman, spatium - temporal patent, adaptive resonance, auto - order, recirculation etc.

The cyclic path of synaptic connections is a recurring neural network.. Basic characteristics:

1. Recurring are all biological neural networks

2. They implement dynamic systems mathematically

3. Various types of algorithms, no clear winner, are known

4. In general, theoretical and practical problems have prevented practical applications up to now.

3.3.4.2 Artificial Neural Networks According to Learning Algorithms

If a network is structured for a particular application, it will be ready for formation. At start and beginning of training, initial weights are selected randomly. Two approaches are available; controlled and unattended.

A. Supervised Training

The inputs and outputs are provided for supervised training. The data is processed by the network and the results are compared with the results. The system spreads errors to adjust network weights. This process takes place time and again when weight is constantly changed. The same data set is processed many times during networking training, as the weight of connections is always improved. The data set for the training is referred to as the "training set."

Sometimes a network can never learn. This can take place as there are no specific information in the input data to produce the required output. Networks also do not converge when there are not enough data to enable complete learning. In order to retain part of the data as a test, there should ideally be sufficient data. Multiple nodes in many layered networks can store data. To monitor the network, the supervised training must retain data for the system to test after training of a system, to determine whether the system simply saves its data.

Then, if a network cannot simply solve the problem, the designer must examine inputs and outputs, number of layers, numeric elements per layer, layer connections, transfer and training capabilities and also initial weights. The training rules govern another part of the creativity of the designer. Many laws (algorithms) are used in the training session to make the required weight adjustment feedback. The most common method is back-propagation. It is a conscious analysis and not only a technique to prevent overtraining of the network. General statistical trends in the data are the initial setup of the artificial neural network. It then' learns' about other aspects of the data which are generally falsified.

The weights can be frozen upon request, if finally the system is properly trained and no further training is necessary. This network can then be converted on certain systems into hardware. During production, other systems do not lock in, but continue learning.

B. Unsupervised or Adaptive Training

Uncontrolled (learning) is the other kind of training. The network has inputs but not desired outputs in this kind of way. The system must then determine its own functions in order to

group the input data. Often this is termed autonomy or adaptation. These networks do not adjust their weight using external influences. Their performance is instead monitored internally. The networks seek the regularity or trends of the input signal and adjust it to the network functions. Although the network still needs to be able to provide information on how to organize itself without knowing whether or not this is right. These data are incorporated into the network's topology and study rules. The cooperation between the processing element clusters may be emphasized by an unchecked learning algorithm. The clusters would work together under such a scheme. If any external input activated a node within the cluster, it could increase the whole activity of the cluster. Furthermore, it can inhibit the whole cluster if external input to the cluster nodes has been decreased. Competition between processing elements could also provide a learning basis. Competitive cluster training could enhance the response of particular groups to special incentives. In that sense, these groups would be linked and a special appropriate response would be provided. The weight of the winning processing element is normally only updated when the learning contest is effective. Unattended learning is not well understood at the present time and a lot of research is in progress.

C. LEARNING LAWS (ALGORITHMS)

There are many common uses for learning laws. The majority are a variation of ' Hebb's rule ' which is the best - known and oldest.

Hebb's Rule: In the Compatibility Organization, Donald Hebb introduced this. The fundamental rule of thumb is that the weight should be strengthened if the neuron comes from another nerve and both neurons are very active (the same sign).

Hopfield Law: Increase connectivity weight by the study rate and when both active and inactive the desired outcomes and inputs.

The Delta Rule: The idea is that the strengths of the input connections can constantly be modified so that the difference of the desired output to the actual output of the element is reduced (delta).

The Gradient Descent Rule: This is similar to the Delta rule because the transfer function derivative is still used to change the delta bug prior to the application of connection weights. However, a proportional additional constant associated with the learning rate accompanies the final weight - based change factor.

Kohonen's Law: This allows processing parts to acquire or update their weight. The strongest item is declared the winner and its competitors can be inhibited and its neighbours excited. Only the winner can adjust his weight and only the winner plus neighbors can adjust it.

3.3.5 Artificial Neural Network's Output Calculations

Sensitivity, septicity and accuracy are preferred statistics for determining the performance of a classifier. Susceptibility is the estimation rate for patients with epileptic diseases, speciality is the estimation speed for healthy people and accuracy is true. Equality. These statistical numbers are calculated using (36), (37) and (38).

$$Sensitivity = \frac{TP}{TP+FN} \tag{3.8}$$

$$Specificity = \frac{TN}{TN+FP} \tag{3.9}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{3.10}$$

In the above equations, the number of TP - diagnosed epilleptic patients, the total number of normal epileptic patients, for whose epileptic disease was diagnosed, and the total number of normal epileptic patients, for which FN was diagnosed.

3.4 BACKPROPAGATION

Backpropagation is a method used to compute the gradient required in artificial neural networks for the calculation of network weights. Backpropagation means "retroactive error propagation" as an error is calculated at the output and reversed across all layers of the network. It is frequently used to train deep neural networks.

Discharge is a common application of the Delta rule in multi - layer feed systems that permits iterating gradients to be measured using the chain rule for each layer. It is closely linked to Newton's Gauss algorithm and forms part of the ongoing research on the neural background.

The reverse propagation is a special case for an automatic differentiation technique. In learning, backpropagation is commonly used to adjust the weight of neurons through the gradient downward optimization algorithm to calculate the degree of loss function.

Consider, for example, the network for one training case: (1), 1,0 and therefore x1 and x2 inputs are 1 and 1. The result is parabolic, when y output is tracked against the error E on the vertical axis of the horizontal axis. The y output which reduces the E-error is the lowest parable. In a single training case, the minimum affects the horizontal axis, which means the error is zero and the network can produce an output y exactly matching the desired output t. In order to optimize the search for least errors, this reduces the problem of mapping inputs to outputs.

$$y = x_1 w_1 + x_2 w_2$$

where x1 and x2 weights are linked to the input unit connection with the output unit. The error therefore depends on the input weights of the neuron that must ultimately be changed over the network to enable learning. The result is a parabolic bowl with a separate horizontal axis, with each weight having a vertical axis fault. The same tract would require an elliptical paraboloid k+1 for a neuron with k weights.

Any complex system can be simply abstracted or at least dissected into its fundamental abstract components. The accumulation of multiple simple layers creates complexity. This post is intended to explain how neural networks work with the most straightforward abstraction. We try to reduce the mechanism of machine learning in NN to its fundamental abstract components. In contrast to other posts explaining neural networks, we try to use as few mathematical equations and code as possible and focus exclusively on the abstract concepts. A supervised neural network, at the highest and simplest abstract representation, can be presented as a black box with 2 methods learn and predict as following:

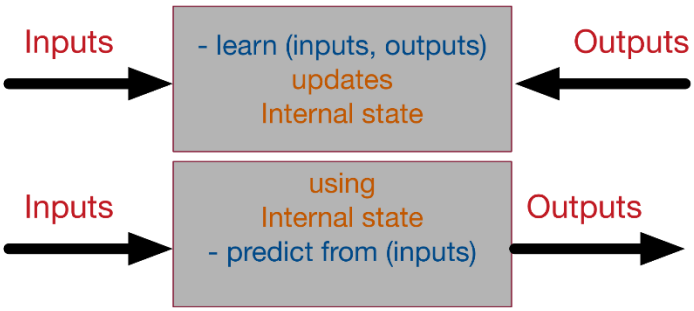

Figure 3.6: Backpropagation Training Method [24]

Learning as a problem of optimisation

It helps us first of all to understand the link between the actual neuron output and the exact output of a specific training example in order to obtain a mathematical derivation of the back propagation algorithm. The neural network has 2 inputs, a output unit and no hidden unit. Each neuron uses a linear output [note 1] as opposed to most neural networks, which is the weighted amount. At first, before training, weights are altered. In this instance, the neuron will learn from examples consisting of the tuples x1, x2, t where x1 and x2 are the network inputs and t is the correct output (network output should be produced when these inputs are trained). The first network calculates a y output which can differ from t (random weights) when x1 and x2 is specified. Squared error measurement is a common method for measuring differences between the expected t and the current y output:

$$E = (t - y)^2 \tag{3.11}$$

where E is an error or discrepancy.

Consider the network for one training case, for instance: (1), 1,0 and therefore x1 and x2 inputs are 1 and 1. The result is a parabolic when the y output on the vertical axis of the horizontal axis is tracked against the E error. The y output that minimizes the E error is the minimal parabola. The minimum also affects the horizontal axis in a single training case, which means the error is zero and that the network can produce an output y that exactly matches the desired output t. This reduces the problem of mapping inputs to outputs in order to optimize the search for the least error.

The neuron output depends, however, on the sum of all the inputs weighted:

$$y = x_1 w_1 + x_2 w_2 \tag{3.12}$$

where the weights of x1 and x2 are connected to the output unit by the input unit connection. The error depends therefore on the incoming weights of the neuron, which must be changed over the network ultimately in order to allow learning. The result is a parabelet bowl with a separate horizontal axis and a vertical axis fault with each weight. For the same tract, an elliptical paraboloid k+1 dimensions would be required for a neuron with k weights.

3.5 GENETIC ALGORITHM

The genetic algorithm biological metaphor is the evolution of species by the survival of the fittest, as Charles Darwin described. A new individual is produced by the crossover of genetic information of two parents in a population of animals or plants. The DNA stores the genetic data for the individual's construction. The genome of human DNA comprises 46 chromosome, four strings, abbreviated A, T, G and C. One of twenty amino acids, or a' start protein building' or' stopp protein building ' signal, is translated into three bases. A total of about 3 billion nucleotides are present. These can be structured into genes that contain information about the individual's construction in one or more parts. The vast majority of genes –the "junk" genes –are not being used, however, and only 3% of all genes contain meaningful information. Genetic data, the genome itself, are called the individual's genotype. The result is called a phénotype. The individual. The same genotype could lead to various phenotypes. This is clearly illustrated by the Twins. The process of natural evolution is simulated by a genetic algorithm. It aims at optimizing a number of parameters. The original idea includes the genetic information, called a parameter string, or a individual, in a bit string of a fixed length. An allle is called a possible value of a bit. A range of different encoding techniques are used in this thesis, but the fundamental principles also apply. A possible solution to this problem is provided by each parameter string. It includes information about building a neural network for the GANN issue. The fitness value is the quality of the solution. Crossover, selection and mutation are the fundamental GA operators. The main structure of a genetic algorithm is shown in Figure 1. It begins with the random generation of an early group of people, the original population.

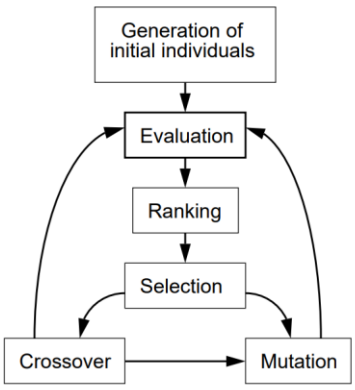

Figure 3.7: The Principle Structure of a Genetic Algorithm [7]

Evaluate and classify individuals. Since the number of persons is constant in each population, an old person, generally the one with the worst fitness value, must be discarded for each new individual. Two basic operators are available to create new people: mutation and intersection. Mutation is simpler. A few bits of the parameter string are rotated at random during mutation. Mutation may apply to offspring created by crossover or randomly to any person in the population as an independent operator.

1. Crossover

Crossover simulates a child's or two parents ' sexual generation. This is done by taking parts of one parent's bit string and the other parent and combining both parts of the child. The crossover consists of three main types: one, two and uniform. Figure 1.4 shows one-point crossover. At the same point, the two parent bit strings are cut. So one part of each parent can generate the child. Notice not moving the parts. The random cutting point is also independent of the true significance of the bits. One bit string parameter can be encoded in more than one bit, with the encoding cut to a new value, different than the two parents, during the crossover.

2. Selection

The choice of people for crossover and mutation is prejudicial to good people. The chance of a selected person is based on the relative fitness of the classical roulette wheel population.

3. Mutation

In real evolution, by misrepresentation or other gene deformations e.g. gamma radiation, the genetic material can randomly modify itself. Mutations can be performed with a certain probability in genetic algorithms as random deformations of the strings. It is positive that genetic diversity is maintained and the local maxima avoided.

4. Sampling

From the first and the next generation, this process calculates. We can indicate the following substantial differences in comparison with conventional continuous optimization methods, such as Newton or the descent gradient:

1. GA manipulates coded versions of the problem parameters, i.e. search room S instead of X itself instead of parameters themselves.

2. While looking for nearly all conventional methods, GAs are always working on a whole population of items (cords). The force of genetic algorithms is greatly enhanced. The chance to reach the best level in the world has been improved and the risk of being caught in a fixed region is reduced vice versa.

3. No additional information about objective value of function like derivatives is used by normal genetic algorithms. It can therefore be used in the event of continuous or discreet optimization problems. Only a valuable decoding function should be specified.

4. Deterministic transitional operators are applied by GAs while conventional continuous optimisation methods. In particular, there are some random components of how a new generation is determined from the current one (examples are shown later).

3.5.1 Binary Strings Genetic Operations

3.5.1.1 Selection

The solution algorithm is chosen by preferred fitness people rather than fitness fitness people. It may be a decisive task, but it has random components in most implementations. The following version is very popular now, where the chances of selecting a certain person are proportional to their health, (We will give a theory explaining the good properties later). A random function experiment can be considered:

$$P[b_{j,t} \text{ is selected}] = \frac{f(b_{j,t})}{\sum_{k=1}^{m} f(b_{k,t})} \tag{3.13}$$

Of course, only when all fitness values are good is this formula meaningful. Unless this is the case, in the simplest case the transformation 5 - 0 · does not decrease. (In the simplest case the shift should be applied). The likelihood can be expressed

$$P[b_{j,t} \text{ is selected}] = \frac{\varphi(f(b_{j,t}))}{\sum_{k=1}^{m} \varphi(f(b_{k,t}))} \tag{3.14}$$

By random test, a generalized Roulette Game, to some degree we can force the property (2.1) to be met. The slotes are not as wide in this roulette game, i.e. the various results can take place with different probabilities. The graphic indication of how this wheel game works is provided by Figure 2.1.

3.5.1.2 Crossover

When parents mix up their gametes, the genetic materials of both parents are mixed as they appear in the real world. As a result, certain genes are generated by one parent and the other parents. Chromosomes are generally divided and combined randomly.

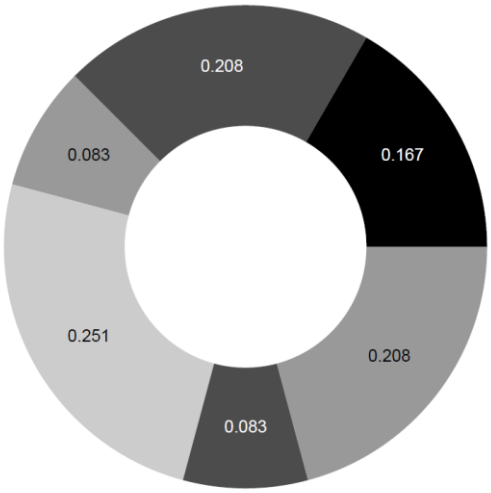

Figure 3.8: A graphic representation of selection of roulette wheels with 6 alternatives m. The numbers within the arcs correspond to the likelihood of selecting the alternative.

This is known as the crossover mechanism. It is an excellent tool for the introduction of new genetic resources and for preserving genetic diversity, but also for providing better children or good parents. Several studies have found that crossover is why species sexually adapted to reproduce faster than those that asexually replicate. Crossover is primarily gene exchange for both parents ' chromosomes. You can achieve this by randomly selecting two strings and in the simplest case, changing the two tails. In figure 2.2 this process is called a crossover with one point below.

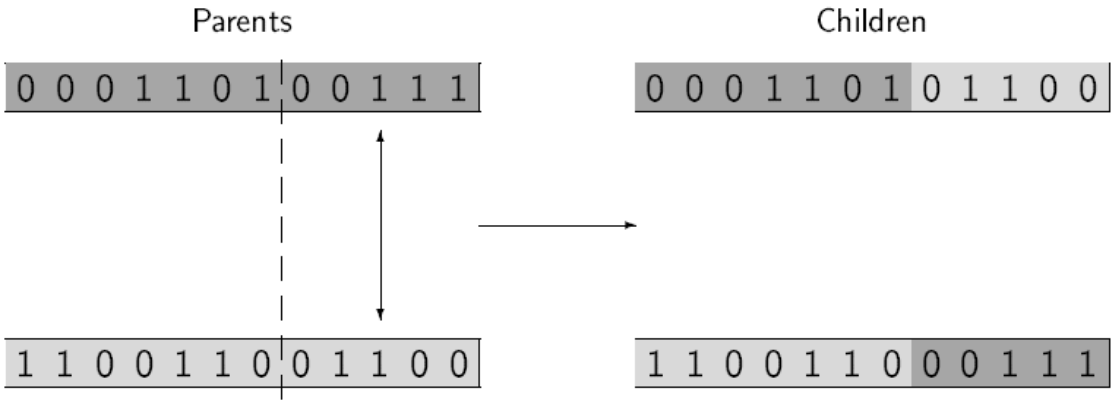

Figure 3.9: One-point crossover of binary strings

One - point crossover for Gas that operates on binary strings is a simple and common method. Additional methods of crossover can be useful or even necessary for other problems or different coding.

N-point crossover: N breaking points are randomly selected instead of just one. Each section is exchanged. Two crossover points are especially important for this class.

Segmented crossover: This is similar to N - point crossroads, as the breaking points differ.

Uniform crossover: It is randomly decided whether the positions are exchanged for each position.

Shuffle crossover: The first two parents receive a randomly selected permutation, then the shuffled parents receive the N point cross, and then the shuffled kids return the other way round.

3.5.2 Mutation

Mutation — a random deformation of an individual's genetic data by radioactivity or other environmental influences is the ultimate part of our genetic algorithm. The probability of a mutation of a certain gene for all genes is nearly equal. In real reproduction Pm should of course be quite low to prevent the GA from acting like random searches chaotically. Again, it is coding and the problem itself that are the choice of the correct mutation method, similar to a crossover.

Single bit reversal: One bit is randomly rejected with probability Pm.

Bitwise inversion: The entire string with prob pm is gradually inverted.

Random selection: The string is replaced by an altered selected one with the probability Pm.

3.5.3 Examples

3.5.3.1 Simple Example

Take into account the problem of finding the maximum global function:

$$f_1: \{0, \dots, 31\} \to R$$

$$x \to x^2 \tag{3.15}$$

Of course, it is obvious but it is possible to calculate some steps alone in order to obtain insight into genetic algorithms by the simplicity of this problem.

Of courses, a good string space along with the relevant coding and decryption scheme will naturally be the first step in the checklist. In order to do GA work, this is necessary. In this example, S >{ 0, 1}5, where a value of{ 0 is close to hand, can be considered. The binary representation of 31} is coded. A string is decoded as a consequence

$$c(s) = \sum_{i=0}^{4} s[4-i].2^i \tag{3.16}$$

Assume that with a population size of m = 4 we have Algorithm 2, 5 as it is with a crossover likelihood of Pc = 1 and Pm mutation likelihood = 0,001. When we randomly calculate the first generation with a consistent distribution over {0, 1}5, in the first step we obtain the following:

Individual No.	String (genotype)	*x* value (phenotype)	f(x) = x^2	Pselect$_i$ $\frac{f_i}{\sum f_i}$
1	0 1 1 0 1	13	169	0.14
2	1 1 0 0 0	24	576	0.49
3	0 1 0 0 0	8	64	0.06
4	1 0 0 1 1	19	361	0.31

The fitness value is 1170, where the average is 293 and the maximum is 576. It can easily be calculated. The last column shows that proportional selecting favors people with high fitness conditions (as No. 2) compared to those with low fitness (as No. 3).

Set of selected individuals	Crossover site (random)	New population	X value	F(x) X^2
0 1 1 0 |1	4	0 1 1 0 0	12	144
1 1 0 0 | 0	4	1 1 0 0 1	25	625
1 1 | 0 0 0	2	1 1 0 1 1	27	729
1 0 | 0 1 1	2	1 0 0 0 0	16	256

Thus we obtain an average of 439 and a maximum of 729 fitness values from the new generation of 1754. This very basic example shows how selection favors qualified people and how cross-over between two parents can create a baby even better than the two parents. The reader is left to follow this example as an exercise.

3.5.3.2 A One - Dimensional Oscillating Feature

Now we want to know how much the function can achieve globally

$$f_2 : [-1, 1] \rightarrow R \tag{3.17}$$

$$x \rightarrow 1 + e^{-x^2}.\cos(36x) \tag{3.18}$$

The function can be seen from the plots in Figure 2.3 and has a global maximum of 0 and numerous local maximums. Firstly, the search space [−1, 1] must be discretized in order to work with binary strings. A uniform grid of 2n points is common practice and grid points are listed then and the binary representation of the index is employed as a codification.

The algorithm reaches the global high point after 52 generations, with an average of 52x 6= 312 fitness evaluations, and the search area's total size is 216= 65536. At least in this example, we can conclude that the GA is definitely better than the random search or the comprehensive method which scans the entire search area dumbly. Take 011111111111111 to find out more about the

coding / decoding scheme. The number of the integer is 32767. Decoding function rates are computed

$$-1 + 32767 \,.\frac{1-(-1)}{65535} = \; -1 + 0.9999847 = \; -0.0000153 \tag{3.19}$$

3.5.3.3 A Two-Dimensional Function

In this example we find that the GA definitely is much faster than a big or random algorithm for research. The problem comes with a smooth f3, which results can be achieved using a standard method by changing the initial values. The maximum global (0, 0) is obviously circled by the minimum ring and in this example has a radius of $\pi/2$. The global maximum is likely to converge when this is an original value, but only for example when BFGS is used (Broyden Fletcher Goldfarb Shanno, an efficient Quasy - Newton method for continuous uncontrolled optimization of functions [10]). The chance of the corresponding neighborhood value is the maximum when the initial value is assumed randomly by [−10, 10].

$$\frac{\left(\frac{\pi}{2}\right)^2 . \, \pi}{10 \,.10} = \; \frac{\pi^3}{400} = 0.0775 \tag{3.20}$$

Consequently, until an initial value is reached, the expected test number is 1/0.0775/13. The BFGS method with line search required the right global optimum during test implementation of the BFGS method for 15 tests (random initial values). For these computations, the total time for SGI O2 (MIPS R5000/180SC) was 5 milliseconds. As mentioned above, it took 1.5 seconds for the genetic algorithm to be compared to the global optimum.

This example demonstrates that GAs are not necessarily quick. In addition, in many cases the methods involved in derivatives are much slower than conventional ones. The next example, however, dramatically demonstrates that even fluid functions in conventional optimization techniques are difficult to implement.

3.5.4 Genetic Programming

In their daily practice, mathematicians and computer scientists only search for programs that correctly solve the problems. You usually attempt to design such programs based on your knowledge, underlying principles, mathematical models, intuition and so on. The questions of Koza look somehow utopian and provocative. However, his answers are remarkable and worthy of further discussion here. The basic idea is simple but appealing–the problem of automatic program induction should involve genetic algorithms. All we need to do is change all the genetic techniques that we have talked about so far. Random initiation, connection and mutation are involved. We need nothing new to choose and sample, because these processes are independent from the display of the data.

Of course, that sounds great. However, the question arises as to whether this type of genetic programming (GP) is working. Koza starts with a rather vague hypothesis in his remarkable monograph [30].

The Paradigm of genetic programming: Provided that we have a resolvable problem, a definition for a suitable programme, a sufficient number of representative test examples, WWa genetic algorithm can determine which (approximately) program solves the problem and provides us with sufficient representative test examples. It appears to be a question of faith. This hypothesis has not been demonstrated by anyone so far and it is doubtful whether this is ever possible. Rather than providing evidence, Koza has developed many well - chosen examples which empirically highlight his hypothesis. The successful solutions with GP include:

- Control of the process (inverted pendulum bang bang control)
- Logistics (simple robotic control, problems with stacking)
- Automatic scheduling (pseudo - random number generators, ANN design)
- Game strategies (Poker, Tic Tac Toe)
- Inverse kinematics
- Classification

- Symbolic computation:
    - Sequence induction (Fibonacci sequence, etc.)
    - Symbolic regression
    - Solve equations (functional, differential and integral equation power series - based solutions)
    - Symbolic differences and integrity
    - Trigonometric identity discovery automatically

A brief introduction to genetic programming has been given in this chapter. We will focus on the basic methodological problems and will omit detailed examples.

3.6 TRAINING OF ARTIFICIAL NEURAL NETWORKS

The neural biological computing systems that make up animal brains are artificial neural networks or connecting systems. Such systems learn tasks by following examples, usually without task-specific programming (a step towards improvements in performance). You can learn how to identify images that contain the Cats and use the results to identify the Cats in the other images through the analysis of flames that manually mark "cat" or "no-cat." They don't know cats, for example, fur, tails, whiskers and cat - like faces. Rather, from the learning material they are processing they develop their own set of relevant features. An ANN is based on an artificial neuronal collection which is linked with the unit or node (analogous to biological neurons in the animal brain). Any link between artificial neurons (similar to a synapse) can signal them. The artificial neuron receiving the signal can proceed and signal the connected artificial neurons.

In the common ANN implementation, the signal is an actual value in the relation between artificial neurons, and the output is calculated by a nonlinear input sum function from each artificial neuron. Artificial neurons and connections usually weigh in accordance with learning. The weight increases or lowers the signal strength of the connection. An artificial neuron threshold can only be present if the aggregate signal crosses this threshold. Artificial neurons are usually arranged in

layers. Inputs can be converted into different types by different layers. Signals travel from the first layer to the last layer (output), maybe several times after the layers have been passed through.

The original objective of the ANN approach was to solve problems like the human brain. The focus was over time on matching specific mental skills that lead to biodiversity. Different tasks such as computer vision, speech recognition, machine translation, social network filtering, video games and playboards and medical diagnosis are performed using ANNs. This section shows a training process that optimizes ANN performance through BP and GA algorithms.

3.6.1 Training Artificial Neural Networks with Backpropagation.

The method of gradient descent includes the derivatives of the squared error function with respect to the weight of the network. This is usually done in return. The squared error function takes the following:

$$E = \frac{1}{2}(t - y)^2 \tag{3.21}$$

where

E is the error squared,

t is the target output for a sample training, and

y is the current output of neuron output.

The 1/2 factor in the exponent is distinguished. Then the term is increased by an arbitrary learning rate. Whether a continuous coefficient is implemented now is not important.

For each neuron *j*, its output o_j is defined as

$$o_j = \varphi(net_j) = \varphi\left(\sum_{k-1}^{n} w_{kj} o_k\right) \tag{3.22}$$

The netj input for the neuron is the weighted sum of previous neuron outputs. The ok of the input layer is simply a xk input in the network. If the neuron is in the first layer after the input layer. The input units in the neuron are numbered by n. The wkj variable shows the weight from the neuronal to the j.

β is non - linear and distinctive with the activation funktion. The logistic function is a commonly used activation feature:

$$\varphi(z) = \frac{1}{1+e^{-z}} \tag{3.23}$$

which has a convenient derivative of:

$$\frac{d\varphi}{da} = \varphi(a)(1 - \varphi(a)) \tag{3.24}$$

Identifying the error derivative

Twice by the chain rule is calculated the partial derivation of an error in relation to a wij:

$$\frac{\vartheta E}{\vartheta w_{ij}} = \frac{\vartheta E}{\vartheta o_j} \frac{\vartheta o_j}{\vartheta net_j} \frac{\vartheta net_j}{\vartheta w_{ij}} \tag{3.25}$$

In the last right side factor of the foregoing w_{ix} depends only one term in the sum net_h

$$\frac{\vartheta E}{\vartheta net_j} = \frac{\vartheta}{\vartheta w_{ij}} \varphi\left(\sum_{k-1}^{n} w_{kj} o_k\right) = \frac{\vartheta}{\vartheta w_{ij}} w_{ij} o_i = o_i \tag{3.26}$$

When the first layer of the neuron after the input layer is, o_i is only X_I.

With respect to the input, only the active function parameters are a derivative of neuron j output (if logistics function is used):

$$\frac{\vartheta o_j}{\vartheta net_j} = \frac{\vartheta}{\vartheta net_j} \varphi(\vartheta net_j) = \varphi(\vartheta net_j)(1 - \varphi(\vartheta net_j)) \tag{3.27}$$

That is why a differentiated activation function is required for back propagation. (The activation function of ReLU, which cannot be differentiated by 0, is very popular in the recent past, for instance in AlexNet)

The first factor to evaluate is simple, because then oj = y and if the neuron is in the output layer.

$$\frac{\vartheta E}{\vartheta o_j} = \frac{\vartheta E}{\vartheta y} = \frac{\vartheta}{\vartheta y}\frac{1}{2}(t-y)^2 = y - t \tag{3.28}$$

But if j is within an arbitrary inner network layer, it is less obvious to determine derivative E with respect to the oj.

$$\frac{\vartheta E(o_j)}{\vartheta o_j} = \frac{\vartheta E(net_1, net_2 \dots net_u)}{\vartheta o_j} \tag{3.29}$$

Taking into account E as a function where all neurons are inputs (L=1, 2 ... u)

$$\frac{\vartheta E}{\vartheta o_j} = \sum_{\partial \in L} \frac{\vartheta E}{\vartheta net_j}\frac{\vartheta net_j}{\vartheta o_j} = \sum_{\partial \in L} \frac{\vartheta E}{\vartheta o_j}\frac{\vartheta o_j}{\vartheta net_j} w_{j\partial} \tag{3.30}$$

Thus, if all derivatives are known with respect to oj — that which is closer to the neuron output — the derivative with regards to oj can be calculated. Combining everything:

$$\frac{\vartheta E}{\vartheta w_{ij}} = \partial_j o_i \tag{3.31}$$

In 1960 Henry J. Kelley and 1961 Arthur E. Bryson derived the basics of continuous back-propagation from control theory. They have used dynamic programming principles. A simple derivation based only on the chain rule was published in 1962 by Stuart Dreyfus. This is a multi - phase method used to optimize dynamic systems by Bryson and Ho in 1969.

In the beginning of the 1960's, background propagation was derived by a number of researchers and launched by Seppo Linnainmaa as early as 1970, for instance Arthur E. Bryson and Yu-Chi Ho, among the researchers of the 1960s. After a careful analysis in his PhD dissertation in 1974, the first person to propose use for neural networks in the United States was Paul Werbos. The prize for work of David E. Rumelhart was awarded in 1986 to Geoffen E. Hinton, Ronald J. Williams & James McClelland.

The general automated differentiation (AD) method of discernible connected networks for nestled differentiating functions was issued by Linnainmaa in 1970. This is the backbone that even with sparsely networked networks is efficient. Backpropagation was used in 1973 by Dreyfus to adapt controller parameters in proportion with error gradients. Werbos noted the potential to apply this

principle to neural artificial networks in 1974, and in its current application on neural networks employed the Linnainmaa AD Method in 1982.

In 1986, Rumelhart, Hinton and Williams proved that the process could produce useful internal images of inbound data in hidden layers in neural networks. Wan was the first winner in 1993 of an international competition for pattern recognition. In the 2000s it was disadvantageous, but it came back in 2010 with cheap, high-performance computer systems based on GPUs. In particular, research was conducted into linguistic structures, where connectivity models could explain a number of first language and second language learning phenomes using this algorithm.

4. IMPLEMENTATION AND RESULTS

This chapter presents the extensive simulation results for methods investigated in this project is Genetic Algorithm optimized structured Artificial Neural Network trained by Backpropagation GA (ANN-BP) by using research data source (Lung Cancer Dataset). We implemented the ANN using GA algorithm to optimize the parameters of ANN to train and test this research's dataset using BP in order to measure the different performance parameters.

Comparative Results

We used the 70 % training and 30 % testing scenario with varying number neurons of the hidden layer is 20 GA (ANN-BP).

Diabetes Dataset Results

First we present the individual for GA (ANN-BP) using Lung Cancer dataset. We used 5 neurons the hidden layer. Figure 1 and 2 are showing the fitness or Root Mean Squared Error (RMSE) or error outcomes by using the existing GA (ANN-BP) approach for 100 iterations for GA and 200 for backpropagation.

Figure 1 contains the error calculation process for 100 iterations GA for parameters and 200 iterations for artificial neural network training with 5 neurons in the hidden layer. Figure 2 contains the prediction and the classification of artificial neural network compared with the targets attribute in the dataset.

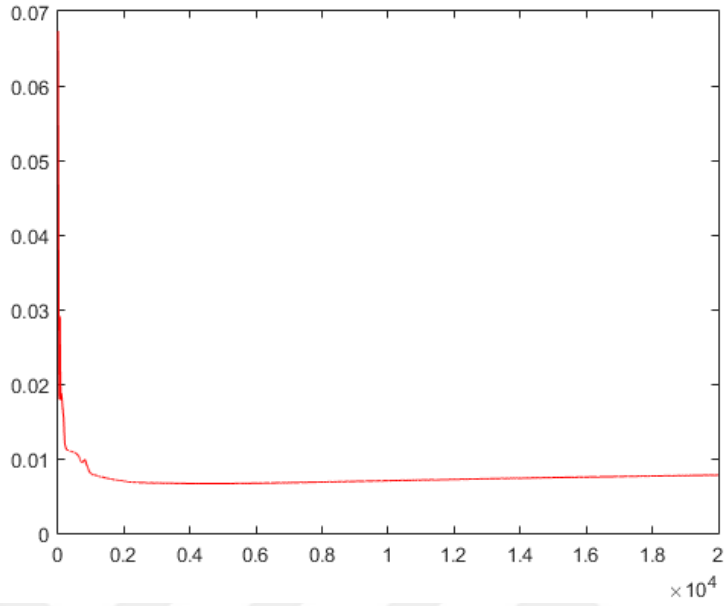

Figure 4.1: Error graph performance using GA (ANN-BP)

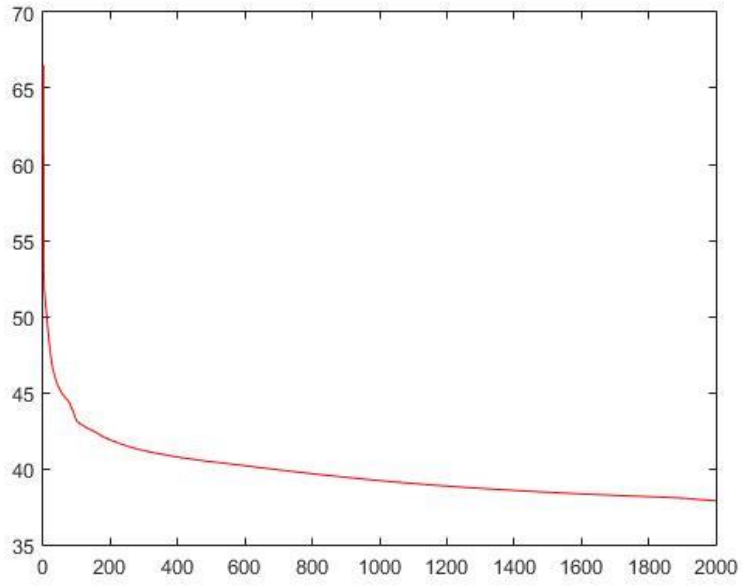

Figure 4.2: Error graph performance using GA.

Table 4.1: The Results of Both GA(ANN-BP) and GA

	GA(ANN-BP)	GA
Training Error (Fitness/RMSE)	0.1067	0.13050075872534137
Training Accuracy	89.33%	86.949924127465863
Testing Error(Fitness/RMSE)	0.08726	0.14110091743119256
Testing Accuracy	91.274%	85.889908256880744
Training Sensitivity	0.950079	0.912047
Training Specificity	0.883084	0.837574032
Testing Sensitivity	0.949879	0.892587
Testing Specificity	0.908452	0.84256799

Table 4.1 contains the results for the method according to Figure 4.1 and Figure 4.2 which show the training error and accuracy, testing error and accuracy (using RMSE as a fitness function) and both the specificity and sensitivity for training and testing.

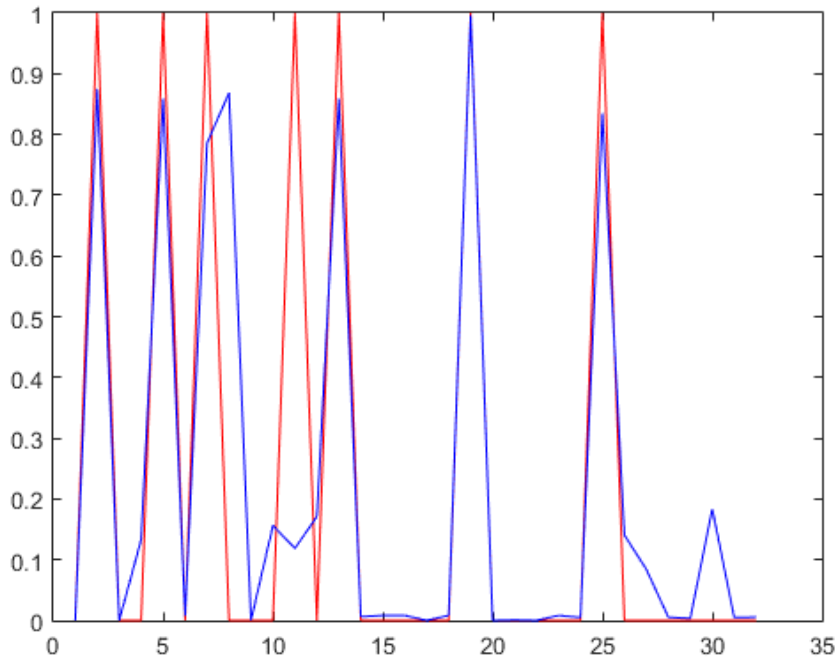

Figure 4.3: Testing performance of GA(ANN-BP)

5. CONCLUSION

The comparative results showing that using the advantages novel BP algorithm with GA parameter modification we can able to optimize the performance of ANN training and testing to solve the real time problems. From these experiments, we observed that the fitness functions that generated the ANN with the best weighted recognition rate were those that used the classification error. The modified BP was compared in terms of the accuracy, error rate, sensitivity rate, specificity rate and accuracy rate for both training and testing perspective with other researchers. The modified BP algorithm achieved the greate performance. The transfer functions that more often were selected for each algorithm were: the Gaussian functions for the basic BP algorithm; the sinusoidal function for modified BP algorithm. In general, the ANNs designed with the proposed methodology were very promising. The proposed methodology automatically designs the ANN based on determining the set connections, the number of neurons in hidden layers, the adjustment of the synaptic weights, the selection of bias, and transfer function for each neuron.

5.1 SUGGESTIONS

In this research work, we perform the ANN training using modified BP algorithm using the research datasets. For future suggestions we want to perform below points:

- The current evaluation is based on single class problems for classification, however under the real time scenario this will not be the case always. So we suggest evaluating the performance of proposed model using multi-class datasets.

Second point is, the consideration of more real time will be the interesting future direction for this research work.

REFERENCES

[1] G. Beni and J. Wang, “Swarm intelligence in cellular robotic systems,” in Robots and Biological Systems: Towards a New Bionics? vol. 102 of NATO ASI Series, pp. 703–712, Springer, Berlin, Germany, 1993.

[2] X. Yao, “Evolving artificial neural networks,” Proceedings of the IEEE, vol. 87, no. 9, pp. 1423–1447, 1999.

[3] E. Alba and R. Martí, Metaheuristic Procedures for Training Neural Networks, Operations Research/Computer Science Interfaces Series, Springer, New York, NY, USA, 2006.

[4] J. Yu, L. Xi, and S. Wang, “An improved particle swarm optimization for evolving feedforward artificial neural networks,” Neural Processing Letters, vol. 26, no. 3, pp. 217–231, 2007.

[5] M. Conforth and Y. Meng, “Toward evolving neural networks using bio-inspired algorithms,” in IC-AI, H. R. Arabnia and Y. Mun, Eds., pp. 413–419, CSREA Press, 2008.

[6] Y. Da and G. Xiurun, “An improved PSO-based ANN with simulated annealing technique,” Neurocomputing, vol. 63, pp. 527–533, 2005.

[7] X. Yao and Y. Liu, “A new evolutionary system for evolving artificial neural networks,” IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 694–713, 1997.

[8] D. Rivero and D. Periscal, “Evolving graphs for ann development and simplification,” in Encyclopedia of Artificial Intelligence, J. R. Rabuñal, J. Dorado, and A. Pazos, Eds., pp. 618–624, IGI Global, 2009.

[9] H. M. Abdul-Kader, “Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting34,” International Journal of Computer Science and Network Security, vol. 9, no. 3, pp. 92–99, 2009.

[10] K. K. Kuok, S. Harun, and S. M. Shamsuddin, “Particle swarm optimization feedforward neural network for modeling runoff,” International Journal of Environmental Science and Technology, vol. 7, no. 1, pp. 67–78, 2010.

[11] B. A. Garro, H. Sossa, and R. A. Vázquez, “Back-propagation vs particle swarm optimization algorithm: which algorithm is better to adjust the synaptic weights of a feed-forward ANN?” International Journal of Artificial Intelligence, vol. 7, no. 11, pp. 208–218, 2011.

[12] B. Garro, H. Sossa, and R. Vazquez, “Evolving neural networks: a comparison between differential evolution and particle swarm optimization,” in Advances in Swarm Intelligence, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds., vol. 6728 of Lecture Notes in Computer Science, pp. 447–454, Springer, Berlin, Germany, 2011.

[13] B. A. Garro, H. Sossa, and R. A. Vazquez, “Design of artificial neural networks using a modified particle swarm optimization algorithm,” in Proceedings of the International Joint Conference on Neural Networks (IJCNN '09), pp. 938–945, IEEE, Atlanta, Ga, USA, June 2009.

[14] B. Garro, H. Sossa, and R. Vazquez, “Design of artificial neural networks using differential evolution algorithm,” in Neural Information Processing. Models and Applications, K. Wong, B. Mendis, and A. Bouzerdoum, Eds., vol. 6444 of Lecture Notes in Computer Science, pp. 201–208, Springer, Berlin, Germany, 2010.

[15] B. A. Garro, H. Sossa, and R. A. Vazquez, “Artificial neural network synthesis by means of artificial bee colony (abc) algorithm,” in Proceedings of the IEEE Congress on Evolutionary Computation (CEC '11), pp. 331–338, IEEE, New Orleans, La, USA, June 2011.

[16] R. C. Eberhart, Y. Shi, and J. Kennedy, Swarm Intelligence, The Morgan Kaufmann Series in Evolutionary Computation, Morgan Kaufmann, Boston, Mass, USA, 1st edition, 2001.

[17] M. Chen, “Second generation particle swarm optimization,” in Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08), pp. 90–96, Hong Kong, June 2008.

[18] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in Proceedings of the Congress on Evolutionary Computation (CEC '99), vol. 3, IEEE, Washington, DC, USA, July 1999.

[19] M. Løvberg, T. K. Rasmussen, and T. Krink, “Hybrid particle swarm optimizer with breeding and subpopulations,” in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01), pp. 469–476, Morgan Kaufmann, San Francisco, Calif, USA, July 2001.

[20] N. Higashi and H. Iba, “Particle swarm optimization with Gaussian mutation,” in Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03), pp. 72–79, IEEE, April 2003.

[21] S. Mohais, R. Mohais, C. Ward, and C. Posthoff, “Earthquake classifying neural networks trained with random dynamic neighborhood PSOs,” in Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07), pp. 110–117, ACM, New York, NY, USA, July 2007

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, D. E. Rumelhart, J. L. McClelland, and PDP Research Group, Eds., pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.

[23] J. A. Anderson, An Introduction to Neural Networks, The MIT Press, 1995.

[24] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990.

[25] D. N. A. Asuncion, UCI machine learning repository, 2007.

[26] R. A. V. E. De Los Monteros and J. H. Sossa Azuela, “A new associative model with dynamical synapses,” Neural Processing Letters, vol. 28, no. 3, pp. 189–207, 2008.

[27] B. A. Garro and R. A. Vázquez, “Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms,” *Computational Intelligence and Neuroscience*, 2015.

[28] V. G. Gudise and G. K. Venayagamoorthy, “Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks,” in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS’03 (Cat. No.03EX706)*, 2003, pp. 110–117.

[29] “Pima Indians Diabetes Database.” [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database. [Accessed: 20-Dec-2018].

[30] BAGLEY, J. D. The Behavior of Adaptive Systems Which Employ Genetic and Correlative Algorithms. PhD thesis, University of Michigan, Ann Arbor, 1967.

[31] BAUER, P., BODENHOFER, U., AND KLEMENT, E. P. A fuzzy algorithm for pixel classification based on the discrepancy norm. In Proc. 5th IEEE Int. Conf. on Fuzzy Systems (New Orleans, September 1996), vol. III, pp. 2007–2012.

[32] BODENHOFER, U. Tuning of fuzzy systems using genetic algorithms. Master’s thesis, Johannes Kepler Universit¨at Linz, March 1996.

[33] BODENHOFER, U., AND BAUER, P. A formal model of interpretability of linguistic variables. In Interpretability Issues in Fuzzy Modeling, J. Casillas, O. Cord´on, F. Herrera, and L. Magdalena, Eds., vol. 128 of Studies in Fuzziness and Soft Computing. Springer, Berlin, 2003, pp. 524–545.

[34] BODENHOFER, U., AND HERRERA, F. Ten lectures on genetic fuzzy systems. In Preprints of the International Summer School: Advanced Control—Fuzzy, Neural, Genetic, R. Mesiar, Ed. Slovak Technical University, Bratislava, 1997, pp. 1–69.

[35] BODENHOFER, U., AND KLEMENT, E. P. Genetic optimization of fuzzy classification systems — a case study. In Computational Intelligence in Theory and Practice, B. Reusch and K.-H. Temme, Eds., Advances in Soft Computing. Physica-Verlag, Heidelberg, 2001, pp. 183–200.

[36] BONARINI, A. ELF: Learning incomplete fuzzy rule sets for an autonomous robot. In Proc. EUFIT’93 (1993), vol. I, pp. 69–75. 121 122 BIBLIOGRAPHY

[37] BONARINI, A. Evolutionary learning of fuzzy rules: Competition and cooperation. In Fuzzy Modeling: Paradigms and Practice, W. Pedrycz, Ed. Kluwer Academic Publishers, Dordrecht, 1996, pp. 265–283.

[38] BONARINI, A. Anytime learning and adaptation of hierarchical fuzzy logic behaviors. Adaptive Behavior 5 (1997), 281–315.

[39] BULIRSCH, R., AND STOER, J. Introduction to Numerical Analysis. Springer, Berlin, 1980.

APPENDIX

```
# -*- coding: utf-8 -*-

"""

Created on We Jan 23 15:22:03 2019

@author: Mohammed Khalaf Abdullah

"""


# Back-Propagation Neural Networks supported by Genetic algorithm for parameter optimization

#

# Written in Python.  See http://www.python.org/


import math

import random

import string

import csv

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt



random.seed(0)
```

```
# calculate a random number where:  a <= rand < b
def rand(a, b):
    return (b-a)*random.random() + a

# Make a matrix (we could use NumPy to speed this up)
def makeMatrix(I, J, fill=0.0):
    m = []
    for i in range(I):
        m.append([fill]*J)
    return m

# our sigmoid function, tanh is a little nicer than the standard 1/(1+e^-x)
def sigmoid(x):
    return math.tanh(x)

# derivative of our sigmoid function, in terms of the output (i.e. y)
def dsigmoid(y):
    return 1.0 - y**2

class NN:
```

```
def __init__(self, ni, nh, no):
    # number of input, hidden, and output nodes
    self.ni = ni + 1 # +1 for bias node
    self.nh = nh
    self.no = no

    # activations for nodes
    self.ai = [1.0]*self.ni
    self.ah = [1.0]*self.nh
    self.ao = [1.0]*self.no

    # create weights
    self.wi = makeMatrix(self.ni, self.nh)
    self.wo = makeMatrix(self.nh, self.no)
    # set them to random vaules
    for i in range(self.ni):
        for j in range(self.nh):
            self.wi[i][j] = rand(-0.2, 0.2)
    for j in range(self.nh):
        for k in range(self.no):
            self.wo[j][k] = rand(-2.0, 2.0)
```

```
    # last change in weights for momentum
    self.ci = makeMatrix(self.ni, self.nh)
    self.co = makeMatrix(self.nh, self.no)


def update(self, inputs):
    if len(inputs) != self.ni-1:
        raise ValueError('wrong number of inputs')


    # input activations
    for i in range(self.ni-1):
        #self.ai[i] = sigmoid(inputs[i])
        self.ai[i] = inputs[i]


    # hidden activations
    for j in range(self.nh):
        sum = 0.0
        for i in range(self.ni):
            sum = sum + self.ai[i] * self.wi[i][j]
        self.ah[j] = sigmoid(sum)
```

```
    # output activations
    for k in range(self.no):
        sum = 0.0
        for j in range(self.nh):
            sum = sum + self.ah[j] * self.wo[j][k]
        self.ao[k] = sigmoid(sum)

    return self.ao[:]

def backPropagate(self, targets, N, M):
    if len(targets) != self.no:
        raise ValueError('wrong number of target values')

    # calculate error terms for output
    output_deltas = [0.0] * self.no
    for k in range(self.no):
        error = targets[k]-self.ao[k]
        output_deltas[k] = dsigmoid(self.ao[k]) * error

    # calculate error terms for hidden
    hidden_deltas = [0.0] * self.nh
```

```
for j in range(self.nh):
    error = 0.0
    for k in range(self.no):
        error = error + output_deltas[k]*self.wo[j][k]
    hidden_deltas[j] = dsigmoid(self.ah[j]) * error

# update output weights
for j in range(self.nh):
    for k in range(self.no):
        change = output_deltas[k]*self.ah[j]
        self.wo[j][k] = self.wo[j][k] + N*change + M*self.co[j][k]
        self.co[j][k] = change
        #print N*change, M*self.co[j][k]

# update input weights
for i in range(self.ni):
    for j in range(self.nh):
        change = hidden_deltas[j]*self.ai[i]
        self.wi[i][j] = self.wi[i][j] + N*change + M*self.ci[i][j]
        self.ci[i][j] = change
```

```
    # calculate error
    error = 0.0
    for k in range(len(targets)):
        error = error + 0.5*(targets[k]-self.ao[k])**2
    return error
def test(self, patterns):
    for p in patterns:
        print(p[0], '->', self.update(p[0]))

def weights(self):
    print('Input weights:')
    for i in range(self.ni):
        print(self.wi[i])
    print()
    print('Output weights:')
    for j in range(self.nh):
        print(self.wo[j])

def train(self, patterns, iterations=10000, N=0.5, M=0.1):
    # N: learning rate
    # M: momentum factor
```

```
        Err=[0] * iterations;
        for i in range(iterations):
            error = 0.0
            for p in patterns:
                inputs = p[0]
                targets = p[1]
                self.update(inputs)
                error = error + self.backPropagate(targets, N, M)
                Err[i] = error;
            if i % 100 == 0:
                print('error %-.5f' % error)
            if i==iterations-1:
                plt.plot(Err)


def demo():
    # Teach network XOR function
    reader=pd.read_excel('bc.xlsx')
    newreader=np.matrix(reader)
    pat = newreader.tolist()


    # create a network with two input, two hidden, and one output nodes
```

```
    n = NN(3, 2, 1)
    # train it with some patterns
    n.train(pat)
    # test it
    n.test(pat)
    print(pat)
    print(newreader)


if __name__ == '__main__':
    demo()
```