T.C.

İSTANBUL ALINBAŞ ÜNİVERSİTESİ

Electrical and Computer Engineering

DAIBETES DIAGNOSIS USING MACHINE LEARNING

Alaa Badr Eysa

Master Thesis

Asst. Prof. Dr. Sefer Kurnaz

İstanbul 2019

DAIBETES DIAGNOSIS USING MACHINE LEARNING

by

Alaa Badr Eysa

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

I give my best efforts to my dear father and to my dear mother who have done everything they can for supporting me, and also thanks and appreciation to all those who stood with me and supported me in my studies.

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

DAIBETES DIAGNOSIS USING MACHINE LEARNING

Eysa, Alaa Badr

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

Supervisor: Asst Prof. Dr. Sefer kurnaz

Date: March 2019

Pages: 51

Artificial neural networks have been in the position of producing complex dynamics in control applications over the last decade, especially when they are linked to feedback. Although ANNs are strong for network design, the harder the design of the network, the more complex the desired dynamic is. Many researchers tried to automate the design process of ANN using computer programs. Search and optimization problems can be considered as the problem of finding the best parameter set for a network to solve a problem. Recently, the problem of optimizing ANN parameters to train different research datasets has been targeted by two commonly used stochastic genetic algorithms (GA) and particle swarm optimization (PSO). The process based on the neural network is optimized with GA and PSO to enable the robot to perform complex tasks. However, using such optimization algorithms to optimize the ANN training process cannot always be balanced or successful. These algorithms simultaneously aim to develop three main components of an ANN: synaptic weight, connections, architecture and transfer functions set for each neuron. Developed with the proposed approach, ANN is also compared with hand-designed Levenberg-Marquardt and Back Propagation algorithms.

Keywords: Artificial neural network, Backpropagation, Partial swarm optimization, Mean square error, optimization, classification.

ÖZET

MAKİNENİN ÖRENMESİ İLE ŞEKER HASTALIĞINI TAŞHİSİ

Eysa, Alaa Badr

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

Danışman: Asst Prof. Dr. Sefer kurnaz

Tarih: March 2019

Sayfalar: 51

Yapay sinir ağları, özellikle geri bildirime bağlı olduklarında, son on yılda kontrol uygulamalarında karmaşık dinamikler üretme konumunda olmuştur. YSA'lar ağ tasarımı için güçlü olsa da, ağın tasarımı ne kadar zorlaşırsa, istenen dinamik o kadar karmaşık olur. Pek çok araştırmacı, bilgisayar programları kullanarak YSA'nın tasarım sürecini otomatikleştirmeye çalıştı. Arama ve optimizasyon problemleri, bir problemi çözmek için bir ağ için ayarlanmış en iyi parametreyi bulma problemi olarak düşünülebilir. Son zamanlarda, farklı araştırma veri setlerini eğitmek için ANN parametrelerini optimize etme problemi, yaygın olarak kullanılan iki stokastik genetik algoritması (GA) ve partikül sürüsü optimizasyonu (PSO) ile hedeflenmiştir. Sinir ağına dayalı oluşum süreci, robotun karmaşık işleri yapmasını sağlamak için GA ve PSO ile optimize edilmiştir. Bununla birlikte, YSA eğitim sürecini optimize etmek için bu tür optimizasyon algoritmalarını kullanmak her zaman dengeli veya başarılı olamaz. Bu algoritmalar aynı anda bir YSA'nın üç ana bileşenini geliştirmeyi amaçlar: her bir nöron için ayarlanan sinaptik ağırlık, bağlantılar, mimari ve transfer fonksiyonları. Önerilen yaklaşımla geliştirilen YSA, elle tasarlanan Levenberg-Marquardt ve Geri Yayılım ününe sahip algoritmalarla da karşılaştırılır.

Keywords: Artificial neural network, Backpropagation, Partial swarm optimization, Mean square error, optimization, classification.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION

Input, output and hidden neuron layers are the components of ANN. By a set of synaptic weights, neurons are linked to each other. ANN is a powerful tool for pattern detection, forecasting, and troubleshooting. During the learning process, ANN continually changes its synaptic values until the knowledge acquired is sufficient (until there are a number of iterations or an error target). Only samples of the problem outside of those used during the training phase should assess the general capability of ANN once an apprenticeship has been completed. In conclusion, the ANN should classify with reasonable precision the patterns of a particular problem during the training and testing phase. Some classic ANN algorithms have been proposed and developed in recent years. However, many can remain trapped, in other words far from the best or best solution, in unsolicited solutions. Moreover, most of the algorithms do not explore multimodal or non-continuous fields.

Other types of techniques such as bioinspired algorithms (BIAs) are therefore needed for ANN formation. The BIAs are well accepted by the Artificial Intelligence community as they are powerful tools for optimisation that can solve extremely difficult problems with optimization. BIAs can explore large multimodal and non-perpetual areas to find the best way to reach the ideal value. BIAs are based on nature's swarming behaviour. This concept is defined in[1] as the property of non-intelligent but collectively intelligent agents ' systems.

Several works use the use of evolutionary and bioinspired Algorithms for training ANN as another basic form of education [3]. Metaheuristic training methods for neural networks, like coevolutionary co - operative models, are based upon local searches and population methods [3]. The authors show an extensive review of the literature on evolutionary algorithms used to develop ANN is an excellent work [2]. Nevertheless, most of the investigations reported only focus on the development of synaptic grades, parameters or the evolution of neuron numbers for hidden layers, but the designer has previously determined the number of hidden layers. The researchers do not, moreover, involve the development of transfer functions that are an important component of an ANN that determines each neuron's output. For example, in [5] the authors proposed a way to find a specific architecture (connections) for the ANN and particle size optimization (PSO) in order to adjust synaptic weights that combines Ant Colony Optimization (ACO). Other investigations such as [6] conducted an amendment to the SA mixed PSO to get a set of synaptic weights and ANN

thresholds. In [7], the authors use evolutionary programming to obtain the architecture and the weight set with a view to solving problems in classification and prediction. Also, [8] is used to obtain graphs which represent different topologies by Genetic Programming. In [9], an ANN was designed to resolve a weather forecasting problem by using Differential Evolution algorithm (DE). In [10], the authors use PSO algorithms to adjust synaptic weights to shape the relationship between rainfall and runoff in Malaysia. In [11], the authors compare the back-propagation method to the basic PSO only to modify the synaptic weights of an ANN to solve problems with classification. In [12], the weight set developed with the differential evolution and fundamental PSO. Other works, such as [13], simultaneously developed the three main elements of the ANN: architecture, transfer functions and synaptic masses. In [14], the authors solved the same problem by using a differential evolution (DE) algorithm. The authors offered the new PSO model (NMPSO) algorithm. Another example is [15], in which authors develop the design of an ANN with two different fitness functions using the 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 is the introduction of this thesis which make a full review on how this work is possible.
2. Chapter 2: The second chapter is the literature review of references of this thesis which make a full review on how other researchers results and what algorithms they used to classify there dataset.
3. Chapter 3: The third chapter is focusing on the methodology of this thesis. Which also, review a clear idea on the characteristics that these algorithms have and what the possible ways to develop them are. Also the advantages and the disadvantages that both BP and PSO have for the training of ANN.
4. Chapter 4: The forth chapter is focusing on the results of this work.
5. Chapter 5: The last chapter is the results and recommendation which gives general look on the results and summarizing the work according to the results that we had from our experiments on.

1.1 PURPOSIS AND THESIS IMPORTANCE

This study aims to present the new PSO algorithm for optimization, in order to improve ANN's training performance. The functionality takes into account the classification error, square error, validation errors, architecture reduction and the combination. This study also examines the behavior of 3 bio-inspired algorithms of various parameter values. The best parameter values for these algorithms are determined for best results during the experimental phase. Furthermore, for each classification problem in the best configuration a set of statistically valid experiments are generated. The results of the proposed methods in this thesis are also presented and discussed, as well, for each ANN the connection number, the neuron number and the transfer functions selected.

1.2 SWARM RESOURCES

Swarm Intelligence is a study of' collective intelligence' inspired computer systems. The cooperation of numerous homogenous agents in the environment creates collective intelligence. Examples include fish schools, bird herds, and ant colonies. Such intelligence is decentralized, autonomous and distributed in an environment. In the nature of these systems, problems like effective food forage, prey evasion, colonial relocation are frequently solved. The information is typically stored in the participating homogenous agents or stored or transmitted in the environment, such as pheromones in the ants, bee dance and fish and birds nearby. The paradigm consists of two dominant subfields, 1) Ant Colony Optimization, which examines probabilistic algorithms based on ant stigma, and 2) Particular swarming (PSO), which investigates probabilistic flocking, schooling and herding-inspired algorithms. Like evolutionary calculation, the ' algorithms ' or ' strategy, ' which are used in search and optimization, are considered adaptive strategies.

1.2.1 Particle Swarm Optimization (PSO)

In 1995, Russell - Eberhart and James Kennedy developed a Particle Swarm Optimization (PSO), inspired by birds and fish flocculations and scholastics. Initially, the two started development and subsequently realized how well their algorithms worked on optimization problems. The simulations of birds that flowed around food resources were developed. Optimization of Particle Swarm may sound complicated but it's really an easy algorithm. There are several iterations in a

set of variables whose values are more close to the Member and which are closest to the goal at any time. Imagine a flock of birds around an area that can smell a hidden food source. The nearest one to the food is waving in his way and the other birds are swinging. If one of the other circular birds is closer to the goal than the first, the chirp is stronger and the other birds wander towards him. It continues to strain until one of the birds occurs on food. It is an easy and easy to implement algorithm.

The algorithm monitors three variables globally:

- Value or status target
- Global best value (gBest) shows which particle data is currently closest to the target
- Stop value that indicates when the targeted algorithm should stop

Each particle consists of:

- Data that can be a solution
- A speed value that shows how much data can be changed
- A best (pBest) personal value indicating the closest data to the target

Particulate data can be whatever. For each bird's co-ordinate the data in the example above is the X, Y, Z.

Lower-located sub-sets of world best value are often observed with population-topological or neighborhood algorithms. These regions may consist of two or more particles, predetermined for combined action or search space sub-components to test the particles. The use of neighborhoods often helps avoid minimum local requirements.

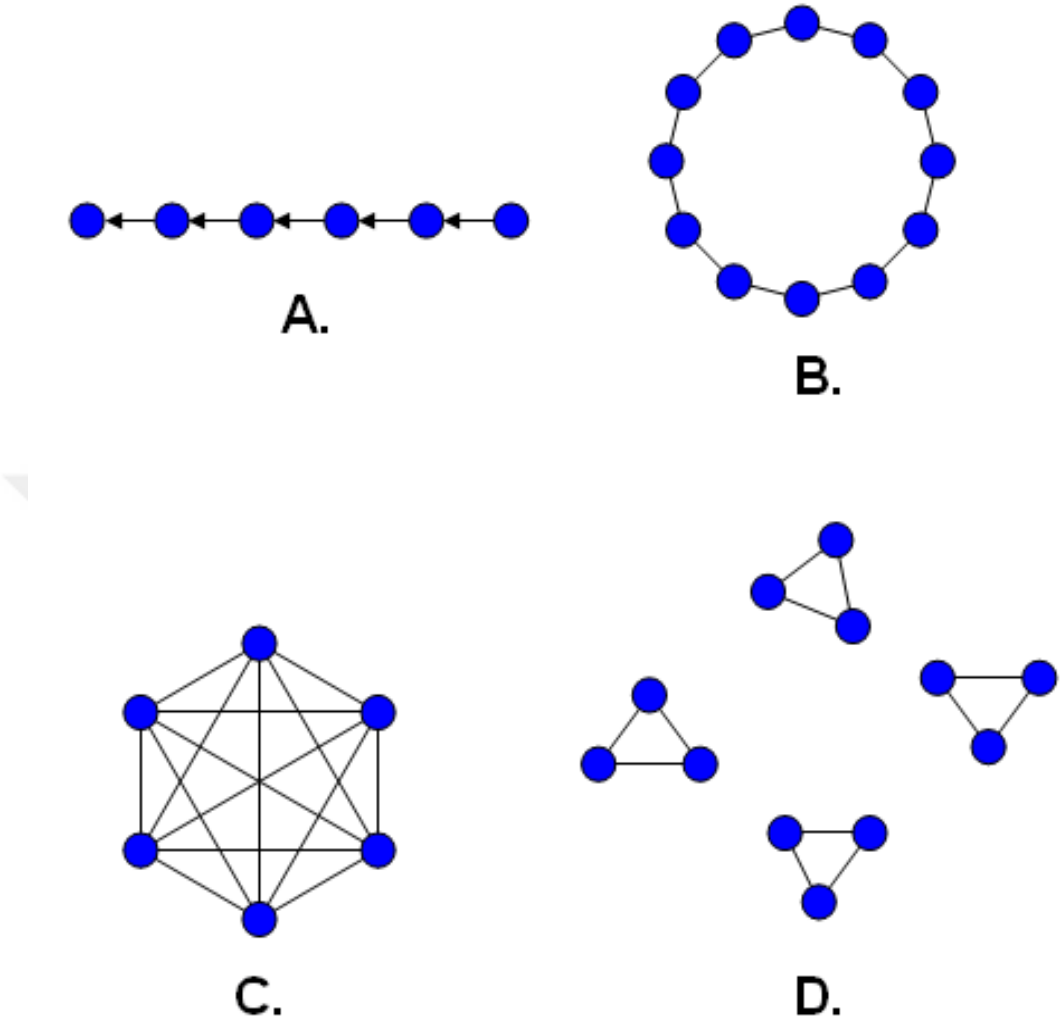

Figure 1.1: Population topologies

Some neighborhood population based topologies. (a) Single, with only the next best comparable. (b) Ring topology in which individuals can only compare those on the left and right. (c) Completely related topology, where all of them are compared. (d) Isolated, where only those in specific groups are compared.

The definitions of the neighborhood and how they are used affect the behavior of the algorithm differently. In diagram followed by pseudo code, the basic PSO algorithm is explored:

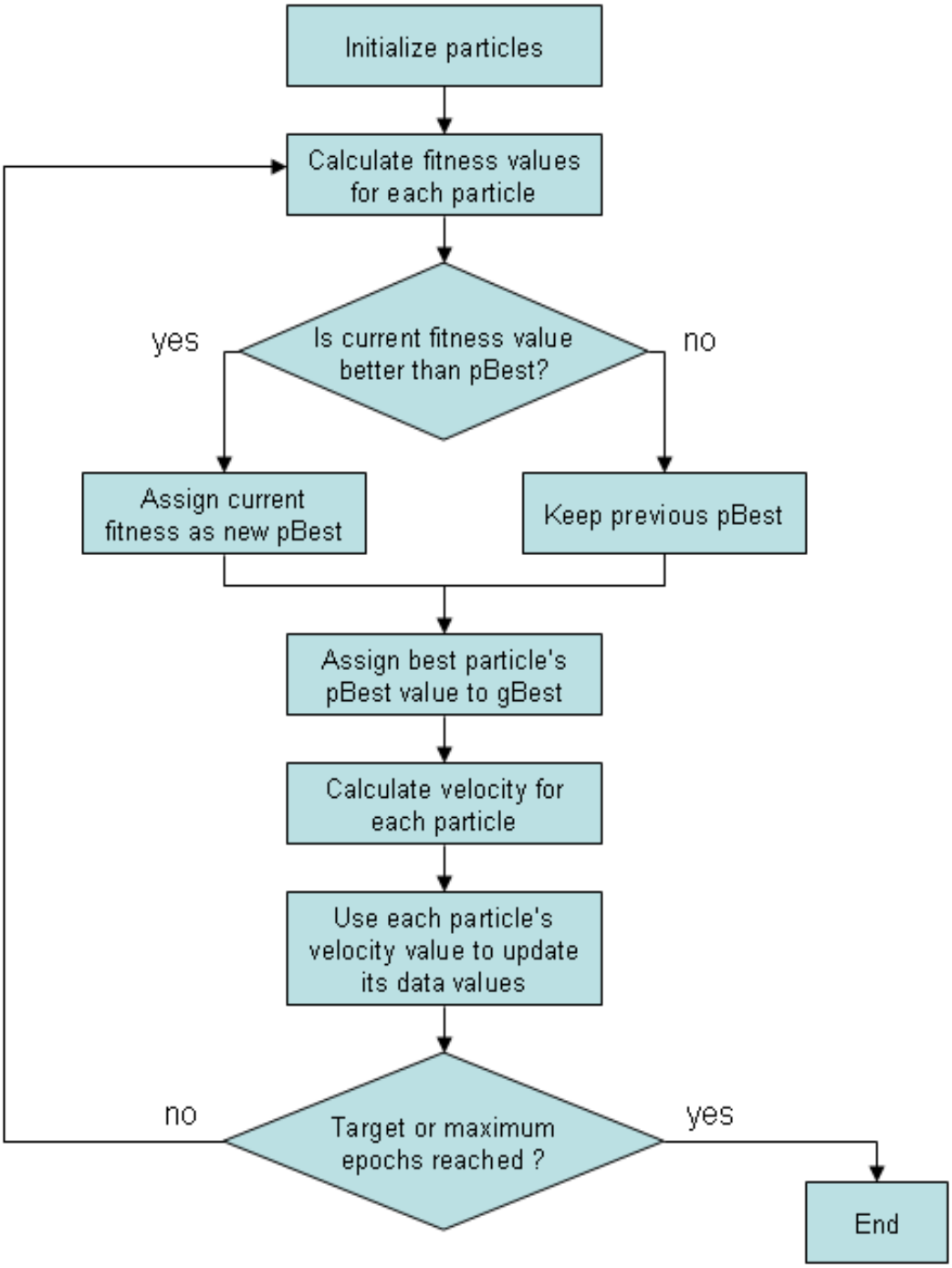

Figure 1.2: Flow chart showing the algorithm for optimizing particle swarm

2. LITERETURE REVIEW

Artificial Neural Networks are a system made up of input, output and hidden layers of neurons. A set of synaptic weights connects the neurons to each other. An ANN is a powerful tool used to detect patterns, predict and regression in a variety of problems. The ANN will constantly change synaptic values during the course of the learning process until the learning knowledge is sufficient (until the objective error value is reached). If the training or stage is over, then a sample problem that is different from those used in the workout phase must be used to evaluate the capacity for generalizing ANN. Finally, the ANN is expected to classify the patterns of a specific problem in the training and testing phase with an acceptable precision.

In recent years, several classic ANN algorithms have been suggested and developed. Many of them, however, can remain trapped in unwanted solutions; that is, far from being the best or the best solution. In addition, the majority of these algorithms are unable to explore multimodal and noncontinuous areas. Therefore, other types of techniques are necessary for ANN training, including the bio-inspired algorithms (BIAs).

As BIAs are powerful optimization tools and can solve very complex optimization problems, they have a good acceptance by the Artificial Intellectual Community. BIAs can explore large, multimodal and non-stop search areas for a given problem and are able to come up with the best solutions close to the best value. BIAs are described as swarm intelligence by nature. BIAs are based. This concept is defined by [1] as the property of unintelligent systems with limited capacity but intelligent collective behavior.

Several works are used to form ANN as a fundamental form of education with both evolutionary and bio - inspired algorithms [2]. Metaheuristic methods for neural network education are based on local search, population methods and others, such as coevolutionary cooperative models [3].

The author shows a thorough literature review of evolutionary algorithms used to develop ANN [2]. This is an outstanding piece of research. The majority of the investigations report, however, focus only on synaptic weight developments, parameters [4], or the development of the neuron numbers for hidden layers, but the designer previously identified the number of hidden layers. In

addition, research does not involve the evolution of transmission functions which are an important part of an ANN that determines each neuron's output.

For example, the writers suggested a method combining ANN and PSO for the purpose of synaptic Weight Adjustment to combine Ant Colony Optimization (ACO) to find a specific architecture (connections). Other research such as [6] has introduced a P Simulated Annealed (SA) PSO modification for a set of synaptic weights and ANN thresholds. 6] In order to solve problems with classification and prediction, the author uses evolutionary programming to get the architecture and weight set. Also in [8] genetic programming is used to obtain graphs representing different topologies. In [9], a weather forecast issue was solved by the Differential Evolution (DE) algorithm for designing an ANN. In [10] authors use the PSO algorithm to change the weight of the synaptics to shape Malaysia's daily rains-runoffs. In [11] authorists only adjust the synaptic weight of an ANN for solving classification problems, comparing back-propagation versus basic PSO. The weight set is developed with the Differential Evolution and the basic PSO in [12].

Other works such as [13] have also developed the three main elements of an ANN: architecture, functional transfer, and synaptic weights. A New Model PSO (NMPSO) algorithm has been proposed by the authors and a Differential Evolution (DE) algorithm in [14] solves the same problem. Another example of this is [15] in which the authors have developed an ANN with two distinct fitness functions using an Artificial Bee Colony (ABC) algorithm.

In comparison with these last three works, this research has made significant contributions. First, eight fitness functions are proposed to address three common issues which arise during the ANN design: precision, over-size and ANN reduction. In this sense, the fitness functions take the classification error into account, mean the square error, validation error, reduction of architectures and a combination of them into account to handle problems that arise during the design of ANN. In addition, this research investigates the behaviour, with different settings, of three bio-inspired algorithms. The best parameter values for these algorithms are established during the experimentation phase to produce the best results. In addition, the best configuration for each classification problem is used to generate a number of statistically valid experiments. In addition, the results achieved with the proposed methodology are presented and discussed with regard to the connection number, the neuron number and the transfer functions chosen for each ANN. Another contribution from this work is linked to a new metric, which enables the results of an ANN

produced with the proposed methodology to be effectively compared. This measures the detection rate during the training and test phases in which the accuracy of the test is more measured in relation to the accuracy of the training. The results of the three bio-inspired algorithms are finally compared to those achieved with two classic learning algorithms. The three bio-inspired algorithms have been selected because NMPSO has a relatively new algorithm (submitted in 2009), based on the metaphor of the essential PSO method, so comparisons with others that inspire the same phenomenon are important.

The problem to be resolved can generally be defined as a series of input patterns, and as a set of desired patterns, so as to find the ANN represented by the minimization of a function defined by the maximum number of neurons. It is important to note that three areas (architecture, synaptic weight, and transfer functions) cover the search space.

The research is a thorough examination of the manner in which the application of bio-inspired algorithms can be automatically designed by an ANN, in particular through Basic Particle Swarm Optimization (PSO), Second General PSO (SGPSO), and New PSO Model (NMPSO). This methodology is developed to design the ANNs that are most accurate for a certain problem while developing the architecture, the synaptic weight and the type of transmission functions. Furthermore, a comparison with traditional study techniques (back-propagation and Levenberg Marquardt) of the Particle Swarm algorithm performance is presented. Furthermore, a new way to select the maximum neuron number (MNN) is presented in this research. The accuracy of the methodology proposed has been tested to solve some real and synthetic patterns. In this paper, we show the results with ten different complexity classification problems.

3. METHODOLOGY

3.1 DATA RESOURCE

There are several limitations in the database to use a larger database to select such cases. All Pima women aged 21 and over are patients in this case. ADAP is an adaptive learning routine that enables digital analog devices such as perception to be generated and operated. This is a specific algorithm. For details see the paper. The dataset is provided by the Diabetes and Digestive and Renal Disease National Institute. The objective is to forecast whether a patient is diagnosed with diabetes.

Vincent Sigillito's 768 DADB is a data database from Pima Indian diabetes which uses a customizing learning routine for phenomenon digital analogies-such as ADAP applications. Persons living near Phoenix are provided with a database of diagnostic reports. 5 76 per cent of the other 192 cases were classified and 576 instances of training were used.

This data set serves as an illustration of the impact of topology on the generalization capability of the proposed network. The Dataset's trial results with 12fold cross validation and special pre-processing are 77.7 percent using the LogDisk algorithm.

The optimal topology (8-5-5-5-14-30-5-6) is shown to provide 76.95% of the rating accuracy. The bin number optimization process can also be observed as additive. So with 5 and six attributes, the accuracy results in 75 percent, which is approximately 2.5% above that produced separately, with a total of 14 and 30 containers each. In order to create the best network possible for a single task, the optimisation of network topology can be automated on a virtual machine in parallel. Since the Bayese naïve networks are also suitable-well-for high performance parallel architecture with parallel calculation of the attribute values. The possibility of network implementation in a Parallel Virtual Machine (PVM) is explored in future investigation.

Information of Pima Indians Diabetes Database are described below:

- Title: Pima Indians Diabetes Database
- Sources:

        - Original owners: National Institute of Diabetes and Digestive and Kidney Diseases
    - Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu)
        - Research Center, RMI Group Leader
        - Applied Physics Laboratory
        - The Johns Hopkins University
        - Johns Hopkins Road
        - Laurel, MD 20707
        - (301) 953-6231
    - Date received: 9 May 1990
- Relevant Information:

  There were various restrictions on the selection of these instances in a larger database. All patients here are women with the heritage of Pima, who are 21 years old or older. ADAP is an adaptive learning routine that produces and runs digital analogs similar to perceptons. This is a particular algorithm; see the paper for details.

- Number of Instances: 768
- Number of Attributes: 8 plus class
- For Each Attribute: (all numeric-valued)
    - ❖ Number of times pregnant
    - ❖ Plasma glucose concentration a 2 hours in an oral glucose tolerance test
    - ❖ Diastolic blood pressure (mm Hg)
    - ❖ Triceps skin fold thickness (mm)
    - ❖ 2-Hour serum insulin (mu U/ml)
    - ❖ Body mass index (weight in kg/(height in m)^2)
    - ❖ Diabetes pedigree function
    - ❖ Age (years)
    - ❖ Class variable (0 or 1)
- Missing Attribute Values: Yes
- Class Distribution: (class value 1 is interpreted as "tested positive for
- diabetes")

- Class Value Number of instances
    - 500
    - 268
- Brief statistical analysis:

Table 3.1: Statistical Analysis of Dataset

Attribute Number	**Mean**	**Standard Deviation**
1.	3.8	3.4
2.	120.9	32.0
3.	69.1	19.4
4.	20.5	16.0
5.	79.8	115.2
6.	32.0	7.9
7.	0.5	0.3
8.	33.2	11.8

3.2 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 makes it possible to develop solutions less technically to reduce human participation.

The neural network implementation in computing provides us with a key computing advance. Computers do well, such as complex mathematics and face recognition. 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, a certain type of cell that does not appear to regenerate, unlike the whole body, is the most important element in the human brain. Since this type of cell is the only part of the body that does not slowly be replaced, 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 unfit for human brain. "These artificial neural networks can only replace the most basic elements of this complex and powerful organ. Neural computing never replaced the developer that is trying to resolve problems with his human brains. "Only artificial neural networks can replace the most basic elements of this complex and strong organ. The development company, who is attempting to resolve problems, has never substituted neural computing for human brain. It's machinery and a new way to resolve problems."

Let us look at human neurons overview. Neuron is the fundamental element of the neural network. The biological neurons are input, then they are subjected to different non-slight operations and then produce final output. There are four typical nerve cells: dendrites, somas, axons, 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 are not too simple, but more complex in structure.

Biology improves neuronal understanding. 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 allows better fit to the data
2. Accurate prediction in case of uncertain data and measurement errors is possible due to presence of noise insensitivity.
3. High parallelism implies hardware failure-tolerance and fast processing.
4. Learning and adaptively allow the system to update its internal architecture in response to changing environment.
5. Application of the model to unlearned data is possible due to generalization.

ANN-based computing mainly seeks to develop mathematical algorithms that allow artificial neural networks to learn through imitation of information processing and the acquisition of human brain-like knowledge. ANN-based patterns may provide virtually exact solutions to specifically formulated issues and processes that are only understood by experimental and field observations. "ANNs have been used in microbiology in a number of applications, from modeling, classification, pattern recognition and multivariate data analysis." Digital image processing is one of the newly developed ANN applications. Digital image processing interests stem from two main applications: improvement of human interpretation image data; and processing of image data for autonomic machine sensing storage, transmission and representation. The two dimensional function can be defined as a picture, f(x, y) in which x and y are spacial coordinates, and the intensity or grey level of the image at that point is off at any coordinate pair (x, y). The amplitude values, called as a digital image, are all finite, discreet amounts.

“Digital image processing refers to digital image processing through a digital computer. A digital image consists of a limited number of elements with a specific position and value. These elements are known as image elements, picture elements, skins and pixels. Digital image processing applications are broad and diverse”.

3.2.1 General Properties of Artificial Neural Networks

1. **Nonlinear I/O mapping**

   Due to the increase of sensor resolution and computer memory capacity, the analysis of high dimensional data becomes increasingly important. Images, spoken signals and multisensor information are common examples. When analyzing these signals, all or part of them is shown in a sample area. A single data point, for example, can constitute a 16x16 (sub) image. An image collection forms the cloud of points in a space of 256 dimensions. There is a great dependence between sensors and sensor elements in most multi-sensor data sets. For nearby image pixels this is certainly true. But more fundamentally, any physical experiment should not contain hundreds of degrees of freedom that are important. Therefore, lower dimensional descriptions may represent multivariate data sets. There are different reasons which could be of interest to such representations. You can reveal the structure of the data or the problem, use them to relate each data point to each other (e.g. to find the most similar in a database) or retrieve missing data.

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 by which information or significance is transferred from one particular topic–analogy or source, to a different target or linguistic expression that corresponds to that process. In a narrower sense, in comparison with the deduction, induction, and abduction analogy is inference or argument from one particular to another where one premise or the conclusion is at least general. The word analogy could also refer to the relation between the source and the target itself, which, as in the notion of biology, is often but not necessarily similar. The atom model of Rutherford analogized the atom with the solar system. In problem solving, analogy plays a major role, including decisions, argument, perceptions, generality, memory, creativity, invention, forecasting, emotions, explanation, conceptualization and communication. It lies behind basic functions, for example in face perception and face recognition systems, such as the identification of places, objects and people; Analogy was argued to be "the nucleus of cognition." Specific analogical language includes, but not metonymous, exemplification, comparisons, metaphors, like, allegories and parables. 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. The concepts of association, comparative approach, correspondence, mathematical and morphological homology, homomorphism and iconicity are closely connected with analogy. 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.

3.2.2 Structure of ANN

The neural network consists of three group, or layers or phases. The input phase, hidden phase and output phase.

- The activity or input unit represents the raw data that is provided to the network.
- 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. The weights of the hidden and input units decide when every hidden unit is active, so that a hidden unit can choose what it represents by adjusting these weights. 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.

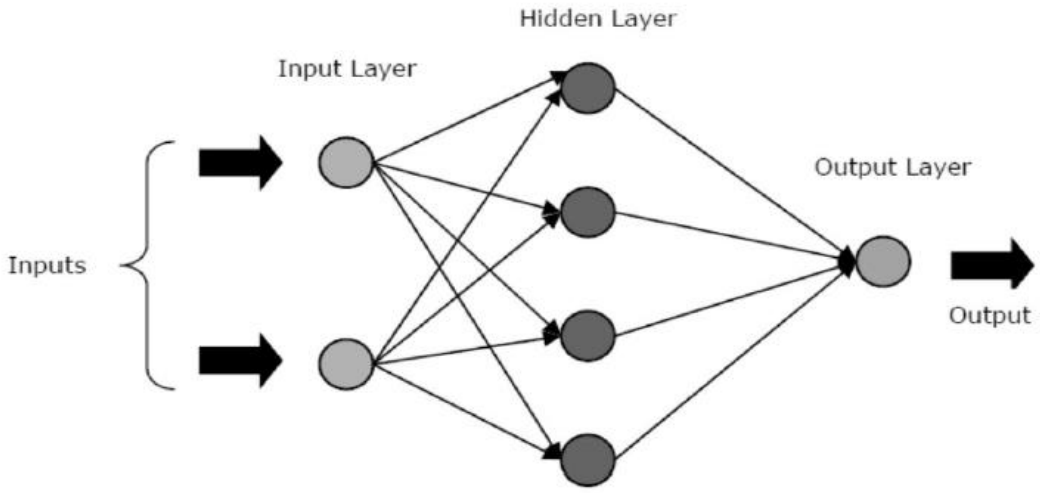

Figure 3.3: General neural network architecture

3.2.3 Elements of Artificial Neural Networks

3.2.3.1 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.

3.2.3.2 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.

3.2.3.3 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.

3.2.3.4 Activation Function

Also referred to as activation functions is the threshold or transfer feature. In order to transform activation levels of neurons into output signals, the activating functions are used. In the neural network, there are numerous activation functions. Identity function, step function, piece-wise 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 $B_1, B_2, \ldots\ldots, B_k$ are the coefficients in the regression equation. As a result, it is uncommon to find a neural network with identity activation used in all its perceptron's.

b. Sigmoid activation function:

Nonlinearity in the model is used in the artificial neural network sigmoid functions. A network neuroelement calculates the output of a linear combination of its input signals by applying a sigmoid function. The sigmoid function fulfills a property between the product and itself, making it easier to perform computationally, making it more popular in neural network.

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{3.1}$$

Sigmoid function derivatives are generally used in learning algorithms. The Sigmoid graph is shaped as' S'. This feature is defined as a function that expands and is used frequently in the development of a neural artificial network. Sigmoid is defined as a function that increases strictly, and shows a balance between linear and nonlinear functions.

One - polar – is the sigmoid function.

c. Step function:

This is a unipolar and also known as a threshold.

$$\varphi(v) = \begin{Bmatrix} 1\ if\ v\ \geq 0 \\ 0\ if\ v > 0 \end{Bmatrix} \tag{3.2}$$

The output of neuron K using a threshold function is

$$y(k) = \begin{Bmatrix} 1\ if\ v_k\ \geq 0 \\ 0\ if\ v_k\ > 0 \end{Bmatrix} \tag{3.3}$$

$v_k\ is\ the\ induced\ local\ field\ of\ the\ neuron$

$$v_k = \sum_{j=1}^{m} W_{kj}\, X_j + \ b_k \tag{3.4}$$

If the neuron output is 1 if the induced local neuron field is non - negative, it takes 0 in this model, otherwise.

d. Piece Wise Linear Function

It can be defined as a unipolar function

$$\varphi(v) = \begin{Bmatrix} 1\ , & v \geq +1/2 \\ v\ , +\frac{1}{2} > & v > -1/2 \\ 0\ , & v \leq -1/2 \end{Bmatrix} \tag{3.5}$$

Where the amplification factor is assumed to be within the linear operating area

1. The specific circumstances of linear functions are
    - A linear combiner is produced if the linear operating region is maintained without saturation.
    - 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 kinds of learning rules in the neural network and they are usually divided into three categories.

- Supervised Learning
- Un Supervised Learning

a. Supervised Learning

Training sets are available for supervised learning. A set of examples with proper network behaviour are provided for this type of rule. The inputs and expected outputs are given as a training set in supervised learning. 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.

$$\{x1, d1\}, \{x2, d2\}, \dots \dots \dots, \{xn, dn\} \quad (3.6)$$

The network input is xn in this case and dn is the required destination input. The output is generated by input. The study rule is used to change the biais and weights of a network in order to make the network outputs more accurate.

In supervised learning, we commit ourselves to providing the system with the desired response (d) every time the entry is applied. As an error measure to externally correct the network parameter, the distance between real and desired response is used. 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

Self-organized learning is also known in unexpected learning. In uncontrolled learning, target output is not available. In this case, weights and biais are altered only by network input. Unattended study grouping is used for pattern reorganization. In unattended learning the answer required is not known, 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. The network needs to detect any existing patterns, properties, regularities, etc. while discovering its parameters in this learning method. Unattended study means learning without the teacher because the teacher does not have to take part in all training steps, but the teacher must set goals. It is also important to have feedback on neural networks. Feedback is called progressive learning, which is very important for uncontrolled learning.

3.2.3.5 Outputs

This layer also consists of neurons, thus producing and displaying the final network outputs resulting from neuron processing at the preceding layers. Considering how the neuron disposition is connected and how its layers are composed, the main architectures of artificial neural networks can be divided accordingly. The following are possible:

(i) Single-layer feedforward network,

(ii) Multilayer feedforward networks,

(iii) Recurrent networks and

(iv) Mesh networks.

3.2.4 Classification of Artificial Neural Networks

3.2.4.1 Artificial Neural Networks According to Constructions

A. Single Layer Feed Forward Network

A neural network where the source node is projected into the neural output layer, but not the opposite is called a single feed or acyclic network. Single layer refers to the calculation node output layer in a single network layer as shown in Fig. 3.2.

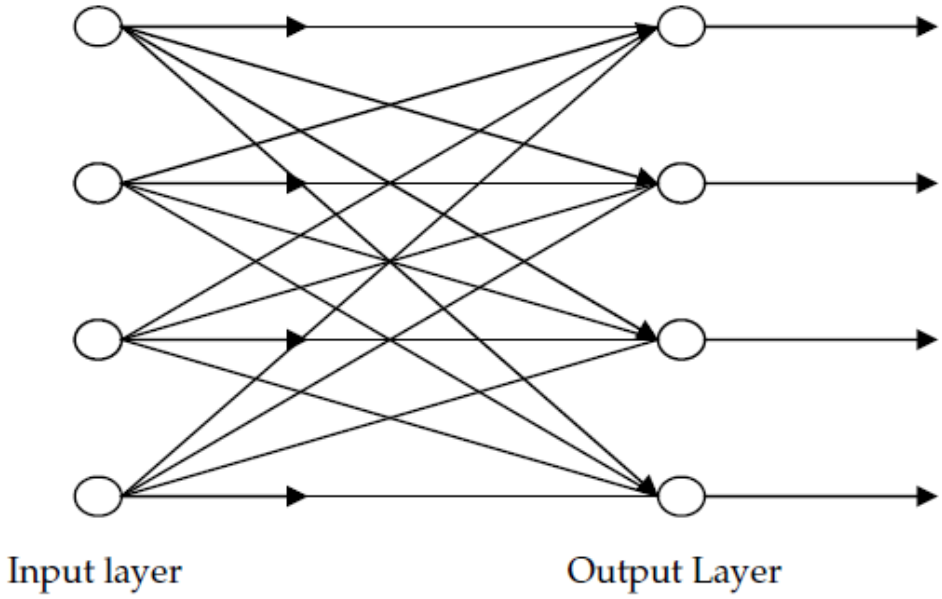

Figure 3.4: Single Layer Feed Forward Network

B. Multilayer Feed Forward Network

The system comprises of at least one concealed layers, which are called shrouded neurons or shrouded unit computational hubs. The capacity of shrouded neurons is to collaborate viably between the outside info and system yield and to separate insights in higher request. The source hubs in the system input layer furnish neurons in the second layer with the info flag. The second layer yield signals are used as the third layer inputs, and so forth. Sets of neuron yield motions in the system yield layer are the general system reaction to the actuation design given by source hubs in the primary layer of the info.

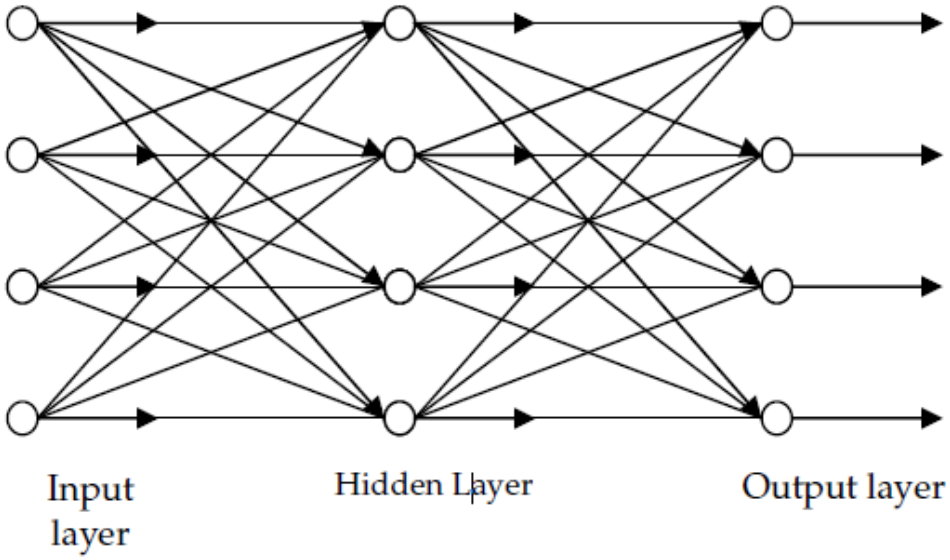

Figure 3.5: Multilayer Feed Forward Network

Short characterization of feedforward networks:

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.

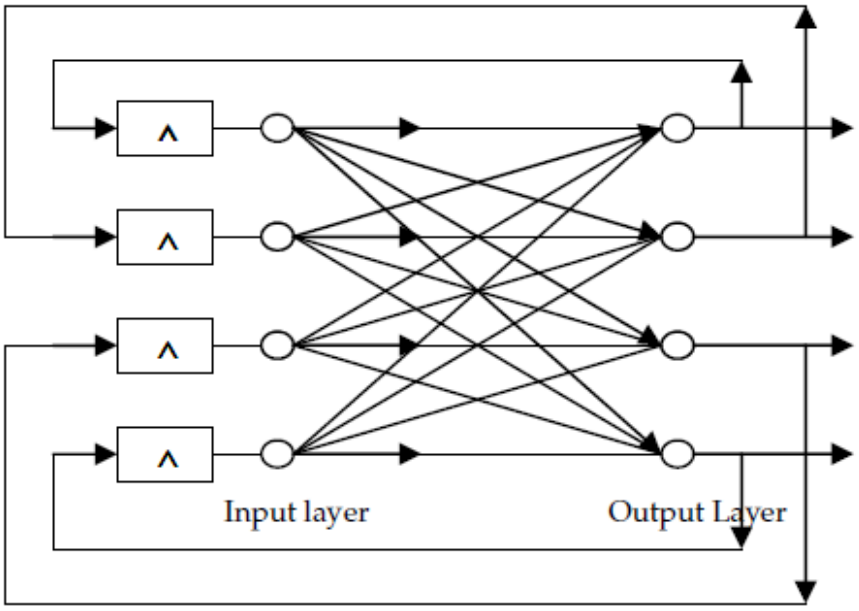

Figure 3.6: Recurrent Network

Other network types are available ; delta - bar - delta, hopfield, vector quantization, counterpropagation, probabilistic, hamming, boltzman, associative bidirectional memory, spatio - 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.

3.2.4.2 Artificial Neural Networks According to Learning Algorithms

When a network for a specific application is structured, it is ready for training. The initial weights are randomly selected at the beginning and then training or learning starts. There are two training approaches; controlled and unattended.

A. Supervised Training

The inputs and outputs are provided in supervised training. The network processes the entries and compares the results to the desired results. The system then spreads errors, so that weights that control the network are adjusted by the system. This process happens time and again when the 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."

A network can sometimes never get to know it. This might happen because there is no specific information in the input data that produces the required output. Networks are also not converging if not enough data is available to enable full 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.

The designer must then examine input and output, number of layers, number of elements per layer, layer connections, transfer and training functions and even the initial weights if the network cannot just resolve the problem. 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.

If the system is finally properly trained and there is no need for further training, the weights can be frozen if requested. This finalized network can then easily be converted to hardware on certain systems. 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. This information is incorporated into the topology and study rules of the network. 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. Also, if external input to cluster nodes has been decreased, this may inhibit the whole cluster. Competition among processing elements could also provide a learning basis. Competitive cluster training could amplify specific incentives in the response of certain groups. 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: Donald Hebb introduced this in the 'Compatibility Organization. The basic rule is: When the neuron comes from another neuron and if the two neurons are both very active (the same sign) the weight should be strengthened.

Hopfield Law: Increase the weight of connections by the rate of study and, if the desired outputs and the inputs are both active and inactive.

The Delta Rule: The idea is that the input connections ' strengths can constantly be changed to reduce (delta) the difference between the desired output value and the actual output of the working element.

The Gradient Descent Rule: This is similar to the Delta rule, as the transfer function derivative is still used to modify the delta bug before connection weights are applied. 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 PARTICLE SWARM OPTIMIZATION

Three different PSO - based algorithms are described in this section. The first is the PSO algorithm. Two algorithms are then shown that improve the original PSO: the second generation of PSO and a new PSO model.

A. Original Particle Swarm Optimization Algorithm

Eberhart et al's proposed approach to the optimization of continuous nonlinear functions is the particulate swarm optimisation (PSO). It is inspired by social and collective observations on bird movements to search for food, survival and fish education. This algorithm is based on The movement and at the same time the experience of the best members of the population are a PSO algorithm. The metaphor indicates that a number of alternatives move in a search area in order to obtain the best position or solution.

In a multidimensional field, a population is considered a cumulus of particles I, each representing position X I to R^d, i=1, ..., M. These particles are evaluated with a particular optimization function to recognize their fitness and save the best solution. The V y velocity function takes into consideration both the best P g / R^D (i.e., social component) position of the particles in a population and the best P g to / R^D position (i.e., cognitive component), according to all participating particles. The particles are moved in each iteration to a different position until they reach their optimal position. The speed of the particle I is updated with each time t.

$$V_i(t+1) = \omega v_{i(t)} + c_1\left(\mathrm{p}_i(t) - X_i(t)\right) + c_2 r_2\left(\mathrm{p}_g(t) - X_i(t)\right) \qquad (3.7)$$

where ω is the inertia weight and typically set up to vary linearly from 1 to 0 during the course of an iteration run; The acceleration coefficients are c (1 and) c 2; the r 1 and r 2 numbers have been distributed uniformly between (0,1). The velocity of v I is [v (max,)v min] restricted. This allows the particle I to search for the best individual p I (t) location and the best global particle position I can be calculated as in

$$X_i\,(t+1) = X_i(t) + V_i\,(t+1). \qquad (3.8)$$

B. Second Generation of PSO Algorithm

The SGPSO algorithm improves the original PSO algorithm, taking account of three aspects: the optimum local solution for each component, a new concept and the best geometric swarm centre. The authors explain that somewhere away from the birds is the swarm center (food). But the swarm center is not accurately calculated by any bird. The bird flow remains in the area where the swarming center is fixed in every bird's eyes for a specified time. Then the swarm enters a new area. All birds must then keep a certain distance in the new swarm centre. This is the basis of the SGPSO.

The geometrical center position (P) at R^D of the optimal swarm is updated to

$$\bar{P} = \frac{1}{M}\sum_{1=1}^{M} P_i, \text{ if } CI \bmod T = 0, \qquad (3.9)$$

where M is the number of particles in the swarm, CI is the current iteration number, and T is the geometric centre updating time of optimum swarm with a value between $[1, MAXITER]$. SGPSO updates the speed of each particle by(4) and the position by(5):

$$\begin{aligned} \mathrm{v}_i(t+1) = \omega \mathrm{v}_i(t) + c_1 r_1\big(\mathrm{p}_i(t) - X_i(t)\big) \\ + c_2 r_2\left(P_{\mathrm{g}}\,(t) - X_i(t)\right) \qquad (3.10) \\ + c_3 r_3\big(\bar{P} - X_i(t)\big), \end{aligned}$$

$$X_i(t+1) = X_i(t) + \mathrm{v}_i(t+1), \qquad (3.11)$$

If c (1,)r 1 and c 3 are the accelerated constant, the r (1)r 2 and r 3 are random numbers within [0,1] and the iv is the inertia of the speed of the acceleration constant.

C. New Model of Particle Swarm Optimization

This algorithm is based on the ideas of other authors to improve the basic PSO algorithm. Garro and others. Garro and others. The following paragraphs describe these ideas. In generations, Shi and Eberhart[18] proposed a linear change in inertia weights, with a significant improvement in Basic PSO performance.

$$\mathrm{w} = (\mathrm{w_1} - \mathrm{w_2}) \times \frac{MAXITER - iter}{MAXITER} + w_2, \qquad (3.12)$$

If w 1 and w 2 are the initial and final weight of the inertia, they are present iter and MAXITER is the maximum acceptable number of iterations. Empirical studies in this study showed that the value could be increased from 0.9 to 0.4 end of the process of evolution. Yu et al. have developed a strategy to select each particle I by the population with a predefined probability, and then random disruption to each vector dimension V I of the selected particle I if the global best position does not improve as more people are generated. The speed reset is calculated as in

$$V_i = V_i + (2 \times r - 1) \times v_{max}, \qquad (3.13)$$

When r is a uniformly distributed random number (0, 1) and v max, each selected particle size has the maximum random perturbation magnitude.

Several effective PSO mutations and crossover operators were proposed based on certain evolutionary systems of genetic algorithms (GA). In terms of a crossover rate α, defined in Løvberg et al. [19] proposed a crossover operator

$$ch_1(X_i) = r_i\, par_1(X_i) + (1 - r_i) par_2(X_i), \qquad (3.14)$$

Where r I is a random number of the same range (0.1), the upper part is the offspring, while the second is the parent randomly selected from the population, the second is the child. The second is the child.

The descending speed is calculated as the sum of the two parent vectors, normalized for each parent vector's original length in the next equation:

$$ch_1(V_i) = \frac{par_1\,(V_1) + par_2\,(V_i)}{|par_1(V_i) + par_2(V_i)|}|par_1(V_i)|. \qquad (3.15)$$

Higashi and Iba suggested a Gaussian mutation operator to improve PSO performance for a certain β-mutation rate

$$ch\,(X_i) = par\,(X_i) + \frac{MAXITER - iter}{MAXITER} N\,(0,1), \qquad (3.16)$$

Where ch is the selected parent of the population, the current iteration number is that of the parent and MAXITER is the maximum iterations, whereas N is the Gaussian parent. The use of these PSO operators can lead to quicker convergence and better solutions. In conjunction with dynamic operators, Mohais et al. [6, 21] used PSO random neighborhoods.

Within the NMPSO it is suggested that dynamic random neighborhoods are used to change certain μ rates. First of all, MAXNEIGH's maximum number of districts is defined by population size divided by 4. At least every K&n, n=1,....,MAXNEIGH district has at least 4 members under this condition. The members of each K n district are selected randomly then, and the best p (gk n) particle is calculated. The speed of each particle I is finally updated as at

$$v_i(t+1) = \omega \boldsymbol{v}_i(t) + c_1 r_1\big(\boldsymbol{P}_i(t) - \boldsymbol{X}_i(t)\big) + c_2 r_2\left(\boldsymbol{p}_{gk_n}(t) - \boldsymbol{X}(t)\right) \qquad (3.17)$$

For all $i \in K_{n,} n = 1, \ldots, MAXNEIGH$.

The NMPSO unites different inertia weight and speed resetting schemes, crossover and mutation operators and random acceleration neighborhoods (C 1 and C 1. The NMPSO algorithm is specified in algorithm 1.

Algorithm 1: New Model of PSO pseudocode.

(1) Given a population of $\boldsymbol{X}_i \in \mathbb{R}^D, i = 1, \ldots, M$ individuals.

(2) Initialize the population at random.

(3) Until a stop criteria is reached:

(4) ***if*** $\boldsymbol{P}_g$ is not improved during $NITER$ **then**

(5) Perform a velocity resting, (7).

(6) **end if**

(7) ***if*** **r**(0,1) $> \gamma$ ***then***

(8) Create new neighbourhoods.

(9) **end if**

(10) Modify inertia weight, (6).

(11) **for** each individual $\boldsymbol{X}_i$ **do**

(12) Evaluates their fitness**.**

(13) **end for**

(14) **for** each individual ***i*** **do**

(15) Update its best position $\boldsymbol{p}_i$

(16) **end for**

(17) **for** each neighbourhood k **do**

(18) Update the best individual $\boldsymbol{p}_{gk}$.

(19) **end for**

(20) **for** each individual i and each dimension**: do**

(21) Compute the velocity update equation $V_i\ (t+1)$, (11)

(22) Compute the current position $\boldsymbol{X}_i(t+1)$.

(23) **if** $\gamma(0,1) > \alpha$ ***then***

(24) Apply crossover operator, (8) and (9).

(25) **end if**

(26) **if** $\gamma\ (0,1) > \alpha$ **then**

(27) Apply mutation operator, (10).

(28) **end if**

(29) **end for**

3.3.1 Particle Swarm Optimization Parameter Check

Sensitivity, septicity and precision are commonly preferred statistics in determining the performance of a classification system. The sensitivity is the estimated data rate of epileptic patients, the specificity is the estimated data rate of healthy people, and the correctness is the true classifying rate. Eq. Equality. The calculation of these statistical numbers is based on (36), (37) and (38).

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.18}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.19}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.20}$$

In the above equations, TP is the total amount of epileptics diagnosed; TN (true negative) represents the total number of healthy patients who are diagnosed regularly; FP is the total number of diagnosed healthy epileptics; FN is the total number of normal epileptics diagnosed; the total number of normal epileptic patients is also diagnosed with FN.

3.4 TRAINING OF ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANN) are systems inspired by neural biological systems which form animals ' brains. Such systems learn tasks (progressively improve their performance), usually without task-specific programming, by considering examples. For example, by analyzing examples that were manually labeled as "cat" or "not cat," and by using the data to identify cats in other pictures, they may be able to identify pictures that contain cats in image recognition. For example, you do this with fur, tails, whiskers and catlike faces without a previous knowledge of cats. Rather, they develop their own set of features through the educational material they use. An ANN is a collection of connected units or nodes based on an artificial neuron (analogous to the biological neurons of an animal brain). Each connection between artificial neurons can transmit a signal from one to the next (analogous to a synapse). The artificial neuron receiving this signal can process it and signal the artificial neurons connected to it.

In the common application of ANN, the signal is an actual number to connect the artificial neurons and the output of every artificial neuron is calculated with a non-linear function of the total of their inputs. In general, the weight of artificial neurons and connections varies with learning. The signal power is increased or reduced at the connection. The threshold of an artificial neuron can only be the signal sent if the total signal passes through that threshold. The layers are generally arranged for artificial neurons. Different layers can transform inputs into different types. After layer crossing signals pass from the first (input), the last (output), and possibly more.

The ANN approach was designed to solve problems in the same manner as a human brain. Over time, attention was directed towards the adaptation of certain mental skills to lead to biological deviations. The ANNs have been used for many tasks such as computer vision, speech acknowledgement, machine translation, social network filtering, playboard games and medical diagnoses. The training process using PSO and AAA algorithms is presented in this section to optimize ANN performance.

3.4.1 Training Artificial Neural Networks with Particle Swarm Optimization.

This is one of the population-based heuristic optimization methods based on social behavior in birdflowers or fish schools while finding food. PSO, which first was developed by Kennedy &

Eberhart in 1995. The PSO algorithm is initialized and the optimum solution is then sought by a group of random particles (candidate solution for the problem). Each particle is updated on two special particles in each generation: For every particle found to date, pbest is the best personalized solution and gbest is the best solution ever found in any swarm particle. Figure 3.1 displays the vector representation updating process of a particle.

The algorithm's pseudo code is the following:

Algorithm 1: ANN using PSO

For each particle do

Initialize the particle with random values

End for

Do

For each particle do

Calculate fitness value of the particle

If fitness value of the current particle < fitness value of the pbest particle then

update the pbest particle

End if

End for

gbest = the particle whose fitness value is equal to min(fitness values of all particles)

For each particle do

update velocity and position of the current particle

End for

While stop

Criteria are provided (maximum generation number or the gbest particle target fitness value). The variables v ij^k andx ij^k in Figure 3.1 are the velocity components jth (J= 1, 2,..., D) of the ith component I= 1, 2, 3,..., N) at k. N is the amount of swarm particles. D is the search space dimension.

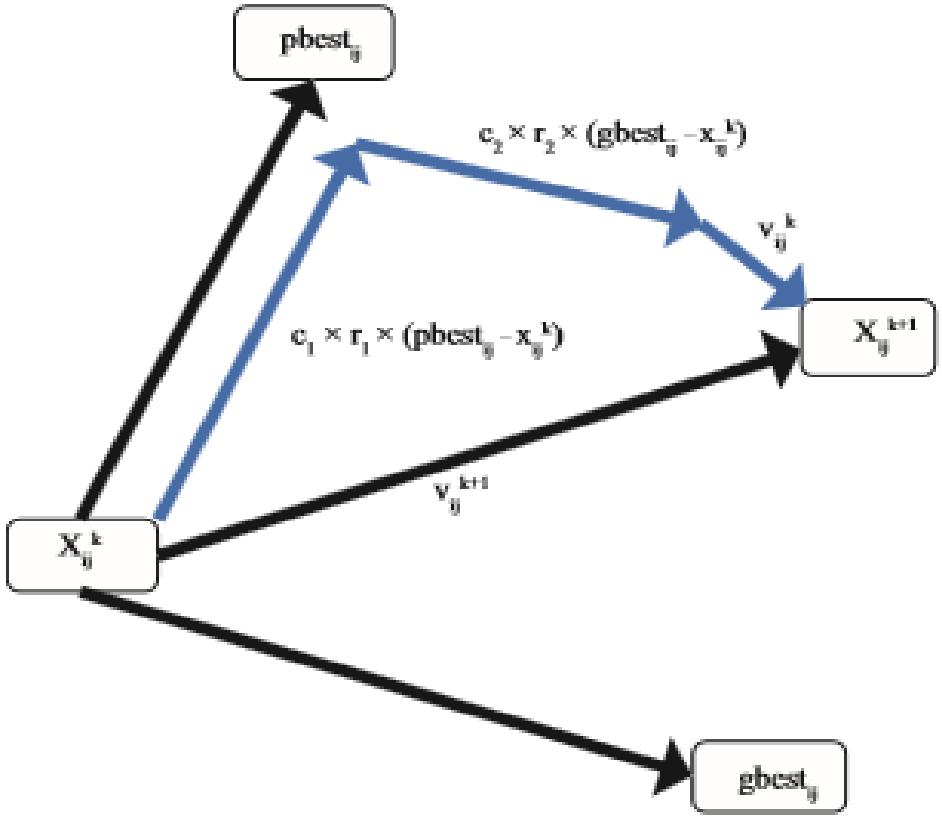

Figure 3.7: The velocity and position updating of a particle at kth generation

The basic PSO is calculated by Eq for updating the speed and position. (3.1) (3.2)) and (3.2). For these equations, r1 and r2, in the interval YALC ̇IN et al. / Turk J Elec Eng & Comp Sci+, are two uniformly distributed random numbers (0, 1). c1and c2 are positive, usually c1= c2= 2 acceleration constants.

$$v_{ij}^{k+1} = v_{ij+c1*r1*\left(pbestij-x_{ij}^{k}\right)+c2*r2(gbest-ix_{ij}^{k})}^{k} \tag{3.21}$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \tag{3.22}$$

An inertia weight (w) parameter in a improved PSO version is displayed in Eq. (3.3) the velocity update equation shall be added.

$$v_{ij}^{k+1} = w^{k} * v_{ij}^{k} + c1 * r1 * \left(pbestij - x_{ij}^{k}\right) + c2 * r2 * (gbesti - x_{ij}^{k}) \tag{3.23}$$

w is used for global and local search balance[24] and can be updated via Eq. (3.24) or (3.25) generations after generation. Equally. The maximum and minimal inertia values (3.24), the maximum and n numbers produced is for wmax and wmin. The Eq α variable. (3.26) is the factor of decrease and the weight of inertia is linearly reduced.

$$w^k = w_{max} - k * \frac{wmax - wmin}{n} \tag{3.24}$$

$$w^{k+1} a * w^k \tag{3.25}$$

The update speed can also be determined by Eq. Eq. and (3.26) (3.29). (3.29). β is the limit factor, providing a convergence to the destination within the given limits and calculated by one pair of equations (3.26)–(3.27) or (3.28)–(3.29).

$$v_{ij}^{k+1} = x \quad * [v_{ij}^k + c1 * r1 * \left(pbestij - x_{ij}^k\right) + c2 * r2 * (gbesti - x_{ij}^k) \tag{3.26}$$

$$B\text{=c1+c2} \tag{3.27}$$

$$B\text{=c1*r1+c2*r2} \tag{3.28}$$

$$X = \frac{2}{2 - B - \sqrt{\beta * (\beta - 4)}} \tag{3.29}$$

An R vector was used by C ′avu ′slu et al in an alternative rate update approach. In Eq, the vector R. (3.30) consists of random numbers normally distributed and provides very small variations in particle velocity updating. CE is a small additional constant learning and 10−5 has been chosen here.

$$v_{ij}^{k+1} = x \quad * [v_{ij}^k + c1 * r1 * \left(pbestij - x_{ij}^k\right) + c2 * r2 * \left(gbesti - x_{ij}^k\right) + \propto \times R_{ij} \tag{3.30}$$

In ecclesiastical terms. The minimum and maximum particle limits during the one generation are Vmin and Vmax (3.31), limitation values. They are used to provide detailed searches and to avoid leaving space.

$$v_{ij}^{k+1}\begin{cases} v_{ij}^{k+1}, V_{min} < v_{ij}^{k+1} < V_{max} \\ V_{min} v_{ij}^{k+1} \leq V_{min} \\ V_{max}, V_{max} \leq v_{vj}^{k+1} \end{cases} \quad (3.31)$$

The PSO study uses an ANN to develop a perfect network model and to increase the ANN's performance. During the training stage the mean squared error (MSE) is used to calculate an Eq particle fitness value. (3.31), where e represents errors in the amount of data in the training data set between the desired result and the resulting data after submission to the network. The Pi particle structure is provided by Eq. (June 3).

$$MSE = \frac{1}{2S}\sum_{i=0}^{S} e_i^2 \quad (3.32)$$

Pi =[W1i11 W1i12 ... θ1i11 ... W2i11 W2i12 ... θ2i11..W3i 1 ... θ3i11...] (3.33)

The flow chart in figure 3.2 depicts PSONN training and testing procedures, the ANN training process begins with the random initiation of weight and bias, which shows the numerical values of layer to layer connections. The weights and biases of each particle as specified in Eq are individuals. (July 26). The number of layer connections refers to the particle size or the space aspect of the search. The stop criterion for the gbest particle in Figure 3.2 is the maximum number of generation or the goal fitness value.

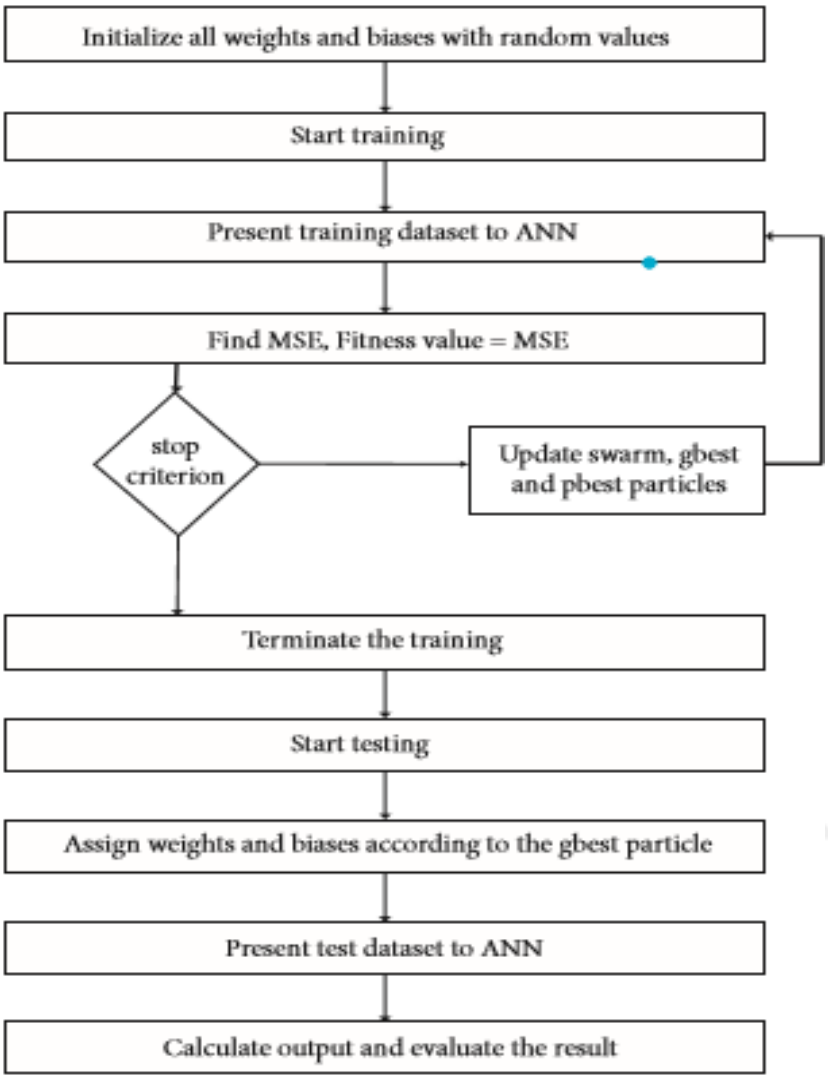

Figure 3.8: Flowchart for the training and testing of the PSONN

4. RESULTS

This chapter describes the extensive results of the simulation for methods investigated through research data sources (Diabetes Dataset), such as ANN-PSO and ANN-BP, in this project. The properties and structure of this dataset have already been discussed. To train, to test and measure the various performance parameters, we have deployed ANN with PSO and BP algorithms.

Comparative Results

We used the 70 % training and 30 % testing scenario with varying number of hidden layers such as 5, 10 and 15 for both ANN-PSO and ANN-BP.

Diabetes Dataset Results

First we present the individual for ANN-PSO and ANN-BP using diabetes dataset. We used the 5 hidden layers.

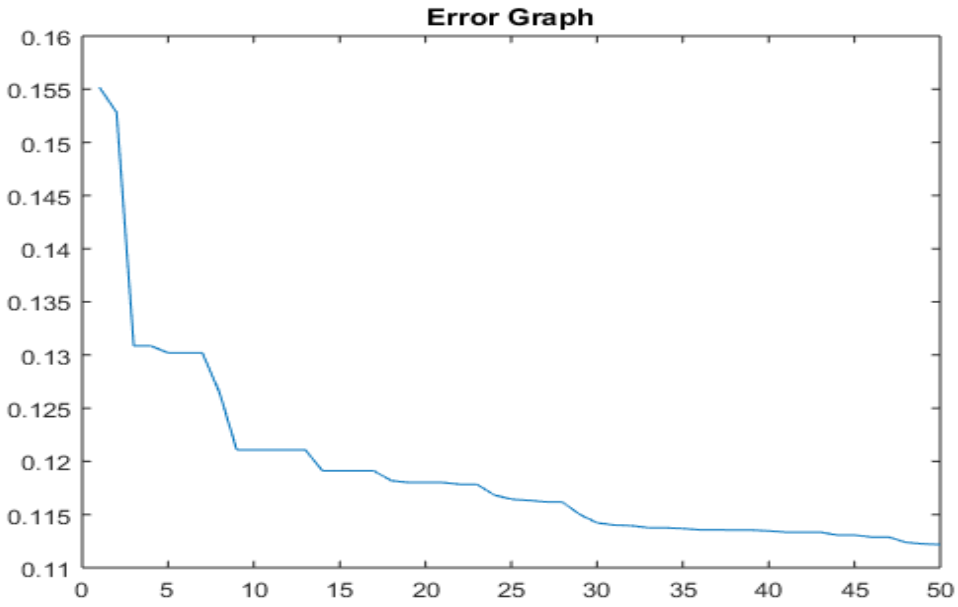

Figure 4.1: Error graph performance using ANN-BP for Diabetes dataset

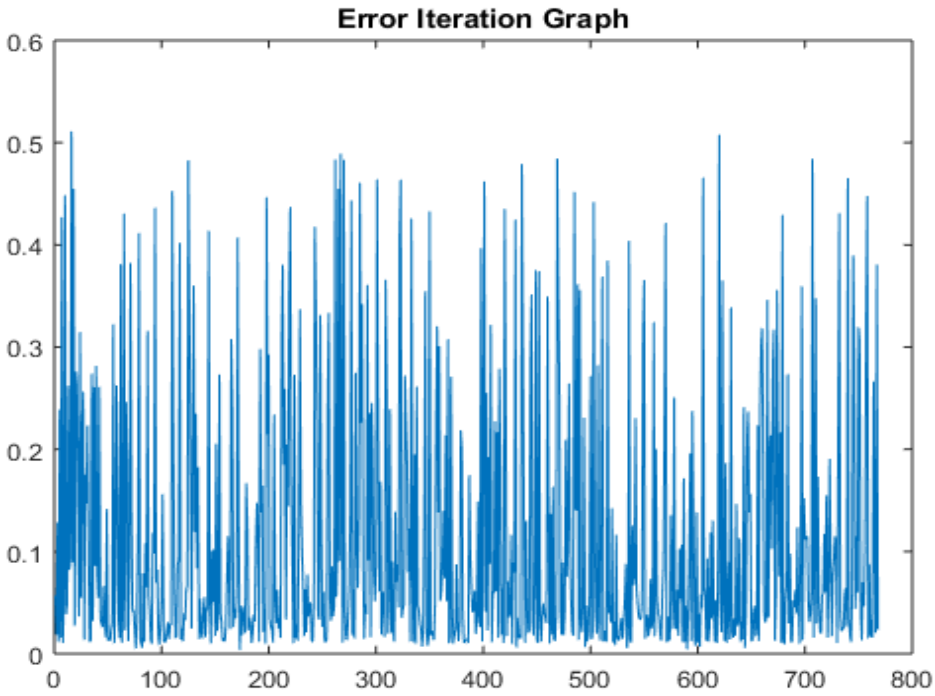

Figure 4.2: Error iteration graph performance using ANN-BP for Diabetes dataset

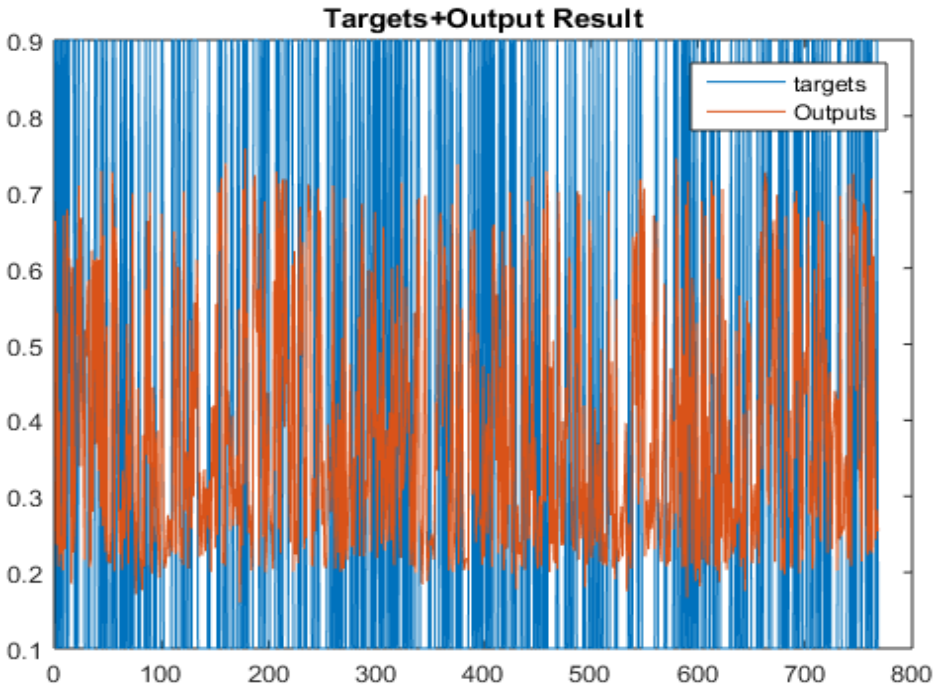

Figure 4.3: Error iteration graph performance using ANN-BP for Diabetes dataset for original target and predicted outcomes.

The results computed are:

**********Results**********

Training Error is (Fitness/MSE): 0.011906%

Training Accuracy is: 98.809364%

Testing Error is (Fitness/MSE): 0.011217

Testing Accuracy is: 98.878309

Training Sensitivity is: 1.000000

Training Specificity is: 0.990099

Testing Sensitivity is: 0.989899

Testing Specificity is: 0.980392

Figure 1, 2 and 3 are showing the fitness or MSE or error outcomes by using the existing ANN-PSO approach. Similarly, the figures 5.4 to 5.6 are showing the error results using same dataset for ANN-BP method.

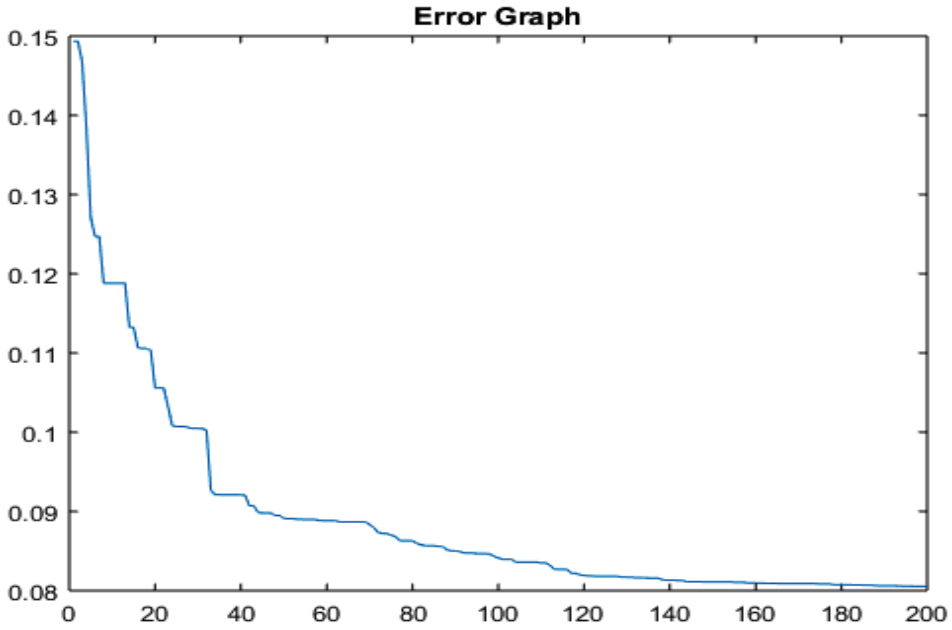

Figure 4.4: Error graph performance using ANN-PSO for Diabetes dataset

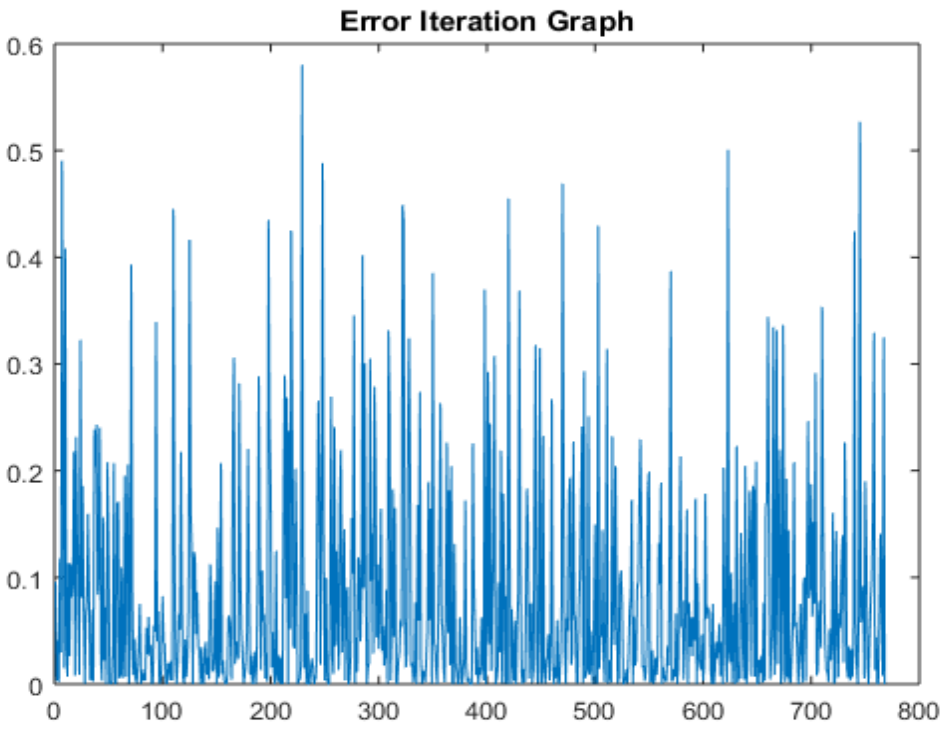

Figure 4.5: Error iteration graph performance using ANN-PSO for Diabetes dataset

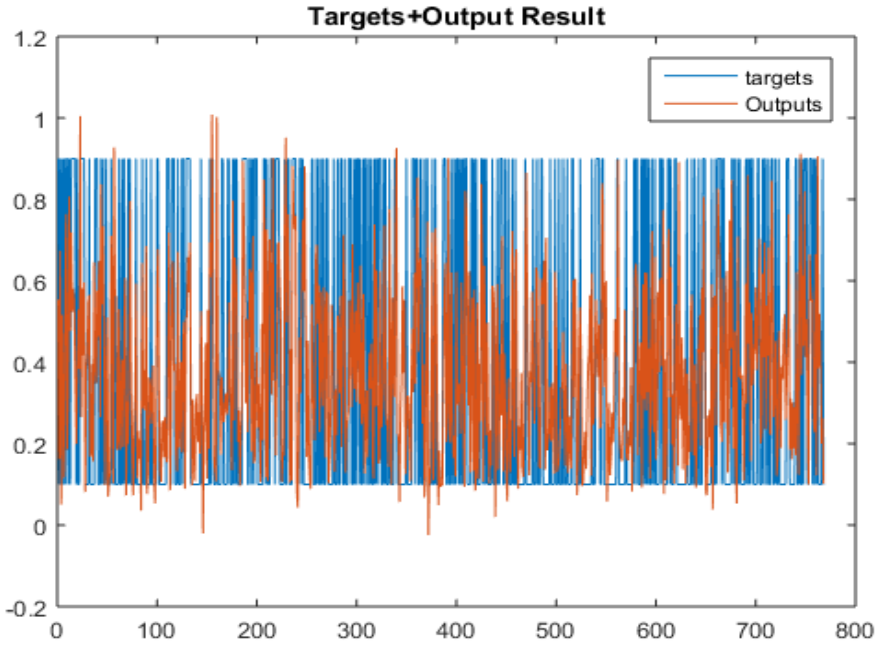

Figure 4.6: Error iteration graph performance using ANN-PSO for Diabetes dataset for original target and predicted outcomes.

The results computed are:

**********Results**********

Training Error is (Fitness/MSE): 0.008900%

Training Accuracy is: 99.110017%

Testing Error is (Fitness/MSE): 0.008052

Testing Accuracy is: 99.194834

Training Sensitivity is: 0.990099

Training Specificity is: 0.990099

Testing Sensitivity is: 1.000000

Testing Specificity is: 0.990099

The result shows that ANN-PSO method outperformed the ANN-BP as there is significant reduction in error/MSE performance for ANN-BP in both testing and training cases as well as improved the other key performance metrics. The difference between error graphs clearly indicating the advantage of using ANN-PSO method. The above results are computed for 5 numbers of hidden layers. The figures 7, 8 and 9 are showing the comparative training error using 5, 10 and 15 hidden layers respectively.

The results are showing that reduction in errors for all training conditions. The notice from these results is that, the different between errors rate using different number of hidden layers is changing. When the number of hidden layers is lower, error performance of the proposed method is superior.

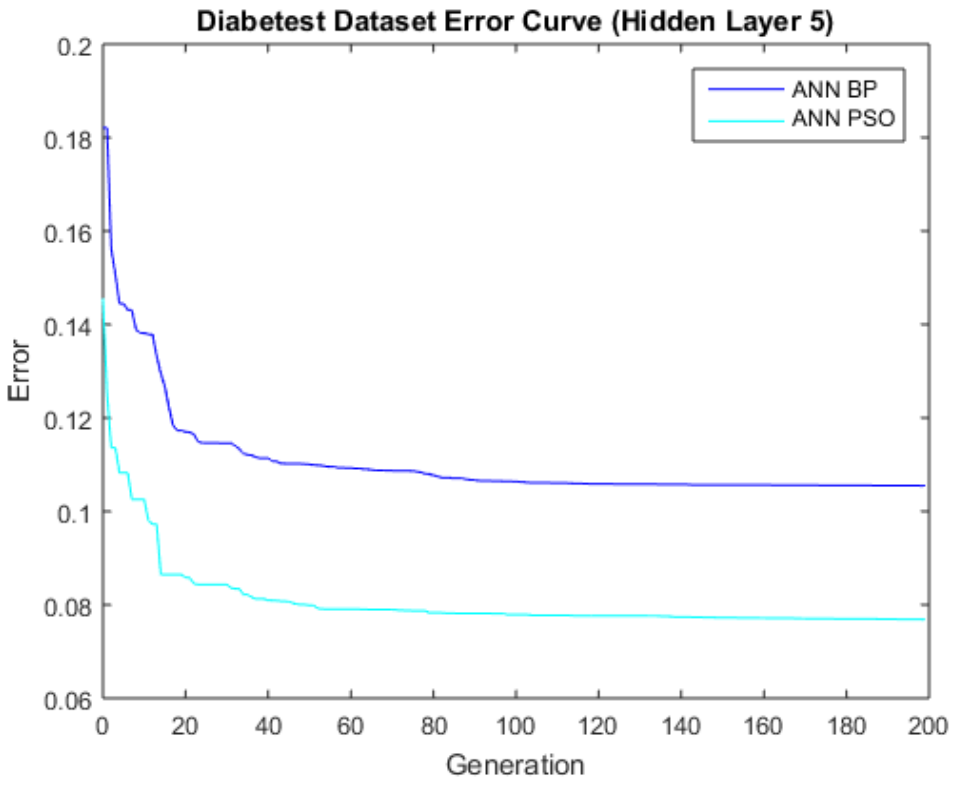

Figure 4.7: Comparative Error analysis for Diabetes dataset using hidden layers 5

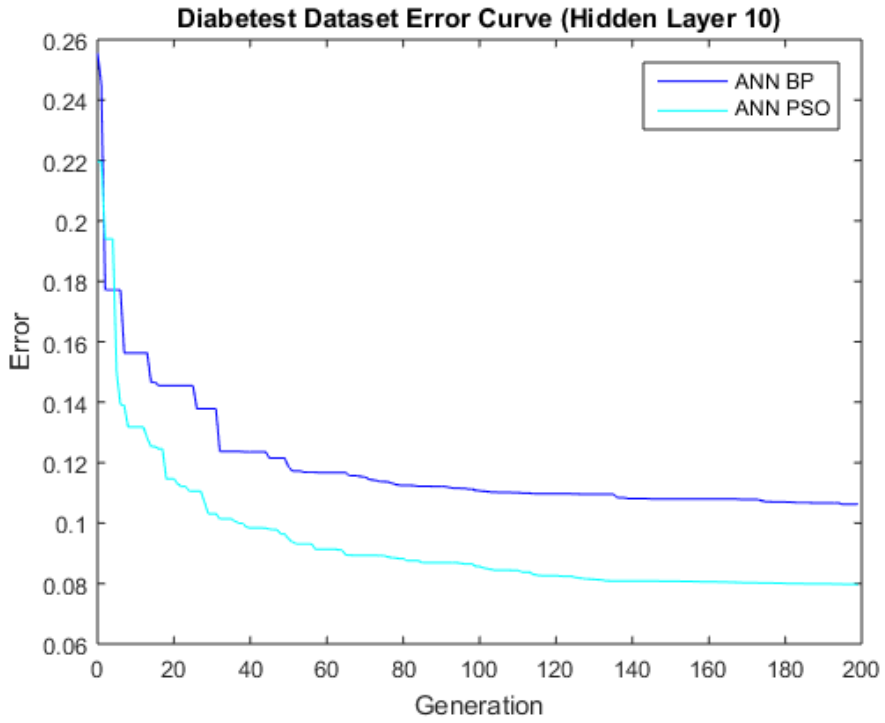

Figure 4.8: Comparative Error analysis for Diabetes dataset using hidden layers 10

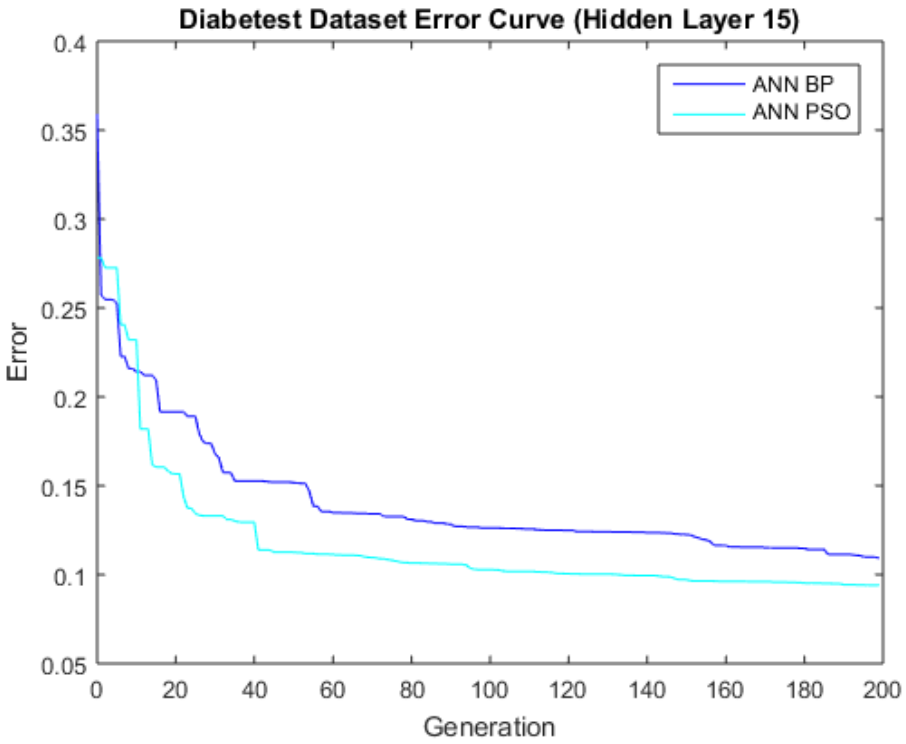

Figure 4.9: Comparative Error analysis for Diabetes dataset using hidden layers 15

5. CONCLUSSION

Comparative results show that we can optimize the performance of ANN training and testing to solve real-time problems with the advantages of a novel PSO algorithm. From these experiments, we noticed that the most weighted recognition rate fitness functions generated by ANN were those used to make use of the classification error. In terms of accuracy, error rate, sensibility rate, specificities and exactness rates for training and tests, the three bio-inspired algorithms such as BP and PSO have been compared. Compared to the BP algorithm, the PSO algorithm achieved best performance. The most frequently selected transmission function were: the gaussian functions for the basic BP algorithm; the PSO algorithm sinusoidal function. In general, it was highly promising to design ANNs with the proposed methodology. ANN is authenticated on the basis of the determination and the proposed methodology of set connections, number of neurons in hidden layers, synaptic weight management, biaisection, and transmission of each neuron.

5.1 SUGGESTIONS

In this research work, we perform the ANN training using BP and PSO algorithms 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].