DIFFERENTIAL EVOLUTION FOR NEURAL NETWORKS LEARNING ENHANCEMENT

ABDUL STTAR ISMAIL WDAA

A project report submitted in partial fulfillment of
the requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computer Science and Information System
Universiti Teknologi Malaysia

OCTOBER 2008

“To my beloved mother , my dear father, my big brother Hakki , my wife ,my children Mohammed & Marwa ,my brothers and my sisters thanks for your encouragement, support and understanding. To all my lecturers and friends , nice knowing you all and always remember our sweet memory”

ACKNOWLEDGEMENTS

First and foremost, I would like to thank ALLAH S.W.T. for all the achievements that I have gained today. Next, I wish to extend my grateful appreciation to all those who have contributed directly and indirectly to the preparation of this study. I would like to take this opportunity to thank my supervisor, Assoc. Prof. Dr. Siti Mariyam Shamsuddin for attention, encouragement and guidance throughout the period of this study. Not forgetting my beloved family for all the supports and understandings that they have given to me. Not forgetting also, my examiners Assoc. Prof. Dr. Puteh binti Saad and Dr. Siti Zaiton binti Mohd Hashim for many helpful suggestions

I am grateful to all my colleagues, friends, staff, and lecturers in Faculty of Computer Science and Information System, Universiti Teknologi Malaysia and Taiz University for their help and support at every step during this course of studies.

ABSTRACT

Evolutionary computation is the name given to a collection of algorithms based on the evolution of a population toward a solution of a certain problem. These algorithms can be used successfully in many applications requiring the optimization of a certain multi-dimensional function. These algorithms have widely been used to optimize the learning mechanism of classifiers, particularly on Artificial Neural Network (ANN) Classifier. Major disadvantages of ANN classifier are due to its sluggish convergence and always being trapped at the local minima. To overcome this problem, Differential Evolution (DE) has been used to determine optimal value for ANN parameters such as learning rate and momentum rate and also for weight optimization. In ANN, there are many elements need to be considered, and these include the number of input nodes, hidden nodes, output nodes, learning rate, momentum rate, bias parameter, minimum error and activation/transfer functions. These elements will affect the speed of neural network learning. DE has been applied successfully to improve ANN learning from previous studies. However, there are still some issues on DE approach such as longer training time to produce the output and the usage of complex functions in selection, crossover and mutation calculation. In this study, DE is chosen and applied to feed forward neural network to enhance the learning process and the network learning is validated in terms of convergence rate and classification accuracy. Three programs have developed; Differential Evolution Neural Network (DENN), Genetic Algorithm Neural Network (GANN) and Particle Swarm Optimization with Neural Network (PSONN) to probe the impact of these methods on ANN learning using various datasets. The results have revealed that DENN has given quite promising results in terms of convergence rate and smaller errors compared to PSONN and GANN.

ABSTRAK

Evolusi perhitungan merupakan koleksi algoritma-algoritma penyelesaian masalah menggunakan konsep evolusi terhadap sesuatu populasi. Algoritma-algoritma ini amat berkesan di dalam aplikasi-aplikasi yang memerlukan pengoptimuman fungsi multi-dimensi tertentu. Ia telah digunakan secara meluas untuk mengoptimumkan mekanisma pembelajaran bagi banyak pengklas, terutamanya Rangkaian Neural Buatan (ANN). Kekangan-kekangan utama ANN ialah kelembapan kadar penumpuan pembelajarannya dan kecenderungan terperangkap di dalam minima setempat. Bagi mengatasinya, Evolusi Pembezaan (DE) digunakan untuk mencari nilai optimum parameter-parameter ANN seperti kadar pembelajaran, momentum dan nilai-nilai pemberat. Di dalam ANN, terdapat banyak elemen yang perlu dipertimbangkan, diantaranya bilangan nod input, nod tersembunyi dan nod output, kadar pembelajaran dan momentum, pekali bias, ralat minimum dan fungsi pengaktifan/pengumpukan. Kesemua elemen akan mempengaruhi kepantasan pembelajaran rangkaian neural. Menurut kajian terdahulu, DE berperanan di dalam penambahbaikan pembelajaran ANN. Namun, pendekatan DE masih berhadapan dengan beberapa isu seperti pelanjutan tempoh latihan dan penggunaan fungsi-fungsi kompleks di dalam proses-proses pemilihan, penentuan lintasan serta aturan mutasi. Di dalam kajian ini, DE telah dipilih dan diaplikasikan pada rangkaian neural suap-hadapan untuk merangsang proses pembelajarannya. Pembelajaran rangkaian pula disahkan berdasarkan kadar penumpuan serta ketepatan pengelasan. Tiga buah program telah dibangunkan; Evolusi Pembezaan Rangkaian Neural (DENN), Algoritma Genetik Rangkaian Neural (GANN) dan Pengoptimuman Kawanan Partikel dengan Rangkaian Neural (PSONN) untuk menyiasat impak setiap kaedah terhadap pembelajaran ANN ke atas pelbagai set data. Keputusan ujikaji mendedahkan kecemerlangan DENN di dalam mempercepatkan kadar penumpuan dan menghasilkan ralat yang lebih kecil berbanding PSONN dan GANN.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Δt	-	Time interval
c_1	-	Acceleration constants for gbest
c_2	-	Acceleration constants for pbest
η	-	Learning rate
W	-	Weights obtain appropriate functional
G	-	Generation to which the population belongs.

LIST OF ABBREVIATION

DE	-	Differential Evolution
ANN	-	Artificial Neural Network
DENN	-	Differential Evolution Feedforward Neural Network
EA	-	Evolutionary Algorithms
EC	-	Evolutionary computing
EP	-	Evolutionary Programming
GA	-	Genetic Algorithm
GANN	-	Genetic Algorithm Neural Network
GP	-	Genetic Programming
LISP	-	List Processing language
MLP	-	Multilayer Perceptron
ES	-	Evolution Strategies
MPPSO	-	Multi-phase Particle Swarm Optimization
NN	-	Neural Network
PDP	-	Parallel Distributed Processing
PSO	-	Particle Swarm Optimization
PSONN	-	Particle Swarm Optimization Feedforward Neural Network

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Evolutionary computing (EC) is an exciting development in Computer Science. It amounts to building, applying and studying algorithms based on the Darwinian principles of natural selection, the main concepts behind evolutionary computing. Figure 1.1 illustrates the cycle of Evolutionary Computing.

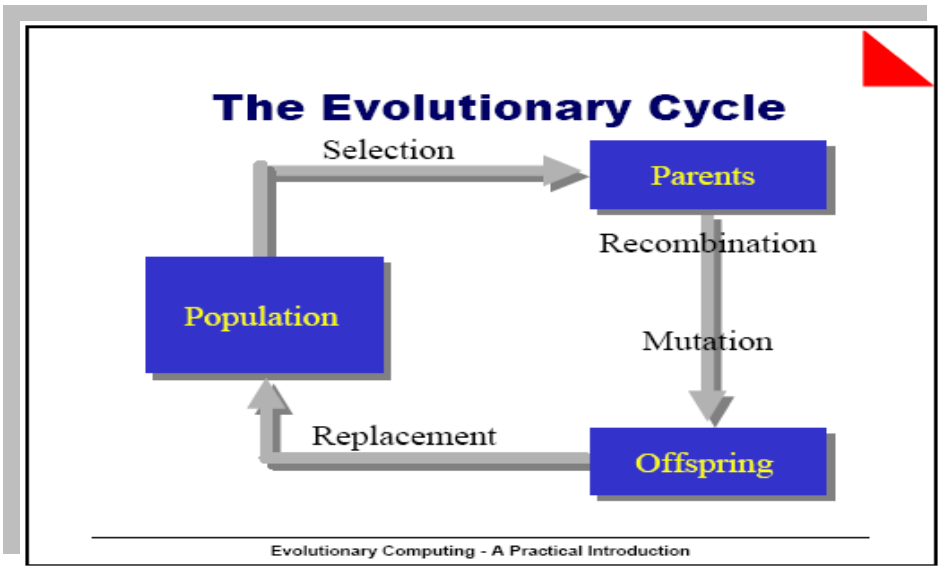

Figure 1.1: The Evolutionary Cycle (Yao, 1999)

Evolutionary computing contain a grub of the algorithms mention some of them. Genetic Algorithm (GA) is one of the famous evolutionary techniques in ANN learning. A basic genetic algorithm (GA) comprises of three genetic operators: selection, crossover, and mutation. Starting from an initial population of strings (representing possible solutions), GA uses these operators to calculate successive generations. First, pairs of individuals of the current population are selected to mate with each other to form the offspring, which then form the next generation.Another algorithm similar to GA called (GP).

John Koza of Stanford University had developed techniques Genetic Programming (GP) in the 1990. GP is a special implementation of GAs. It uses hierarchical genetic material that is not limited in size. The members of a population or chromosomes are tree structured programs and the genetic operators work on the branches of these trees. The structures generally represent computer programs written in LISP. based on similar spirit of (GA).was developer but the different detail Called (EA)

Evolutionary algorithms (EA) do not require separation between a recombination and an evaluation space. The genetic operators work directly on the actual structure. The structures used in EAs are representations that are problem dependent and more natural for the task than the general representations used in GAs. another algorithm from Evolutionary computing is good and widespread called (EP)

Evolutionary programming (EP) is currently experiencing a dramatic increase in popularity. Several examples have been successfully completed that indicate EP is full of potential. Koza and his students have used EP to solve problems in various domains including process control, data analysis, and computer modeling. Although at the present time the complexity of the problems being solved with EP lags behind the complexity of applications of various other evolutionary computing algorithms, the technique is promising. Because EP actually manipulates entire computer

programs, the technique can potentially produce effective solutions to very large-scale problems. To reach its full potential, EP will likely require dramatic improvements in computer hardware. In this study will use an algorithm(DE).

.Differential Evolution (DE) algorithm is an evolutionary algorithm, which was proposed by Rainer Storn in 1995.It is a small and simple mathematical model of a big and naturally complex process of evolution. So, it is easy and efficient. According to (Abbass *et al*, 2001), this algorithm is simple and one of the most powerful tools for global optimization. A genetic algorithms, evolution strategies, or evolutionary programming is the three basic trends of evolutionary optimization, also well known under the common term of evolutionary algorithms. Recently with the advent of new ideas and new methods in optimization, including DE too, they have got a second fashionable name of artificial evolution, and DE belongs to this suite. The intelligence usage of differences between individuals realized in a simple and fast linear operator, so-called differentiation, makes differential evolution unique; hence, Differential Evolution (DE) is proclaimed.

1.2 Problem Background

This section will discuss the briefly and issues of EC and DE algorithms in practical situations based on previous studies. The summary of these issues are given in table 1.1.

1.2.1 Evolutionary Computing (EC)

Evolutionary computation is the use of evolutionary systems as computational processes for solving complex problems, is a tool used by computer scientists and engineers who want to harness the power of evolution to build useful

new artifacts, by biologists interested in developing and testing better models of natural evolutionary systems, and by artificial life scientists for designing and implementing new artificial evolutionary worlds. In this clear and comprehensive introduction to the field, Kenneth De Jong presents an integrated view of the state of the art in evolutionary computation. Although other books have described such particular areas of the field as genetic algorithms, genetic programming, evolution strategies, and evolutionary programming, Evolutionary Computation is noteworthy for considering these systems as specific instances of a more general class of evolutionary algorithms. This useful overview of a fragmented field is suitable for classroom use or as a reference for computer scientists and engineers.

EC does not require any in-depth mathematical understanding to be applied. They can provide solutions to problems that were previously out of range which cannot be solved mathematically. DE is extremely robust; they cope well with noisy, inaccurate and incomplete data. This approach is also relatively cheap, has quick implementation, and easily hybridized. When DE is combined with other techniques such as greedy methods, heuristics, simulated annealing and neural networks, the outcomes are promising. They are extremely adaptable; changed the priorities can be incorporated simply by altering the weights in the fitness function. It is modular and therefore portable because the evolutionary mechanism is separate from the problem representation. They can be transferred from one problem to other problem, and provides an extremely open and flexible approach to design, allows arbitrary constraints, simultaneous multiple objectives with mixing continuous and discrete parameters. Unlike other methods, evolutionary algorithms perform better when they are implemented on parallel computers.

What is the problem in Evolutionary Computing? It has no guarantee for optimal solution within finite time - lacks the killer instinct. It has weak theoretical basis, thus may need extensive parameter tuning. Furthermore, it is often computationally expensive and slow.

1.2.2 Differential Evolution (DE)

Differential Evolution is a small and simple mathematical model of a big and naturally complex process of evolution. So, it is easy and efficient! First and foremost Differential Evolution (DE) is an optimization algorithm. And without regard to its simplicity DE was and is one of the most powerful tools for global optimization.Perhaps you have heard about genetic algorithms, evolution strategies, or evolutionary programming. These are three basic trends of evolutionary optimization, also well known under the common term of evolutionary algorithms. Lately, with the advent of new ideas and new methods in optimization, including DE too, they have got a second fashionable name of artificial evolution. DE belongs to this suite.

Why use Differential Evolution? Global optimization is necessary in fields such as engineering, statistics and finance. But many practical problems have objective functions that are non differentiable, non-continuous, non-linear, noisy, flat, multi-dimensional or have many local minima, constraints or stochastic. Such problems are difficult if not impossible to solve analytically .DE can be used to find approximate solutions to such problems.

According to lonen *et al.*, (2003), Differential Evolution (DE) has been introduced successfully and applied in many artificial and real optimization problems and applications such as aerodynamic shape optimization automated mirror design optimization of radial active magnetic bearings optimization of fermentation using a high ethanol tolerance yeast and mechanical engineering design. DE based neural network training algorithm, was initially introduced in the form of probing ANN parameters as global optimizer and were compared to other neural network training methods. Similar results were also reported by others, and more comprehensive studies are needed to reliably evaluate the necessity of using differential evolution or other stochastic methods in the training of artificial neural networks and to establish the kind of advantages contemporary stochastic methods can provide.

According to lonen et al. (2003), Differential Evolution has been used for feed-forward neural network training. A new parallelization scheme for the computation of the fitness function is proposed. This scheme is based on data decomposition. Each of the learning set and the population of the evolutionary algorithm are distributed among processors. The processors form a pipeline using the ring topology. In a single step each processor computes the local fitness of its current subpopulation while sending the previous subpopulation to the successor and receiving next subpopulation from the predecessor. Thus it is possible to overlap communication and computation using non-blocking MPI routines. The approach was applied to several classification and regression learning problems. The scalability of the algorithm was measured on a compute cluster consisting of sixteen two-processor servers connected by a fast Infiniband interconnect. The results of initial experiments show that for large datasets the algorithm is capable of obtaining very good, near linear speedup.

1.2.3 Particle Swarm Optimization

PSO algorithm has been paid more and more attention to by researchers. Lee et al. (2005) have used PSO and GA for excess return evaluation in stock market. Based on their experiment, it is proven that PSO algorithm is better compared to GA. PSO can reach the global optimum value with less iteration, keep equilibrium versus GA and shows the possibility to solve the complicated problem using only basic equation without crossover, mutation and other manipulation as in GA. The application for stock trading using PSO also has been done by Nenortaite et al. (2004) where it shows good profit accumulation results. Another study by Zhang et al. (2000) applied two real problems in medical domain which are breast cancer and heart disease to feed-forward ANN with PSO called Particle Swarm Optimization Feed-forward Neural Network (PSONN). The result shows that PSONN has better accuracy in classified data compared to other algorithms. Al-kazemi et al. (2002) was conducted a study on Multi-phase Particle Swarm Optimization (MPPSO) where it evolves multiple groups of particle. Based on the previous researcher works, it gives a good credit to PSO.

1.2.4 Genetic Algorithm

Genetic Algorithm (GA) is one of the algorithms proposed to determine the learning rate and momentum rate and will produce a set of weight that can be used for testing related data. Table 1 briefly described the finding from several researchers in order to increase learning speed (Fnaiech *et al.,* 2002), avoid from trapped into local minima (Wyeth *et al.,* 2000) and better classification result.

Table 1.1 illustrates the comparison each of Evolutionary Computing Techniques, and these include Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Differential Evolution (DE). (PSO, GA AND DE).

Table 1.1 A Comparison of PSO,GA and DE

EC TECHNIQUE	Strength	Weaknesses
PSO	1. PSO. The gradient-based method has the advantage of being computationally very efficient. 2. PSO is easy to implement and there are few parameters to adjust. 3. PSO has been successfully applied in many areas	1. Disadvantage of settling into local minima. 2. PSO executed for number of iterations, can not identify the nonlinear system with higher precision with another algorithm. 3. The PSO algorithm required excessive training time. The training process is often off-line work and the time cost doesn't influence practical use.
GA	1. Genetic algorithms are intrinsically parallel. Can explore the solution space in multiple directions at once. 2. Genetic algorithms are particularly well-suited to solving problems where the space of all potential solutions is truly huge - too vast to search	According ((Fnaiech *et al*., 2002). 1. The genetic algorithm may be unable to find a solution to the problem, or may end up solving the wrong problem. 2. The genetic algorithm may not explore enough of the solution space to consistently find good

	exhaustively in any reasonable amount of time. 3. genetic algorithms is that they perform well in problems for which the fitness landscape is complex - ones where the fitness function is continuous, noisy, changes over time, or has many local optima.	solutions.3. Problem that can occur with a GA is known as premature convergence.
DE	According lonen (et al 2000) 1. Efficiency. A special stress is laid on the efficient creation of a new member of a population, instead of the mutation of current individuals. 2. Flexibility. The new algorithm is more flexible to use and adapts to modification; it is preferred for research purposes. 3. In fixing the differentiation strategy and appropriate crossover, variation operations can be convoluted to a single equation similar. Moreover, differentiation Synthesizes in itself the fundamental ideas in optimization.	

1.3 Problem Statement

In Artificial Neural Network (ANN), there are many elements to be considered such as the number of input, hidden and output nodes, learning rate, momentum rate, bias, minimum error and activation transfer function. All these elements will affect the convergence of ANN learning. As mentioned before, GA can be used to determine some parameters and provide the best pattern of weight in order to enhance the ANN learning. In this study the Evolutionary Optimization technique called Differential Evolution is employed to see the convergence speed for feed forward neural network learning. In order to evaluate the performance, three programs have been developed and these include: Differential Evolution

Feedforward Neural Network (DENN), Genetic Algorithm Neural Network (GANN) and Particle Swarm Optimization (PSO).

The hypothesis of this study can be stated as:

> How efficient is DE algorithm in enhancing and optimizing feed forward neural network learning compare to GA and PSO?

1.4 Project Aim

This project aims to investigate the efficiency of DE in optimizing and improving the performance of ANN learning.

1.5 Objectives

The objectives of this study are as follows:

1. To develop a learning mechanism using Differential Evolution algorithm for Artificial Neural Network learning .
2. To analyze and evaluate the effectiveness of applying the Differential Evolution in Neural Network learning.
3. To compare the results between DENN with GANN and DENN with PSONN in terms of Convergence rate and Classification accuracy.

1.6 Project Scope

The focus of the study is on Differential Evolution technique to optimize and enhance neural network learning. The scopes of this project are as follows:

1. four dataset will be used and these are XOR, Cancer, Iris and Heart. The performance of DENN will be compared with PSO-based NN and GA-based NN.
2. The DE program will be integrated to feedforward neural network using Matlab version 7.
3. The PSO program has been developed and applied to feedforward neural network using Microsoft Visual C++ 6.0.
4. The GA program has been developed and applied to feed forward neural network using Microsoft Visual C++ 6.0.
5. Testing data is not conducted in this study.

1.7 Significance of Project

The performance between DE-based neural network with GA-based neural network and PSO based neural network will be compared and analyzed. Thus, we can determine which method is better for neural network learning. This is important to identify the suitable learning method for future study and can be implemented in real world application.

1.8 Organization of Report

This thesis consists of five chapters. Chapter 1 presents the introduction of the study, problems background, the problem statement, objectives and the scope. Chapter 2 gives literature reviews on ANN, DE, GA and PSO. Research methodology is discussed in Chapter 3 and Chapter 4 presents the experimental results. The conclusion and suggestions for future work are explained in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

The major discussion in this chapter is on Evolutionary Computing, particularly on DE algorithm. The discussions also include Artificial Neural Network (ANN), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). The first part reviews Artificial Neural Network (ANN), and Evolutionary Computing (EC) algorithm. While, the second part focuses on Differential Evolution technique, their differences and applications.

2.1 Introduction

Artificial Neural Network (ANN) is the most popular supervised learning technique. In ANN, there are many elements to be considered such as number of input, hidden and output nodes, learning rate, momentum rate, bias, minimum error and activation/transfer function. These elements will affect the convergence of BP learning. The learning consists of the following steps:

1. An input vector is presented at the input layer.
2. A set of desired output is presented at the output layer.
3. After a forward pass is done, the errors between the desired and actual output are compared.
4. The comparison results are used to determine weight changes (backwards) according to the learning rules.

To guide ANN learning, GA is employed to determine the best learning rate, momentum rate and weight optimization. This is because ANN takes long time for training (Cho et al., 1991).

2.2 Artificial Neural Network (ANN)

ANN consists of a parallel collection of simple processing units (neurons/nodes) arranged and interconnected in a network topology (Yao, 1993). ANN that based from biological nervous system, are known as parallel distributed processing (PDP) systems since they are based on the idea that intelligent function is created through adaptation of the interconnections between simple interacting processing units in a network (Luger, 2002). ANN consists of a set of interconnected processing units known as node, neurons or cells as shown in figure 2.1. Each node has it activation functions and the common activation function is the sigmoid function. The activation signal sent (output) by each node to other nodes travel through weighted connection and each of these nodes accumulates the inputs it receives, producing an output according to an internal activation function. (Zhang et al., 2000) addressed that the information processing capability ANN is closely related to its architecture and weights. Figure 2.2 shows the interconnection between nodes is usually referred as a fully connected network or multilayer perceptron (MLP). Multilayer architecture means that the network has several layers or nodes usually referred to as input layer, hidden layer and output layer. MLP network can be used with great success to solve both classification and function approximation problems (Van den Bergh, 1999). There are two types of learning networks which are supervised learning and unsupervised or self-organizing. Supervised learning is when the input and desired output are provided while for unsupervised learning, only input data is provided to the network. According to (Ben Kröse *et al.* ,1996).The nodes in an ANN with unsupervised learning are trained to respond to patterns within the inputs. Thus the system must discover features of the input population on its own, without a priori set of input output training pairs (Ben Kröse *et al*, 1996).

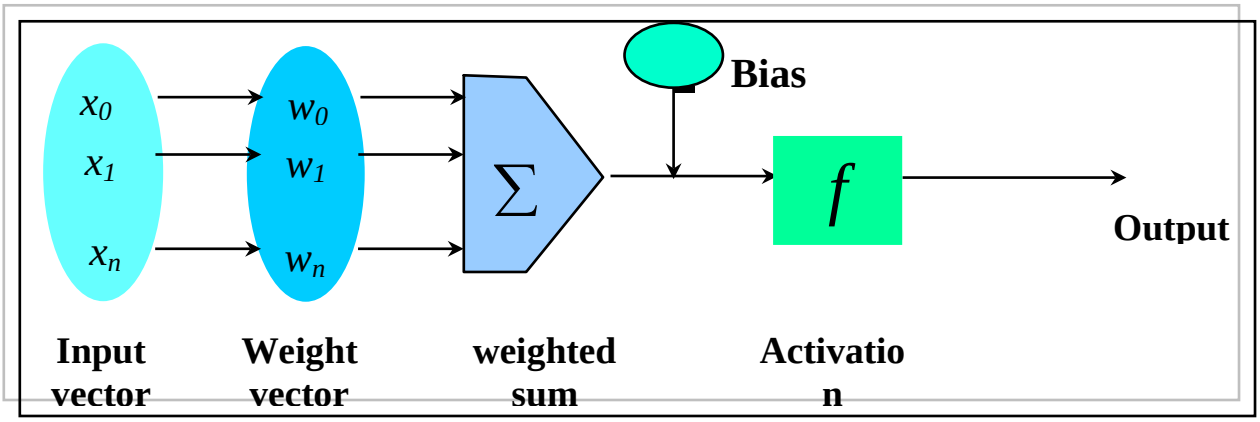

Figure 2.1: Artificial Neural Network Model

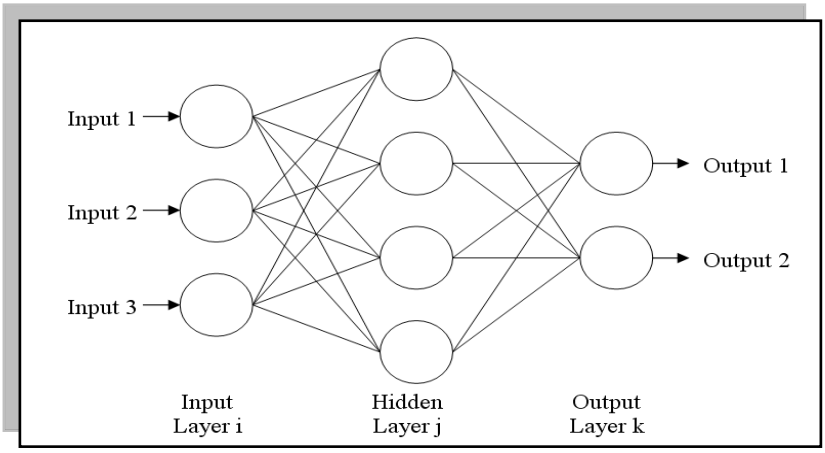

Figure 2.2: Combination of ANN –> Multilayer Perceptron (MLP)

In order to get the desired output from ANN, the output from the network is compared to actual desired output. During training, the network tries to match the outputs with the desired target values. Network need to review the connection weight to get the best output. One of the learning techniques that are commonly used is BP. According to (Rumelhart *et al.*, 1986). BP algorithm is a supervised learning method, which it is the most widely used algorithm for training MLP neural network. The idea of the BP is to reduce this error, until the ANN learns the training data. The training begins with random weights, and the goal is to adjust them so that the learning error will be at minimal. ANN nodes in BP algorithm are organized in layers, send their signals forward and then the learning error (difference between actual and expected results) is calculated and propagated backwards until met satisfactory learning error.

There are two phases in ANN learning algorithm which are feed forward phase and backward phase. In feed forward process, the dataset is presented to the input layer and the network propagates the input pattern from layer to layer until the output pattern is generated. The output is obtained from a summation of the weighted input of a node and maps to the network activation function. Equation 2A and 2B show the calculation formula from input layer (*i*) to hidden layer (*j*) and equation 2C and 2D show formula for hidden layer (*j*) to output layer (*k*). The network activation function as in equations 2 A and 2C is Sigmoid Activation Function.

Between input (*i*) and hidden (*j*)

$$O_j = f(net_j) = \frac{1}{1+e^{-net_j}} \tag{2A}$$

$$net_j = \sum_j w_{ij} O_i + \theta_j \tag{2B}$$

where:

O_j is the output of node *j*

O_i is the output of node *i*

w_{ij} is the weight connected between node *i* and *j*

θ_j is the bias of node *j*

Between hidden (*j*) and output (*k*)

$$O_k = f(net_k) = \frac{1}{1+e^{-net_k}} \tag{2C}$$

$$net_k = \sum_k w_{jk} O_j + \theta_k \tag{2D}$$

where:

O_k is the output of node *k*

O_j is the output of node *j*

w_{jk} is the weight connected between node *j* and *k*

θ_k is the bias of node *k*

Error is calculated using equation 2D to measure the differences between desired output and actual output that has been produced in feedforward phase. Error than propagated backward through the network from output layer to input layer as represented below. The weights are modified to reduce the error as the error is propagated.

$$error = \frac{1}{2}(Output_{desired} - Output_{actual})^2 \qquad (2E)$$

Based on the error calculated, backpropagation is applied from output (*k*) to hidden (*j*) as shown by equation 2E and 2F

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t+1) \qquad (2F)$$

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t) \qquad (2G)$$

with

$$\delta_k = O_k(1 - O_k)(t_k - O_k)$$

where:

$w_{kj}(t)$ is the weight from node *k* to node *j* at time *t*

Δw_{kj} is the weight adjustment

η is the learning rate

α is the momentum rate

δ_k is error at node *k*

O_j is the actual network output at node *j*

O_k is the actual network output at node *k*

t_k is the target output value at node *k*

Backward calculations from hidden (*j*) to input (*i*) as shown by equation 2H and 2I.

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t+1) \quad \text{(2H)}$$

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t) \quad \text{(2I)}$$

with

$$\delta_j = O_j(1-O_j)\sum_k \delta_k w_{kj}$$

$$O_k = f(net_k) = \frac{1}{1+e^{-net_k}}$$

$$net_k = \sum_k w_{kj} O_j + \theta_k$$

where

$w_{ji}(t)$ is the weight from node *j* to node *i* at time *t*

Δw_{ji} is the weight adjustment

η is the learning rate

α is the momentum rate

δ_j is error at node *j*

δ_k is error at node *k*

O_i is the actual network output at node *i*

O_j is the actual network output at node j

O_k is the actual network output at node k

w_{kj} is the weight connected between node j and k

θ_k is the bias of node k

This process is repeated iteratively until convergence is achieved (targeted learning error or maximum number of iteration).

2.1 Evolutionary Computing.

Evolutionary computation. is a biologically inspired method of computation and has been applied to a wide variety of problems. The paradigm is inspired by the evolution exhibited by living organisms. It consists of a population of individuals (solutions for a problem) on which reproduction,recombination, mutation and selection are iteratively performed resulting in the survival of the fittest solution occurring in the population of solutions.

The Evolutionary Computation techniques were proposed in the late 1950's by a number of different researchers(Jong et al., 1997). However the research area did not begin to gather much interest until the works by (Fogel et al., 1966). proposing evolutionary programming, (Holland,1975) proposing genetic algorithms and Rechenberg (Jong et al.,1997).proposing evolutionary strategies were ublished.Each of these strategies developed independently and it was not until the early 1990's that a generic term would itself evolve: evolutionary computation.

The field of evolutionary computation was proposed so as to unify efforts from each of the evolutionary based search techniques. Prior to this uniting of the field, a significant amount of cross collaboration had already occurred, specific operators were tried on differing evolutionary based methods, and a number of joint conferences were held to highlight the area of evolutionary computation.

2.3.1 Evolutionary Programming

(Fogel et al., 1966). while working in the artificial intelligence field identified evolutionary programming (EP) as a computational technique. Fogel began to explore a differing approach to the traditional AI. Principally he investigated the area of intelligent behaviour which he felt exemplified real artificial intelligence.

> A population of finite-state machines is exposed to the environment, that is, the sequence of symbols that have been observed up to the current time. For each parent machine, as each input symbol is offered to the machine, each output symbol is compared with the next input symbol. The worth of this prediction is then measured with respect to the payoff function (e.g. all-none, absolute error, squared error, or any other expression of the meaning of the symbols). After the last prediction is made, a function of the payoff for each symbol (e.g. average payoff per symbol) indicates the fitness of the machine (Jong, et el., 1997).

The concept of evolutionary programming consists of basically 3 steps:

1. A random set of solutions is generated to form the initial population. The number of solutions that are present in the population varies, but requires a sufficient number of solutions to provide the potential of a wide sample of the solution space.

2. The solutions undergo asexual replication to form a new generation of the population. Each parent produces offspring for the next generation. Each offspring is mutated according to a probability distribution of mutation types, ranging from a small change to very large.

3. The fitness of each offspring is computed. This fitness is then used in a competition between the individuals so as to decide which solutions will form

the parents of the next population. There is no requirement that the population size must remain constant as it is permitted that parents may produce more than one offspring. Termination of the EP process occurs after a preset number of generations or when an adequate solution is obtained. There is a related field –

genetic programming (GP). GP was proposed by (Friedberg et al., 1958; Friedberg et al.,1959)and was further explored by (Cramer,1985; Dickmanns et al., 1987), (Koza ,1992).Genetic Programming is a technique that evolves a population of computer programs according to an objective function that is determined by a program's ability to perform a given computational task. GP has been applied to a wide range of problems (Koza, 1999).

2.3.2 Evolutionary Strategies

The technique of evolutionary strategies (ES) was devised by three students at the Technical University of Berlin. They were developed as a result of the failure of numerical techniques to solve complex problems (Beasley & Heitkpetter 1997). They have been applied to a wide variety of problems, including network & routing problems, biochemistry, optics and engineering design. They have remained a popular choice as they have been shown to generate adequate solutions for users, in acceptable time frames.

The individual solutions in the ES population are described as *chromosomes*, an analogy with chromosomes that occur in biology. These chromosomes are fixed length vectors of real numbers, and together populate the population in the ES.

The steps of the original ES algorithm were relatively simple:

1. The problem is defined as an optimisation problem: there is a cost function $f:R_n\_R$, which when presented with a n-dimensional vector, returns a single real number, *the fitness*. The goal is to find the n-dimensional vector for which

the fitness is maximized.

2. A population of *p* vectors, labelled *xp* is randomly generated.

3. An offspring vector *x'p* is created from each vector *xp*, by adding a Gaussian error variable with zero mean and a pre-selected standard deviation to each component of *xp*.

4. Fitness of parents and offspring are evaluated using: *f(xp)* and *f(x'p).*

5. A new population of chromosomes is created. The population consists of the fittest chromosomes that are present in the population. A number of different approaches are in use for the selection of the next population. Plus strategy and comma strategy are two such techniques. Plus strategy takes into account the parent generation when deciding the next generation, while comma strategy only considers the offspring.

6. The process continues until the termination criteria are satisfied. (De Jong et al ., 1997)

2.3.3 Genetic algorithm (GA) .

Genetic algorithm (GA) was introduced by John H. Holland in 1960's where GA was a probabilistic optimization algorithm. GA is a family of computational model inspired by evolution. The original idea came from biological evolution process in chromosomes. GA exploits idea of the survival of fittest where best solutions are recombined with each other to form new better solutions. Figure 2.1 show the Genetic Algorithm architecture.

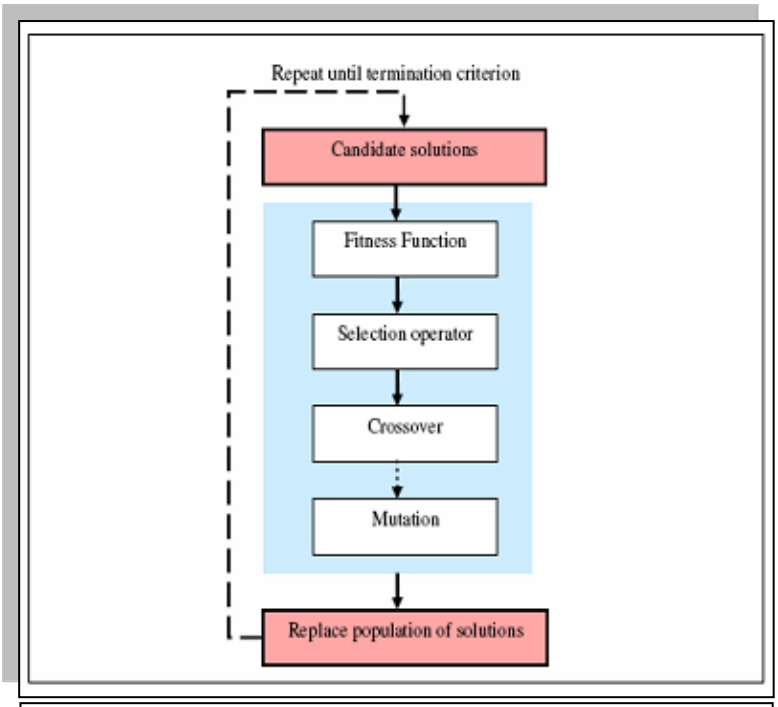

Figure 2.3 Genetic Algorithm architecture

There are three processes in GA which are selection, crossover and mutation. In the standard GA, the population is a set of individual number. Each individual represents the chromosome of a life form. There is a function that determines how fit each individual and another function that selects individuals from the population to reproduce. The two selected chromosomes crossover and split again and next the two new individuals mutate. The process is then repeated until the stop condition is met.

There are several terms in GA. *Fitness* is a measure of the goodness of a chromosome where it measure how well the chromosome fits the search space or solves the problem. *Selection* is a process for choosing a pair of organisms to reproduce while *Crossover* is a process of exchanging the genes between the two individuals that are reproducing.

Mutation is the process of randomly altering the chromosomes. Figure 2.2 shows the crossover process in GA.

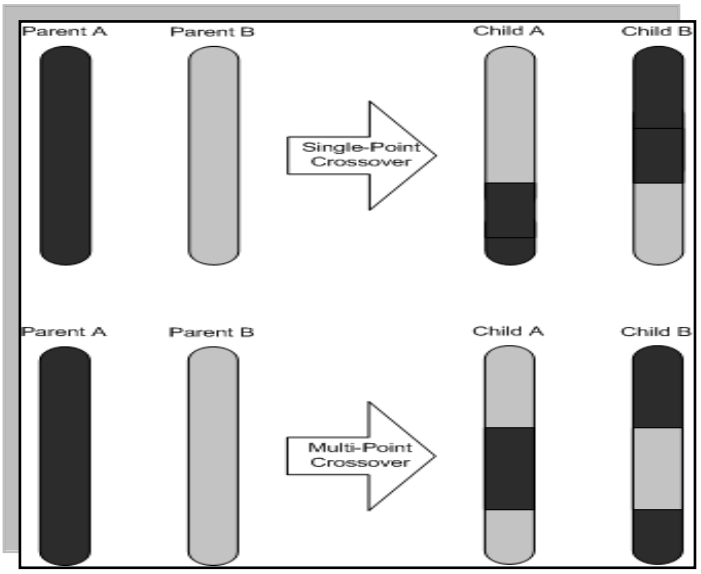

Figure 2.4: GA Crossover

The standard genetic algorithm procedures as follows:

1. Start with a population of randomly generates initial population with a number of chromosomes.
2. Define fitness function.
3. Calculate the fitness of each individual chromosome.
4. Based on fitness, select a pair of fit chromosomes for combination process.
5. Perform crossover and mutation to generate new chromosomes.
6. Place the new chromosomes in the new population (the next generation).
7. Repeat step d until size of new population equal to size in initial population.
8. Replace initial chromosome (parent) with new populations.
9. Go to step c until the termination condition met.

GA is usually applied in ANN to optimize the network because of its efficiency in giving the best parameters such as learning rate and momentum rate to avoid from being trapped in a local minima and making the convergence speed faster. GA also has been used to produce best NN architecture and for NN weight optimization.

According to (Randall *et al.*,2001). GA starts at multiple random points (initial population) when searching for a solution. Each solution is then evaluated based on the objective function. Once this has been done, solutions are then selected for the second generation based on how well they perform. After the second generation is drawn, they are randomly paired and the crossover operation is performed. This operation keeps all the weights that were included in the previous generation but allows for them to be rearranged. This way, if the weights are good, they still exist in the population. The next operation is mutation, which can randomly replace any one of the weights in the population in order for a solution to escape local minima. Once this is complete, the generation is ready for evaluation and the process continues until the best solution is found. Figure 2.3, 2.4 and 2.5 show the process of weight optimization using GA.

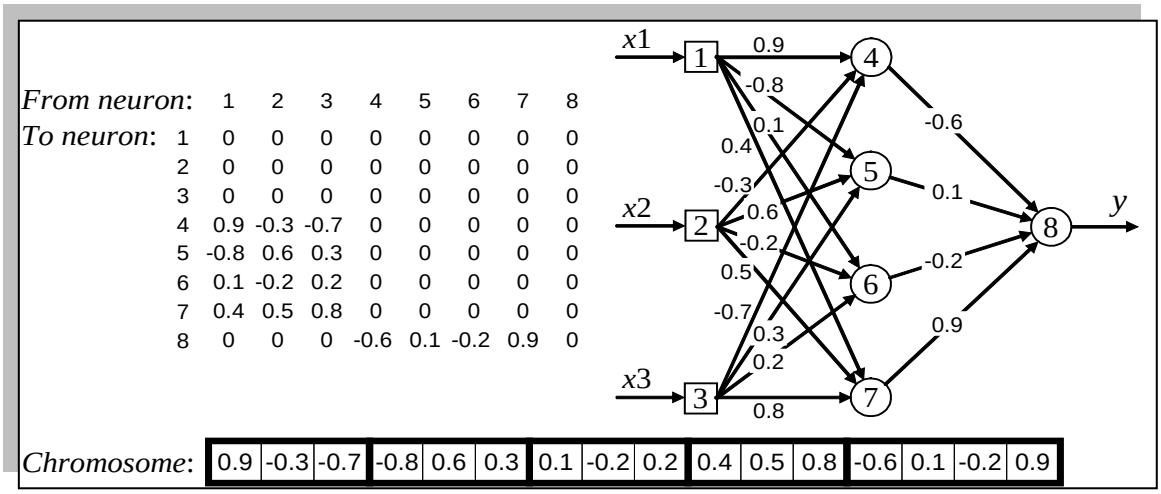

Figure 2.5: Encoding a set of weights in a chromosome

The first step, weights are encoded into chromosome format and the second step is to define a fitness function for evaluating the chromosome's performance. This function must estimate the performance of a given neural network. The function usually use is the Sum of Squared Errors (SSE). The training set of examples is presented to the network and the SSE is calculated. The smaller value of SSE means the fitter the chromosome. In NN, GA attempts to find a set of weights that minimizes the learning error. The third step is to choose the genetic operators, crossover and mutation. A crossover operator takes two parent chromosomes and creates a single child with genetic material from both parents as shown in figure 2.4.

Each gene in the child's chromosome is represented by the corresponding gene of the randomly selected parent. A mutation operator selects a gene in a chromosome and adds a small random value usually between -1 and 1 to each weight in this gene as shown in figure 2.5.

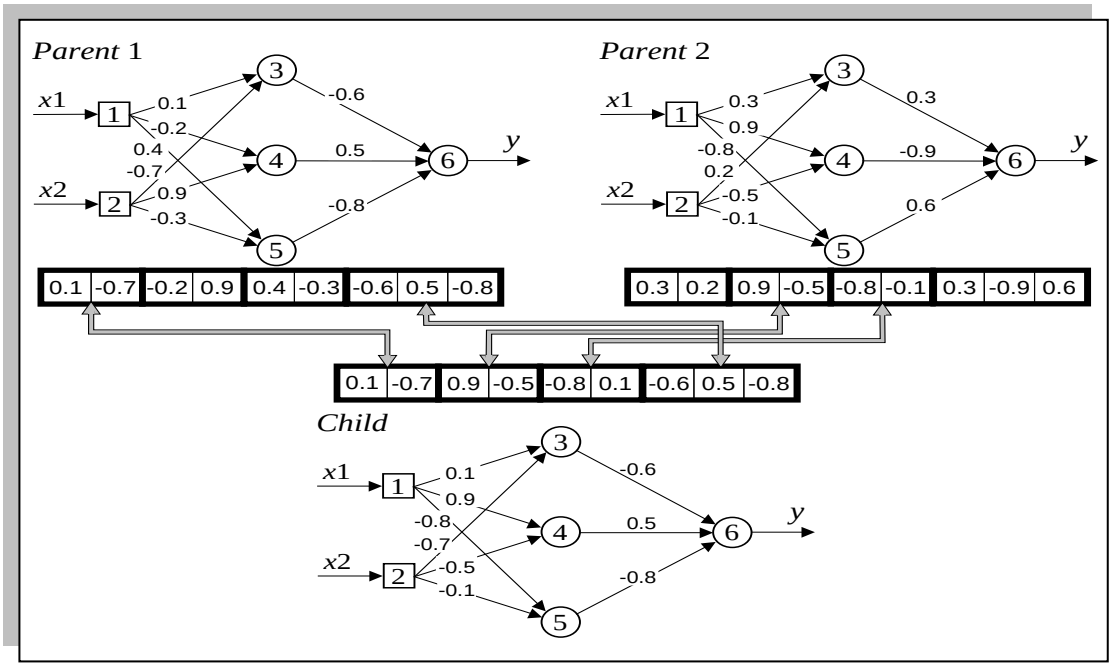

Figure 2.6: Crossover in weight optimisation

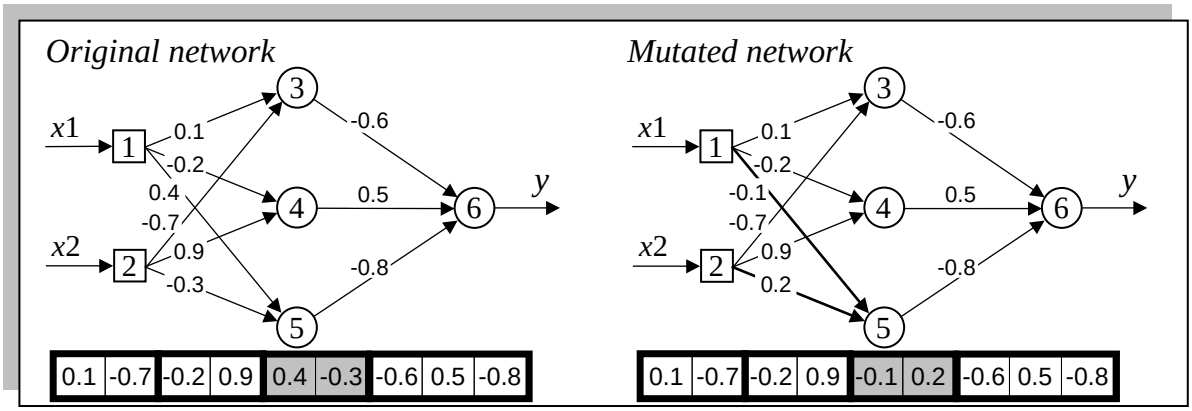

Figure 2.7: Mutation in weight optimisation

2.3.4 Differential Evolution

(Rainer & Storn,1997). DE belongs to the class of evolutionary algorithms which include Evolution Strategies (ES) and GA. It uses the steps of GA, namely, mutation, crossover and selection; see the flowchart in Figure 1.1. In the initialization, a population of NP vectors, each of dimension D (number of decision variable in the optimization problem), is randomly generated over the feasible region. Typical value of NP is about 5-10 times D to ensure DE has enough vectors to work with. The fitness of each individual is evaluated. Out of these NP vectors, one of them is randomly selected as the target vector. The DE algorithm is outlined below: Figure 2.13 shows the basic pseudo-code for the DE algorithm.

Procedure DE

Require: D – problem dimension (optional)
NP, F, Cr – control parameters
GEN – stopping condition
L, H – boundary constraints
Initialize population $Pop_{ij} \leftarrow rand_{ij}[L,H]$ and Evaluate fitness $Fit_j \leftarrow f(Pop_j)$
for $g = 1$ to GEN do
  for $j = 1$ to NP do
    Choose randomly $r_{1,2,3} \in [1,\ldots,NP]$, $r_1 \neq r_2 \neq r_3 \neq j$
    Create trial individual $X \leftarrow \mathcal{S}(r, F, Cr, Pop)$
    Verify boundary constraints if $(x_i \notin [L,H])$ $x_i \leftarrow rand_i[L,H]$
    Select better solution (X or Pop_j), and update $iBest$ if required
  end for
end for

Figure 2.8 shows the basic pseudo-code for the DE algorithm.

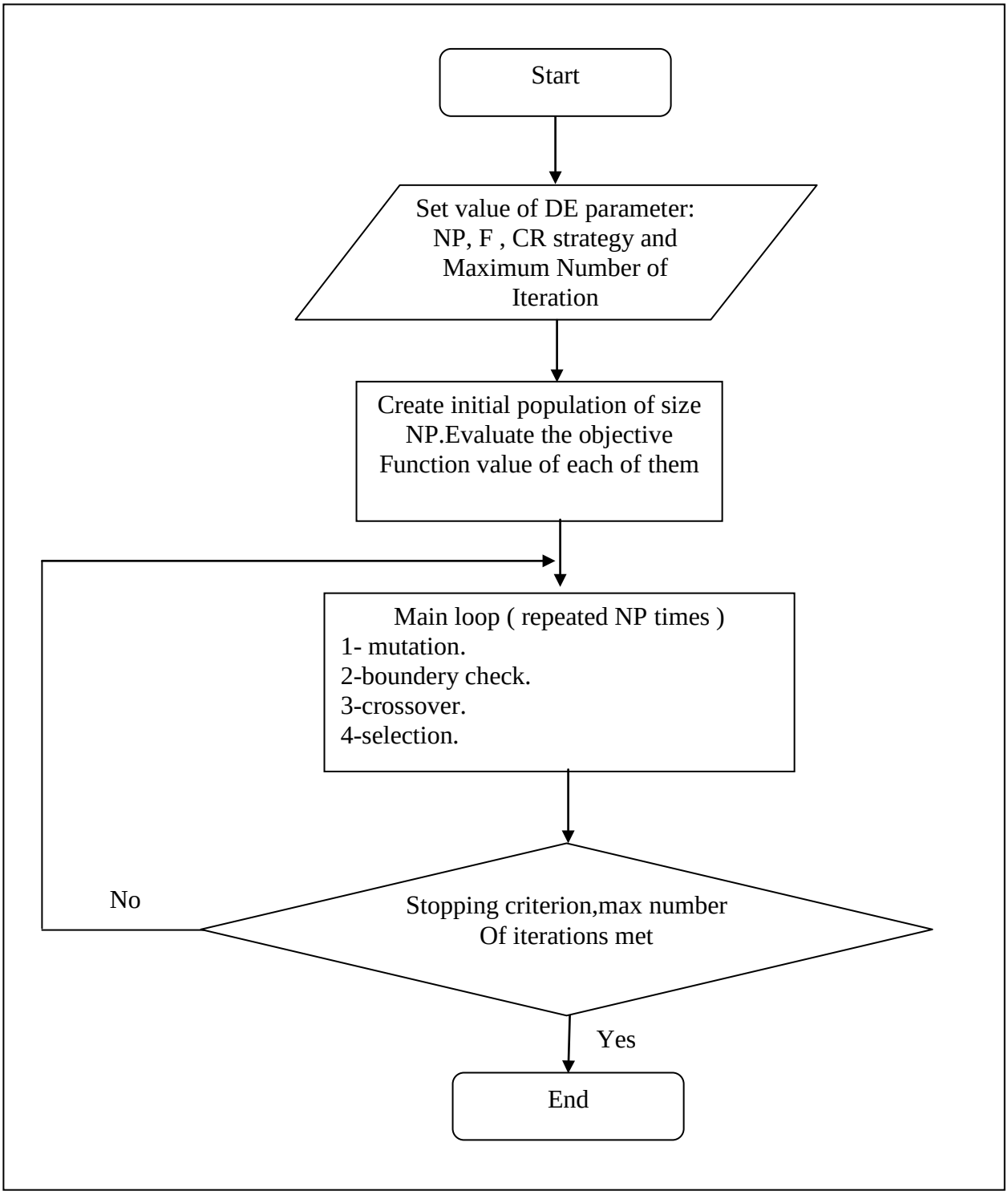

Figure 2.9: Flowchart of DE algorithm.

Differential evolution algorithm contents there are only three real control parameters in the algorithm. These are: differentiation (or mutation) constant *F*, crossover constant *Cr*, and size of population *NP*. The rest of the parameters are.

1. Dimension of problem *D* that scales the difficulty of the optimization task.
2. Maximal number of generations (or iterations *GEN*, which serves as a stopping condition in our case.
3. Low and high boundary constraints, *L* and *H*, respectively that limit the feasible area. You can vary all these parameters at will.

2.3.4.1 Properties of differential evolution

A. Very simple to programme it, the same for more dimensions.
B. Very reliable to find global extreme of coast function F(in contrast with geometrical optimization methods).
C. If function has more global extremes, differential evolution finds them.
D. can with number formats float, integer, with binary numbers and Combinations.
E. Solution of optimizations problem is one (or more) best solution

2.3.4.2 Proper algorithm

1. Definition of parameters .
2. Creation of population .
3. Reproduction loop - grade between specimens from actual generation creation of new specimens
5. Testing coast function f for test specimen.
5. Determination of new population

loop 3. – 5. is repeating until stop condition is satisfied. Best specimen is proclaimed as a result of optimization problem.

6. Interpretation of results of evolution.

2.3.4.3 The Basics of Differential Evolution

DE has three operations: Mutation, crossover and selection.

1. **Mutation.**

Fro each target vector x_{ij}^{t} , a mutant vector *v* is generated according to.

$$V_i^{t+1} = x_{r1}^t + F.(x_{r2}^t - x_{r3}^t).$$

The *r1* , *r2* and *r3* are randomly chosen indexes and *r1* ,*r2*,*r3* $\in$ { 1,2,…..,*NP*}.

F is a real number to control the amplification of the difference vector.

$$(x_{r2}^t - x_{r3}^t)$$

Range of *F* is in [0, 2] 0<F<=2

1. **Crossover.**

The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector *u*.

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1}, & rand(j) \le CR \quad or \quad j = randn(i) \\ x_{ij}^{t}, & rand(j) > CR \quad and \quad j \ne randn(i) \end{cases}$$

where *j*=1,2,…..,*D*, rand (*j*) $\in$ [0,1] is the *j*th evolution of a uniform random generator nu$\in$ r .CR $\in$ [0,1] is the crossover probability constant ,which has to be determined previously by the user rand (i) v_i^{t+1} (1, 2,….D) is a randomly chosen index which ensures that u_i^{t+1} gets at least one element from

u_i^{t+1} .Otherwise ,no new parent vector would be produced and the population would not alter

1. **Selection**

DE adapts greedy selection strategy. If and only if, the trial vector u_i^{t+1} yields a better fitness function value than x_i^t then u_i^{t+1} is set to .Otherwise , the old value is x_i^t retained.

2.4 Particle Swarm Optimization

The Particle Swarm Optimization algorithm (PSO) was originally designed by Kennedy and Eberhart in 1995, the idea was inspired by the social behavior of flocking organisms. The algorithm belongs to the broad class of stochastic optimization algorithm that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. PSO uses a population of individuals to probe promising regions of the search space. The population in this context is called a "swarm" and the individuals are called "particles". Each particle moves in the search space with a velocity that is dynamically adjusted according to its own flying experience and its companions' flying experience and retains the best position it ever encountered in memory. The best position ever encountered by all particles of the swarm is also communicated to all particles. Depending on the topology, in the local variant, each particle can be assigned to a neighborhood consisting of a predefined number of particles . (Van den Bergh *et al*., 2000). Figure 2.9 shows the basic PSO procedure.

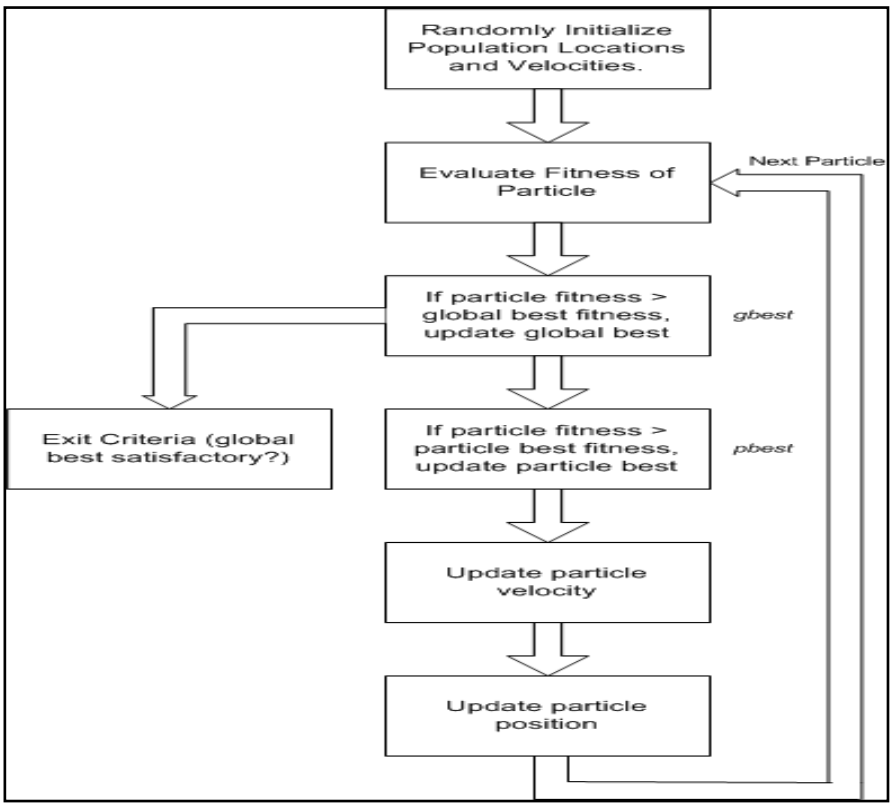

Figure 2.10: Basic PSO Procedure

2.4.1 PSO Origins

Particle Swarm Optimization has roots in two methodologies. Its links to Artificial Life in general, and to bird flocks, fish schools and swarm theory in particular are very evident. Nonetheless, PSO is also tied to Evolutionary Computation, namely to Genetic Algorithms and Evolutionary Programming.

Several scientists have created computer simulations of various interpretations of movement of organisms in a bird flock or in a fish school. Particularly, Reynolds and Heppner and Germander have presented simulations of bird flocks. Reynolds (Dickmanns et al., 1987) was intrigued with the aesthetics of flock choreography, and Heppner, a zoologist, was interested in discovering the rules which allow large flocks to move synchronously, often suddenly changing direction, scattering and regrouping, and so on. Both these scientists felt that local processes, like those modelled by cellular automata, could underlie the unpredictable group dynamics of bird social behavior. Both models were strongly based upon

manipulation of inter-individual distances, i.e. synchrony of flock behavior was thought to be a function of birds' efforts to keep an optimal distance between themselves and their neighbors.

Sociobiologist Wilson hypothesized that individual members of a fish school could take advantage of discoveries and experience gained by all other group members during search for food. He also supposed that this has a decisive advantage with respect to competition for food, if resources are distributed in an unpredictable way. This statement suggests that social sharing of information among members of a same species offers an evaluative advantage and has been fundamental to PSO development .

2.4.2 PSO Technique

The Particle Swarm Optimization algorithm (PSO) was originally designed by (Bishop, 1995), the idea was inspired by the social behavior of flocking organisms. The algorithm belongs to the broad class of stochastic optimization algorithm that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. PSO uses a population of individuals to probe promising regions of the search space. The population in this context is called a "swarm" and the individuals are called "particles". Each particle moves in the search space with a velocity that is dynamically adjusted according to its own flying experience and its companions' flying experience and retains the best position it ever encountered in memory. The best position ever encountered by all particles of the swarm is also communicated to all particles. Depending on the topology, in the local variant, each particle can be assigned to a neighborhood consisting of a predefined number of particles .

2.4.3 Original Version

The original form of particle swarm optimizer is defined as:

$$v_{id}(t+1) = v_{id}(t) + c_1 r_1 (P_{id}(t) - x_{id}(t)) + c_2 r_2 (P_{gd}(t) - x_{id}(t)) \quad (2.21)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2.22)$$

Where:

v_{id} is the velocity of particle *i* along dimension *d*

x_{id} is the position of particle *i* in *d*

c_1 is a weight applied to the cognitive learning portion

c_2 is a similar weight applied to the influence of the social learning portion

r_1 and r_2 are separately generated random number in the range of zero and one

p_{id} is the previous best location of particle *i* also known as *pbest*

p_{gd} is the best location found by the entire population, also known as the *Gbest*

Figure 2.4 shows the concept of modification of searching points described by equations (2.21) and (2.22), where vpbest and vgbest represent the terms $c_1 r_1 (p_{id}(t) - x_{id}(t))$ and $c_2 r_2 (p_{gd}(t) - x_{id}(t))$ respectively.

The PSO algorithm consists of three parts a momentum, a cognitive, and a social part. The momentum part consists of the V_{id} term and keeps the particle moving. The purpose of the momentum term is to allow the particle to fly through local minima without getting trapped. The cognitive part is expressed by the P_{id} term. Every particle in the swarm remembers its best personal explored location which is stored in its memory, term. The social term represents the information shared between all particles and is expressed in the P_{gd} term. Each particle shares its

best location with other members of the swarm, and at any given point in time, each member of the swarm knows the best location previously explored by the swarm.

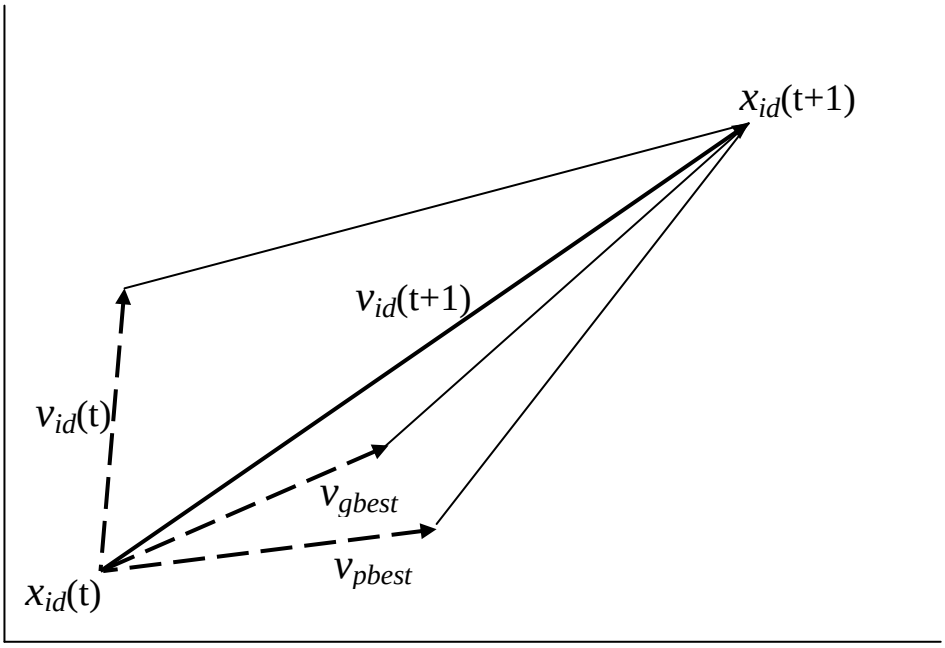

Figure 2.11 Concept of modification of searching point(PSO)

The general PSO algorithm may be applied to any optimization problem. Figure 2.11 shows the basic pseudo-code for the PSO algorithm.

```
For each particle do
    initialize particle position and velocity
end for

while stopping criteria are not fulfilled do
      for each particle do
          calculate fitness value using problem specific objective function
          if fitness value is better than best fitness value pbest in particle
             history  then
            set current position as pbest
          end if
      end for
      choose as gbest the particle with best fitness value among all particles in
      current iteration
      for each particle do
          calculate particle velocity based on Eq. (2.21)
          update particle position based on Eq. (2.22)
     end for
end while
```

Figure 2.12 PSO algorithm

2.3.4 Comparison GA, DE and PSO.

Most of evolutionary techniques (Including GA) use the following procedure:

1. Random generation of an initial population.
2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to step b.

DE algorithm has many advantages, such as faster convergence speed, stronger Stability, easy to realize and so on, so it is noticed by many researchers. But the individuals of the basic DE algorithm is random during the period of evolutionary procedures and it is prone to cause unsteady easily Basic DE algorithm is a kind of evolutionary algorithm, which is used to optimize the minima of functions, and its code is based on real number, the whole structure is similar to the GA, and the main difference between standard GA and DE is mutation operation The mutation is a main operation of DE, and it revises each individual's value according to the difference vector of the population. Its basic idea lies in applying the difference of current population individual to reorganize to obtain the middle population, then, using the direct offspring and parents individual fitness value competition to get the new generation.

From the procedure, it shows that PSO shares many common points with GA.all algorithms is used for optimization, PSO similar to GA where both start with a population of random solutions (Eberhart *et al*., 2001). Both have fitness values to evaluate the population. Both update the population and search for the optimum with random techniques and both algorithms do not guarantee success. However unlikeGA and other evolutionary computation techniques, each particle in PSO is also associated with a velocity. Particles fly through the search space with velocities which are dynamically adjusted according to their historical behaviors. Therefore,

the particles have a tendency to fly towards the better and better search area over the course of search process. (Shi, 2004)**.** PSO also does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm. Compared with GA, the information sharing mechanism in PSO is significantly different. In GA, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gbest gives out the information to others. It is a one way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution.

2.6 Applications Artificial Neural Network and Differential evolution.

2.61. Applications of Artificial Neural Network (ANN)

ANN is soft computing technique that is widely used in many applications especially in intelligence application/system, prediction and classification. This is because of its effectiveness and the successful is proven in many applications. Below are some ANN-based applications:

1. Flight Control Using an Artificial Neural Network, University of Queensland and CSIRO Manufacturing Science & Technology that was implemented in Helicopter Unmanned Ariel Vehicle (UAV) where ANN was used to generate hover command, which are used to directly manipulate the flight. (Wyeth ,G *et al*., 2002).

2. Profitability Analysis in Malaysian Market by (Siti Mariyam Hj. Shamsuddin, Saiful Hafizah Jaaman, Noriza Majid & Noriszura Ismail 2004) where neural network model has been introduced with an improved backpropagation error function for predicting profitability of selected firms at Kuala Lumpur Stock Exchange (KLSE).

3. Consumer choice prediction by Gan from Commerce Division, Lincoln University, New Zealand. This is to understand and to accurately predict the consumer decision can lead to more effectively target the products (and/or services), cost effectiveness in marketing strategies, increasing in sale and result in substantial improvement in the overall profitability of the firm. (Limsombunchai, M *et al*., 2000).

2.6.2 Applications of Differential evolution (DE)

As mentioned before,DE was introduced by (Storn & Pricein 1997).However in 1995, nowadays this concept has been explored by many otherresearchers around the globe and has been applied in many applications. Below are some application examples using DE for optimization:

The practical efficiency of differential evolution, there are present two famous problems from different fields of application. The first one is a classical identification problem in which DE is used to evaluate the parameters of a Choquet integral. This problem is a striking example from the decision-making domain. The second one belongs to the engineering design domain, the so-called bump problem. DE is applied here to find the global optimum, which is placed exactly on a constraint function. It presents the real challenge of this field. As you will see, differential evolution demonstrates prominent results on both cases in comparison with the methods that were used previously.
(Feoktistov, 2006 **).**

1. Application of Differential Evolution to Training Algorithm for Feed-Forward Neural Networks absorbers by Jarmo Ilonen,Jon I-kristian Kamarainen and Jouni Lampinen Dept. of Electrical & Computer Engineering, University of Delaware. The synchronous DE was applied to optimize multilayer coatings and polygonal sorbers for wide band frequency and/or wide incident range. (Storn & Pricein 1997).

2. Engineering Design with Differential Evolution.

The bump problem is a well-known benchmark. It was first introduced by A. (Feoktistov, 2006).Many research efforts have been directed towards its solving . The problem belongs to a class of multipeak problems that is typical for engineering design. They are hard problems. Its author's description can be found on. (Feoktistov,2006 **).**

3. Decision Making with Differential Evolution

Differential evolution algorithm to solve a classical identification problem. The task is to learn the behavior of a decision makers' group. The aggregation operator is based on the family of Choquet integrals and represents a two-additive measure. The goal is to use differential evolution (Feoktistov, 2006 **).**

4. Recent Applications.

Design of digital filters. Optimization of strategies for heckers.Maximisation of profit in a model of a beef property. Optimization of fermentation of alcohol. (Feoktistov,2006 **).**

2.7 Summary

This chapter describes the concept of ANN, EC and DE algorithms. But it seems that they have several problems in convergence speed and being trapped to local minima. To overcome this problem, this chapter has explained a technique called DE.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter describes the methodology for this study. The first part explains the presentation of dataset used in the experiment, and the second part discusses the DENN, PSONN and GANN architecture, follows by the experiment and analysis of the study.

3.2 A Framework of the Study

In order to implement DE in neural network and compare the performance with PSO and GANN , the following steps have been followed:

1. Determine training pattern from datasets.
2. Define neural network architecture.
3. Determine network parameters.
4. Run DENN.
5. Run GANN.

6. Run PSODE
7. Comparison and analysis.

The general framework for this study is shown in Figure 3.1.

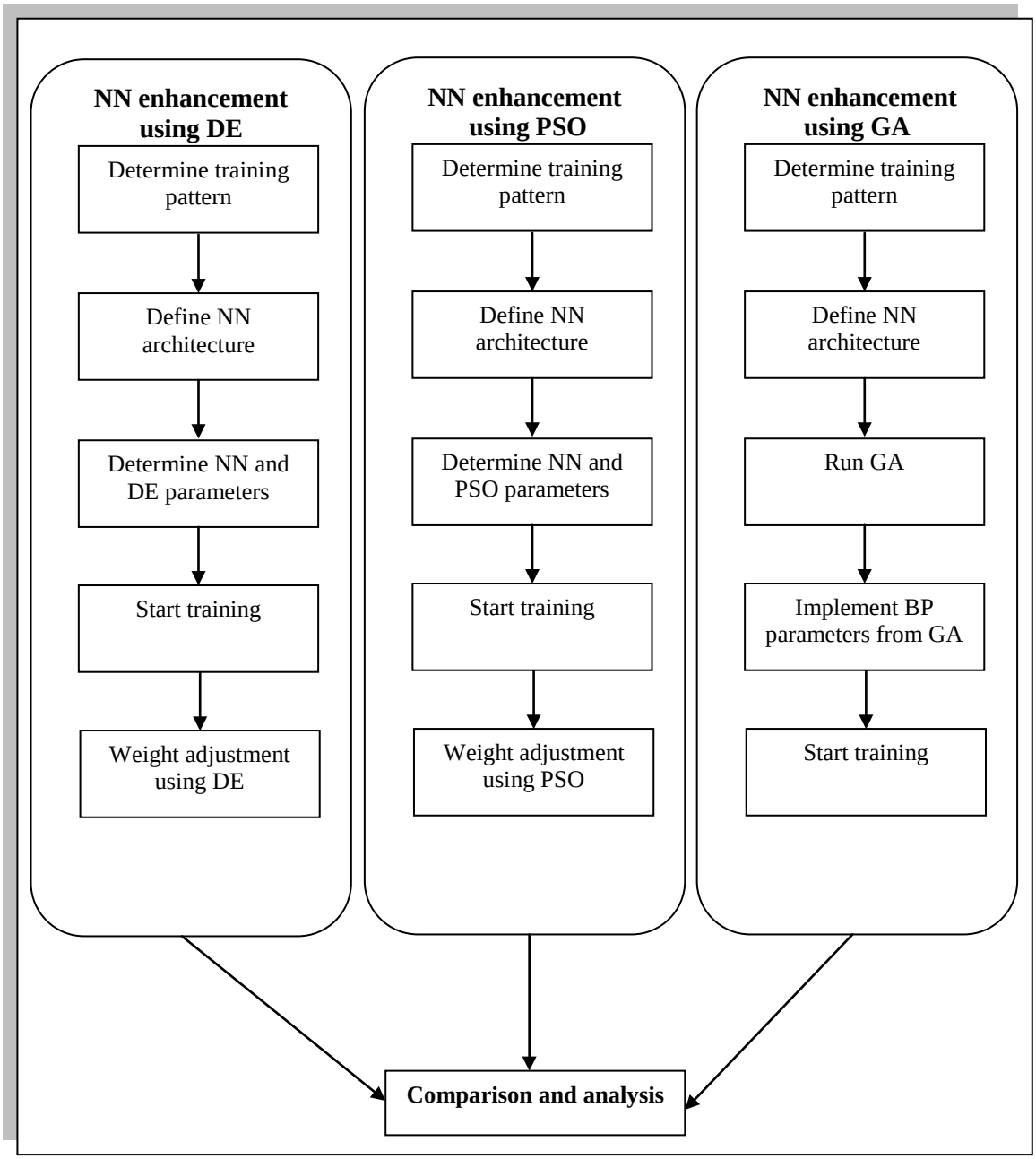

Figure 3.1: Framework of the study

3.3 Data Preparation

The dataset is required to represent the problem. In this study, universal machine learning data has been used for testing and training the network, and these include XOR, Cancer , Iris and Heart .

1. XOR

A connective in logic known as the "exclusive or" or exclusive disjunction is a logical operation on two operands that results in a logical value of true if and only if one of the operands but not both has a value of true. XOR is a basic dataset that widely use to train and test NN. In this study, 4 data patterns are used in both algorithms.

2. Cancer

The Cancer dataset requires the decision maker to correctly diagnose breast lumps as either benign or malignant based on data from automated microscopic examination of cells collected by needle aspiration. The dataset includes nine inputs and one output. The exemplars are split with 599 for training, and 100 for testing, totaling 699 exemplars. All inputs are continuous variables and 65.5% of the examples are benign. The data set was originally generated at hospitals at the University of Wisconsin Madison, by Dr. William H. Wolberg. In this study, 150 data patterns are used in both algorithms.

3. Iris

The Iris dataset is used for classifying all the information into three classes which are iris setosa, iris versicolor, and iris virginica. The classification is based on its four input pattern which are sepal length, sepal width, petal length and petal width. Each class refers to type of iris plant contain 50 instances. In NN learning, the network has four input patterns and 3 output patterns. In this study, 120 data patterns are used in all algorithms.

4. Heart

heart data for binary data for the same classification task. The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal.

The classification is based on its twenty two input pattern In NN learning, the network has twenty two input patterns and one output patterns. In this study, 80 data patterns are used in all algorithms.

3.4 Neural Network Structure for DENN, PSONN and GANN.

When DE was prompted, its potential performance on weights modification of neural network had been detained and was thought to be an alternative to BP methods because of its convenience (Song *et al*., 2004). In this study, DE is applied with feedforward neural network compared to PSO and GA which applied to BP network. For both algorithms, 3-layer ANN is used for classification on this study for all datasets. The network architecture consists of input layer, hidden layer and output layer. The total number of nodes for every layer is different depending on the classification problem. Number of input layer and output layer usually come from number of attribute and class attribute. However there is no appropriate standard rule

or theory to determine the optimal number of hidden nodes (Kim *et al*., 2004). There are many suggestions by researcher to determine the suitable number of hidden node. Some suggested techniques are summarized as follows:

1. The number of hidden nodes should be in the range between the size of the input layer and the size of the output layer.
2. The number of hidden nodes should be 2/3 of the input layer size, plus the size of the output layer.
3. The number of hidden nodes should be less than twice the input layer size.
4. Pyramidal shape topology (Siti Mariyam Hj Shamsuddin, 2004).
5. Kolmogorov theorem: One hidden layer and 2N+1 hidden neurons sufficient for N inputs (Siti Mariyam Hj Shamsuddin, 2004).
6. The number of hidden nodes is selected either arbitrarily or based on trial and error approaches (Charytoniuk & Chen, 2000).
7. The number of hidden nodes should be nm* where m is the number of input nodes and n is number output nodes (Charytoniuk & Chen, 2000).

In this study, Kolmogorov theorem has been used to determine number of hidden node and the activation function used to calculate output for each neuron except input neuron is Sigmoid Activation/Transfer Function Equation (3a).

$$f(x) = \frac{1}{(1+e^{-x})} \qquad \text{Where x = input} \qquad (3a)$$

Figure 3.2 shows neural network architecture for XOR dataset which consist of 2 input layers, 1 output layer and 5 hidden layers based on Kolmogorov Theorem. The neural network architecture for Cancer is shown in figure 3.3 which consist of 9 input nodes, 19 hidden nodes and 1 output node. For Iris dataset neural network architecture, it consists of 4 input nodes, 9 hidden nodes and 3 output nodes (Figure 3.4).
For Heart .Dataset neural network architecture, it consists of 22 input nodes, 45 hidden nodes and 1 output nodes (Figure 3.5).

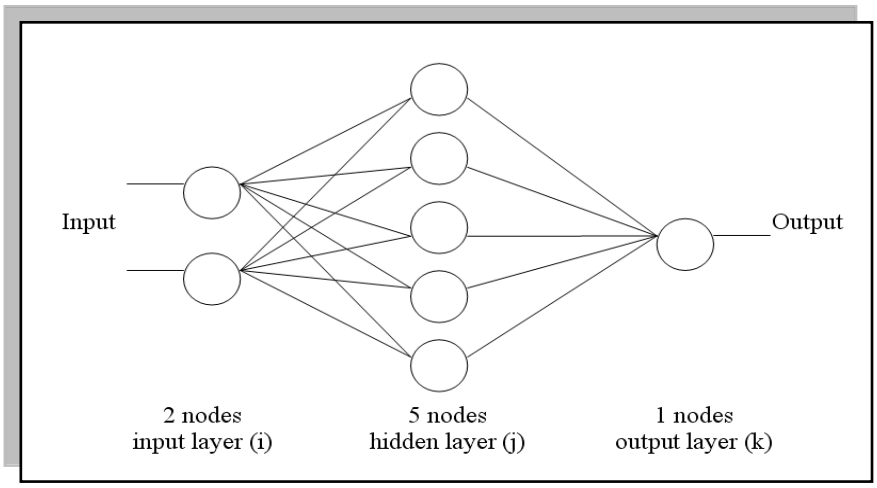

Figure 3.2: Neural Network Architecture for XOR

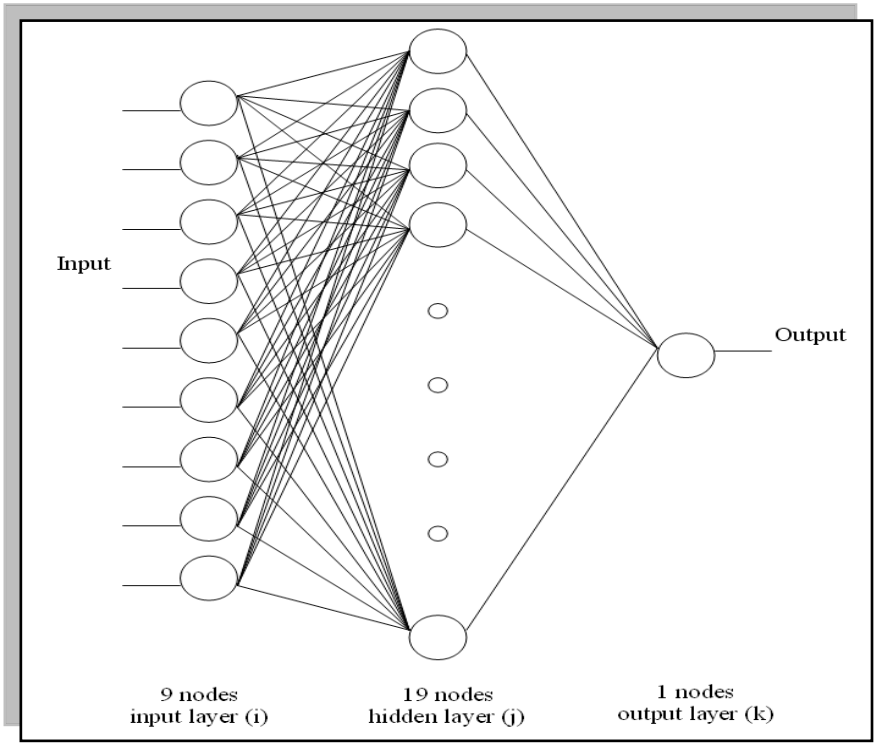

Figure 3.3: Neural Network Architecture for Cancer

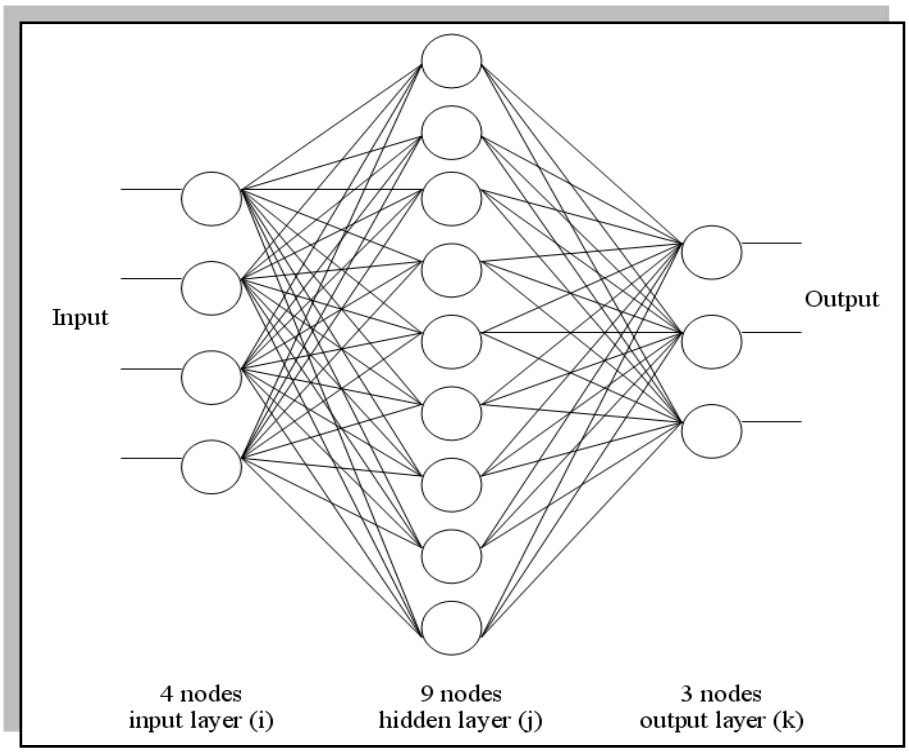

Figure 3.4: Neural Network Architecture for Iris

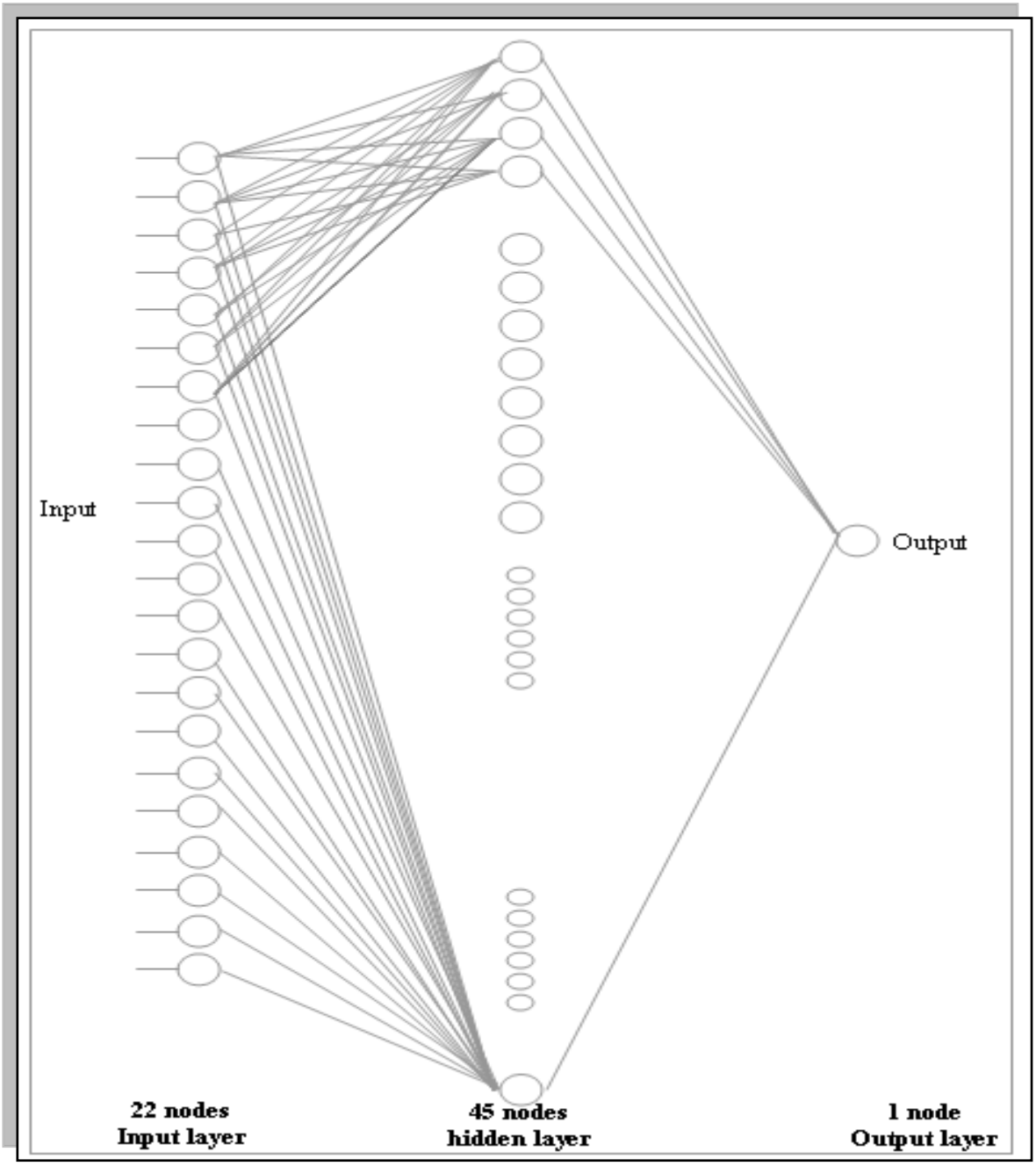

Figure 3.5: Neural Network Architecture for Heart

3.5 Differential Evolution Training Algorithm

Differential evolution can be classified as a floating-point encoded evolutionary algorithm for global optimization over continuous spaces. As such, it can be applied to global searches within the weight space of a typical feed-forward neural network. Output of a feed-forward neural network is a function of synaptic weights W and input values x, i.e. , $y = f(x, W)$. In standard training processes, both the input vector x and the output vector y are known. The synaptic weights in W are adapted to obtain appropriate functional mappings from the input x to the output y. Generally, the adaptation can be carried out by minimizing the network error function E which is of the form

$$E(y, f(x, W)) : (y D1, x D2, x D3, f) \rightarrow R. \quad (1)$$

The optimization goal is to minimize the objective function $E(y, f(x, W))$ by optimizing the values of the network weights (now D = D3)

$$W=(W1,\ldots\ldots\ldots\ldots WD) \quad (2)$$

Similar to other evolutionary algorithms, DE operates on a population, P_G,of candidate solutions ,not just a single solution. These candidate solutions are the individuals of the population. DE maintains a population of constant size that consists of NP,real- value vectors, $W_{i,G}$, wher is i index to the population and G is the generation to which the population belongs.

$$P_G=(W_{1,G},\ldots\ldots,W_{NP,G}), \quad G=0,\ldots\ldots\ldots,Gmax \quad (3)$$

Additionally, in network training each vector contains D network weights (chromosomes of individuals):

$$W_{i,G} = (w_{1,i,G},\ldots\ldots\ldots w_{D,i,G}), \quad i =1,\ldots\ldots NP , G = 0,\ldots\ldots\ldots..G_{max} \quad (4)$$

DE's self-referential population reproduction scheme is different from other evolutionary algorithms. After initialization of the first population, vector s in the current population, P_{G+1},are randomly sampled and combined to create candidate vectors for the subsequent generation, PG+1.The population of candidates, trial vectors PG+1= Ui,G+1= uj,i,G+1 is generated as follows:

$$uj,i,G+1 = w\,j,r3,G + F\,.\,(wj,r1,G - wj,r2,G)$$

$$uj,i,G+1 = \begin{cases} uj,i,G+1 , & \text{if } \text{rand}\,j\,[\,0,\,1\,] \le CR \\ w\,j,i,G , & \text{otherwise} \end{cases} \qquad (5)$$

Where

$i=1,\ldots.NP\,, \quad j = 1,\ldots\ldots D$

$r1, r2, r3 \in \{1\ldots\ldots..NP\}$, randomly selected except $r1 \neq r2 \neq r3 \neq i$

$CR \in [0, 1], F \in (0, 1)$

CR is a real-valued crossover factor that controls the probability that a trial vector parameters will come from the randomly chosen, mutat ed vector $u_j,i,_{G+1}$ instead of the current vector $w_j,i,_G$. In general, F and CR affect the convergence speed and robustness of the search process. Their optimal values depend both on objective function characteristics and the population size NP,and thus, the selection of optimal parameter values is an application dependent task.

DE's selection scheme also differs from other evolutionary algorithms. The population for the next generation, P_{G+1},is selected from the current population P_G or the child population according to the following rule

$$W_{i,G+1} = \begin{cases} U_{i,G+1} , & \text{if } E(y, f(x, W_{i,G+1})) \le E(y, f(x, W_{i,G})) \\ W_{i,G} & \text{otherwise} \end{cases}$$

Thus, each individual of the temporary population is compared to its counterpart in the current population. Assuming that the objective function is to be minimized, the vector with the lower objective function value wins a place in the next generation's population. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. The interesting point concerning DE's replacement scheme is that a trial vector is only compared to one individual, not to all individuals in the current population. It is also noteworthy that the replacement scheme ensures that the population does not diverge

from or lose the best solution found so far.

Parameters that have been used in this study are shown in Table 3.1, Differentiation constant F is set to 0.9 and crossover constant Cr is set to 0.6 . The size of population NP equals to 60, the problem dimension D is based on NN architecture, generations is set to 100 (GEN =100), Low constraints (L) and high boundary constraints (H), the stopping condition is set to NN minimum error or maximum number of iterations (Eberhart et al, 2003).

Table 3.1: DE parameters

PARAMETER	VALUE
Differentiation(or mutation) constant *F*	0.9
crossover constant *Cr*	0.5
size of population *NP*	60
Dimension of problem *D*	Based on dataset NN architecture
Maximal number of generations (or iterations *GEN*)	100
Low and high boundary constraints, *L* and *H*	Based on dataset NN architecture
Stop condition	NN minimum error or maximum number of iteration

The pseudo code of the procedure is as follow:

```
Require: D – problem dimension (optional)
         NP, F, Cr – control parameters
         GEN – stopping condition
         L, H – boundary constraints
Initialize population Pop_ij ← rand_ij[L, H] and Evaluate fitness Fit_j ← f(Pop_j)
for g = 1 to GEN do
  for j = 1 to NP do
    Choose randomly r_1,2,3 ∈ [1, ..., NP], r_1 ≠ r_2 ≠ r_3 ≠ j
    Create trial individual X ← S(r, F, Cr, Pop)
    Verify boundary constraints if (x_i ∉ [L, H])  x_i ← rand_i[L, H]
    Select better solution (X or Pop_j), and update iBest if required
  end for
end for
```

Figure 3.6: DENN procedure

3.6 PSO Parameters

Jones M.T. (2005) explained that there are four basic parameters in PSO:

1. The acceleration constants for gbest (C_1).
2. The acceleration constants for pbest (C_2).
3. The time interval (Δt).
4. The number of particle.

The acceleration constants are used to define how particles swarm in the simulation. According to Eberhart *et al.* (2001), the acceleration constant C_1 and C_2 represent the stochastic acceleration that pulls each particle toward pbest and gbest position. The C_1 constant affects the influence of the global best solution over the particle, whereas the C_2 constant affects how much influence of personal best

solution has over the particle. When C_1 is higher then C_2, the swarm tends to flow more around the global best solution, whereas the inverse causes greater flow around the individual personal best solution.

The *dt* (or Δt) parameters defines the time interval over which movement takes place in the solution space. Decreasing these parameters provides higher granularity movement within the solution space, and higher Δt performs lower granularity movement (greater distance achieved in less time). For a number of particles in the simulation or swarm, the more particles that are presented, the greater the amount of space that is covered in the problem, thus optimization becomes slower.

Depending on the solution space, this parameter can be adjusted to achieve better optimization. Besides these basic parameters, there are also some other parameters that depend on the problems such as particle dimension, number of particles and stopping condition. In PSONN, number of dimension is referring to number of weight and bias that is based on the dataset and ANN architecture. Equation 3b shows how PSONN dimension is calculated in this study. The number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (fewer particles) must be sought. (Van den Bergh *et al*., 2000). PSO with well-selected parameter set can have good performance (Shi, 2004).

$$Dimension = (input * hidden) + (hidden * output) + hidden_{bias} + output_{bias} \quad (3b)$$

Parameters that have been used in this study are shown in Table 3.1, while particle position (weight and bias) values are initialized randomly with initial position velocity value is set to 0 (as advised via email by Prof A.P. Engelbrecht, University of Pretoria, South Africa). The C_1 and C_2 constant are set to 2 as suggested by Eberhart *et al.* (2001).

Table 3.2: PSO parameters

PARAMETER	VALUE
C_1	2.0
C_2	2.0
Δt	0.1
Number of particles	20
Problem Dimension	Based on dataset NN architecture
Range of particles	Not specified. Free to move anywhere
Stop condition	NN minimum error or maximum number of iteration

The pseudo code of the procedure is as follows:

```
For each particle
    Initialize particle for NN problem
End

Do
    For each particle
        Calculate fitness value (feedforward error or MSE in NN)
        If the fitness value is better than the best fitness value
        (pBest) in history
            Then set current value as the new pBest
End

Choose the particle with the best fitness value of all the particles as the
gBest

For each particle
        Calculate particle velocity
        Update particle position (NN Weight)
End
```

Figure 3.7: PSONN procedure

3.7 GA-based Neural Network

As mentioned in Chapter 2, GA is used as parameter tuning and weight optimization to enhance the BP learning. In this study, GA provides a set of best weight and parameters to NN. Weight in BP is replaced with weight from GA, while η and α (learning and momentum rate) value in BP are replaced with value from GA process. GA algorithm is proven to be effective to guide the ANN learning, thus, it is widely used in many real world applications. As a result, DE technique is proposed to see the performance and the results are compared with GANN learning performance. Table 3.2 shows the parameters for GA that have been used in this study.

Table 3.3: GA parameters

PARAMETER	VALUE
Maximum population	100
Population size	50
Maximum generation	100
Crossover rate	0.6
Mutation rate	0.02
Reproduction rate	0.6

3.8 Experiment and Analysis

In this study, DE has been implemented in feedforward neural network. This is to prove that the convergence rate is faster and the classification result is better in most cases.. The following steps are used as guidance throughout this study:

1. Construct DENN based on DE program for function optimization.
2. Construct PSONN and GA.
3. Comparison between DENN ,PSONN and GANN.
4. Evaluate performance of (2) and (3).

In order to achieve the objective of this study, the experiment is performed in two parts. The first part is an experiment on neural network using DE to help the training process, and the second part is on neural network using PSO to get the best parameters. The results are compared on several performances such as convergence time and the accuracy for the classification process.

3.9 Summary

This chapter discussed the methodology that is implemented in this study. The algorithms for DENN ,PSONN and GANN are executed and compared to seek the effectiveness of the solutions. The programs are coded using Microsoft Visual C++ 6.0 and Matlab is employed to visualize the behavior of the parameters accordingly.

CHAPTER 4

EXPERIMENTAL RESULT

This chapter discusses the experimental results of the study. Three programs have been developed: Differential Evolution Forward Feed Neural Network (DENN), Particle Swarm Optimization Feed Forward Neural Network (PSONN) and Genetic Algorithm Feed Forward Neural Network (GANN) using four dataset: XOR, Cancer , heart and Iris. The results for each dataset are compared and analyzed based on the convergence rate and classification performance.

4.1 Results on XOR Dataset

The network size that has been used to train the XOR problem consists of 2 input nodes, 5 hidden nodes and 1 output node, and the total number of patterns is 4 (refer to Appendix A). For DE parameters, differentiation (or mutation) constant is set as F=0.8, crossover constant *Cr =0.* , size of population NP=60, dimension of problem *D*=20 , low and high boundary constraints, *L=0* and *H* =1 , problem dimension (total number of weight and bias) = 21 and the stopping condition is minimum error of 0.005 , or maximal number of generations (or iterations) *GEN* =10000 . For PSO parameters, 20 particles are used, C1 and C2 = 2, t = 0.1, and the stopping condition is set to 0.005 or maximum iteration of 10000. The stopping condition for GANN is a minimum error of 0.005 or the iterations have reached

10000. Table 4.1 and Figure 4.1 illustrate the experimental results of DENN , PSONN and GANN.

From Table 4.1, the results illustrate that DENN converges at 7 seconds with 43 iterations compared to PSONN with 12 seconds at iteration of 51. GANN takes 37 seconds for overall learning process to converge (Figure 4.1). All the algorithms are converged based on the minimum error criteria. For the correct classification percentage, it shows that DENN is better compared to PSONN and GANN with an accuracy of 98.97%, PSONN yields 95.17%, and GANN produces 85.66%.

Table 4.1: Result of DENN , PSONN and GANN on XOR dataset

	DENN	PONN	GANN
Learning Iteration	41	51	61
Error Convergence	0.00488652	0.00473763	0.04125
Convergence Time	7 Sec	12 Sec	37 Sec
Classification (%)	98.97	95.17	85.66

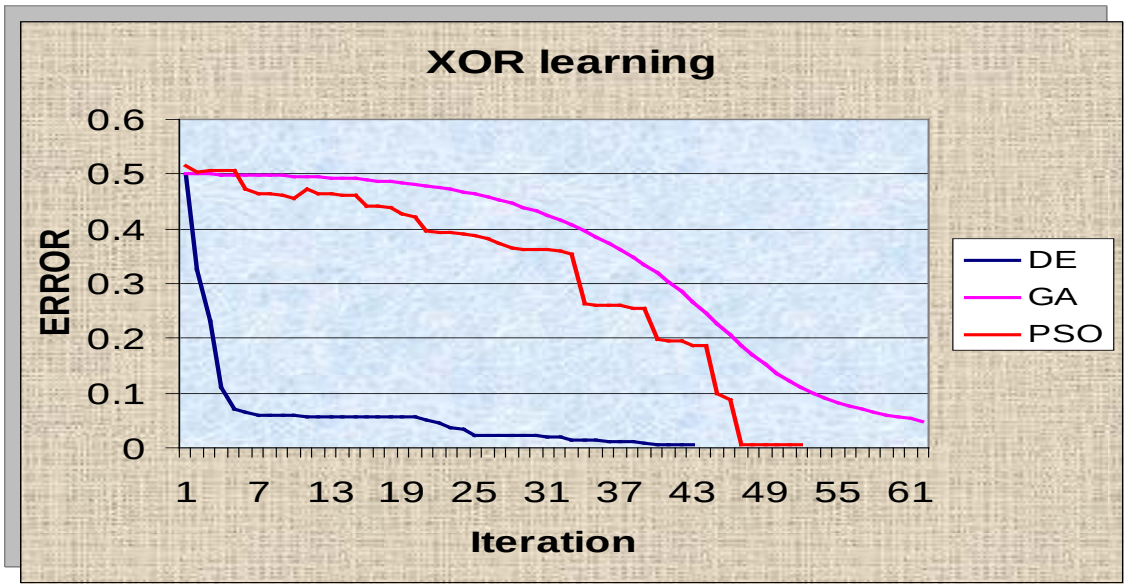

Figure 4.1: Convergence of XOR dataset

From Table 4.1, the result shows that DENN convergence time is 7 seconds at iteration 43 compared to , PSONN convergence time is 12 seconds at iteration 51,GANN where it takes 37 seconds for overall learning process. All algorithms are

converged using the minimum error criteria. For the correct classification percentage, it shows that DENN result is better than compared to PSONN and GANN, DENN=98.97% , PSONN =95.17% ,GANN= 85.66%. Figure 4.1 shows the learning process

4.2 Results on Cancer Dataset

For Cancer problems, 150 data patterns have been used where the network size consists of 9 nodes in the input layer, 19 nodes in the hidden layer and 1 node in the output layer. Cancer data pattern that has been used in this experiment is shown in Appendix B.For DE parameters differentiation (or mutation)constant F=0.9 crossover constant *Cr =0.7* , size of population NP=60, dimension of problem *D* =210 , low and high boundary constraints, *L=0* and *H* =0.9 , problem dimension (total number of weight and bias) = 210 and the stop conditions are minimum error 0.005 , or maximal number of generations (or iterations) *GEN* =10000 .

For PSO parameters, 20 particles are used, C_1 and C_2 = 2, ?t = 0.1, problem dimension (total number of weight and bias) = 210 and the stop conditions are minimum error 0.005 or maximum iteration of 10000. The stop conditions for GANN are minimum error 0.005 or reach maximum iteration of 10000. The experimental results for DE-based NN , PSO-based NN and GA-based NN are shown in Table 4.2 and Figure 4.2 and Figure 4.3.

Table 4.2: Result of DENN , PSONN and GANN on Cancer dataset

	DENN	PSONN	GANN
Learning Iteration	443	219	10000
Error Convergence	0.004999	0.00487004	0.50049
Convergence Time	195 Sec	110	273 Sec
Classification (%)	98.40	98.65	97.73

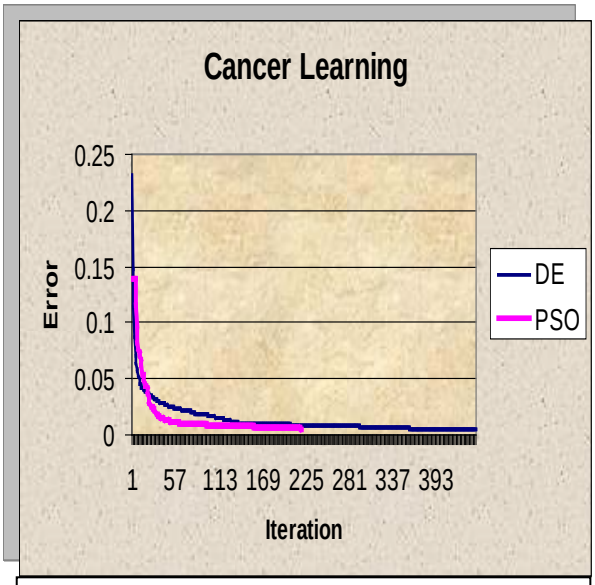

Figure 4.2: DE,PSO Convergence of Cancer dataset

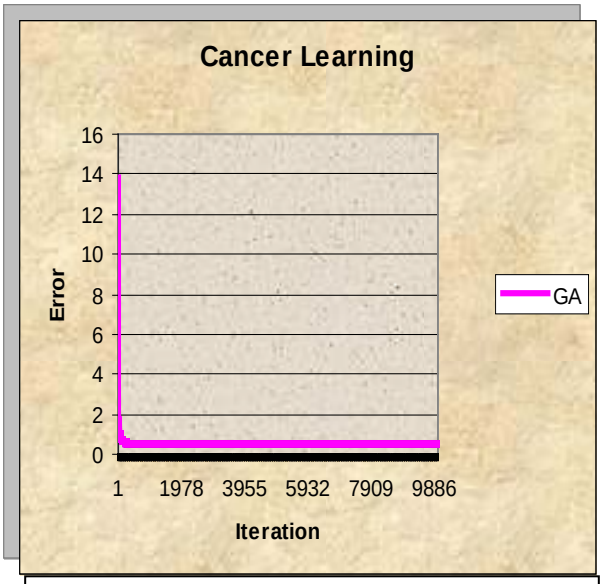

Figure 4.3: GA Convergence of Cancer dataset

In Cancer learning process DENN take 95 seconds compared to PSONN takes 110 seconds and 273 second in GANN to converge as shown in Table 4.2. In this experiment, DE is managed to converge using error at iteration 443, while PSONN converge at a using minimum error at iteration 219 and GANN converge at a maximum iteration of 10000. For the correct classification percentage, it shows that PSONN result is better than GANN and DE with 98.65% compared to GA =97.73% and DE=98.40. Figure 4.2 shows PSONN significantly reduce the error at small number of iteration compared to DENN and GANN.

4.3 Results on Iris Dataset

The network architecture used for Iris dataset consists of 4 input nodes, 9 hidden nodes and 3 output nodes. 120 data patterns used to train the network. Iris data pattern that has been used in this experiment as shown in Appendix C. For DE parameters differentiation (or mutation)constant F=0.8 crossover constant $Cr = 0.5$, size of population NP=60, dimension of problem D =75 , low and high boundary constraints, $L=0$ and H =1 , in Table 4.3 and Figure 4.3.

problem dimension (total number of weight and bias) = 75 and the stop conditions are minimum error 0.05 , or maximal number of generations (or iterations) *GEN* =10000 . For PSO parameters, 20 particles are used, C_1 and C_2 = 2, ?t = 0.1, problem dimension (total number of weight and bias) = 75 and the stop conditions are minimum error 0.05 or maximum iteration of 10000. The stop conditions for GANN are minimum error 0.05 or reach maximum iteration of 10000. The experimental results are shown in Table 4.3 and Figure 4.4 DE , Figures 4.5, PSONN, Figures 4.6 GA.

Table 4.3: Result of DENN, PSONN and GANN on Iris dataset

	DENN	PSONN	GANN
Learning Iteration	61	818	10000
Error Convergence	0.049803	0.049994	1.88831
Convergence Time	16 Sec	170 Sec	256 Sec
Classification (%)	95.014972	93.86	97.72

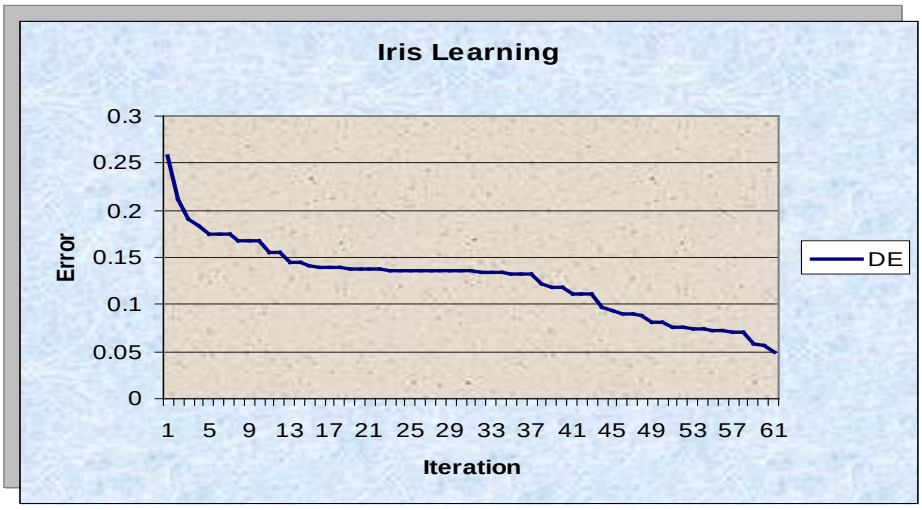

Figure 4.4: DE Convergence of Iris dataset

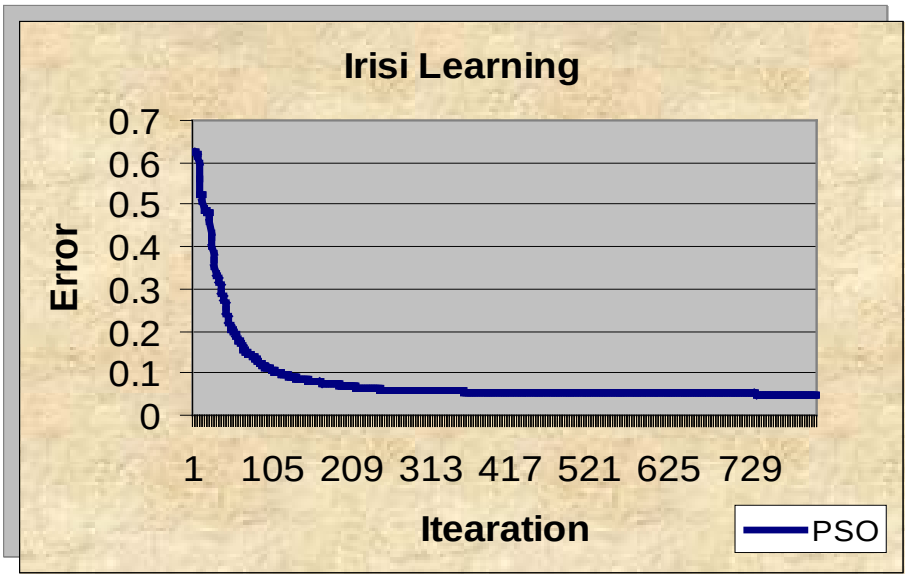

Figure 4.5: PSO Convergence of Iris dataset

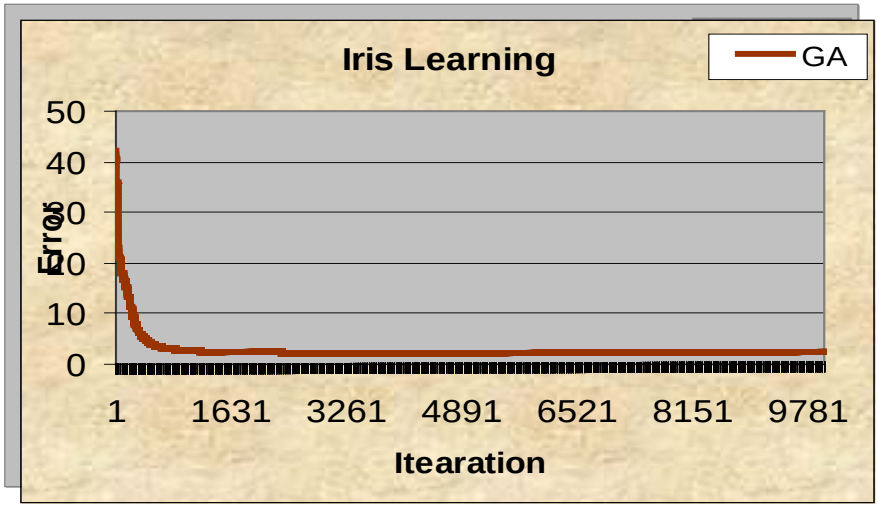

Figure 4.6: GA Convergence of Iris dataset

For Iris learning, all algorithms converge using the maximum number of pre-specified iteration. DENN takes 16 second to converge at minimum error of 0.049803 at 61 iterations .PSONN takes 170 second to converge at minimum error of 0.00487 while minimum error for GANN is 1.88831 at 10000 iterations. shows that GANN result is better than compared to PSONN and DENN, GANN= 97.72 % , DENN=95.014972%, PSONN=93.86% However, DENN convergence is faster at 61 iteration. Compared PSONN convergence is at 818 iteration and 10000 iteration in

GANN.DENN , PSONN significantly reduces the error with minimum iterations compared to GANN (Figure 4.3).

4.4 Results on Heart Dataset

The network architecture used for Heart dataset consists of 22 input nodes, 45 hidden nodes and 1 output nodes. 80 data patterns used to train the network. Heart data pattern that has been used in this experiment as shown in Appendix D. For DE parameters differentiation (or mutation)constant F=0.8 crossover constant $Cr = 0.5$, size of population NP=60, dimension of problem D =10 , low and high boundary constraints, $L=0$ and H =1 , problem dimension (total number of weight and bias) = 1081 and the stop conditions are minimum error 0.05 , or maximal number of generations (or iterations) GEN =10000 .

For PSO parameters, 20 particles are used, C_1 and C_2 = 2, ?t = 0.1, problem dimension (total number of weight and bias) = 1081 and the stop conditions are minimum error 0.05 or maximum iteration of 10000. The stop conditions for GANN are minimum error 0.05 or reach maximum iteration of 10000. The experimental results are shown in Table 4.4 and Figure 4.7 DE , Figures 4.8, PSONN, GA .

Table 4.4: Result of DENN , PSONN and GANN on Heart dataset

	DENN	PSONN	GANN
Learning Iteration	58	10000	9000
Error Convergence	0.048925	1.46392	3.00
Convergence Time	16 Sec	170 Sec	110 Sec
Classification (%)	85.50	89.56	92.83

.

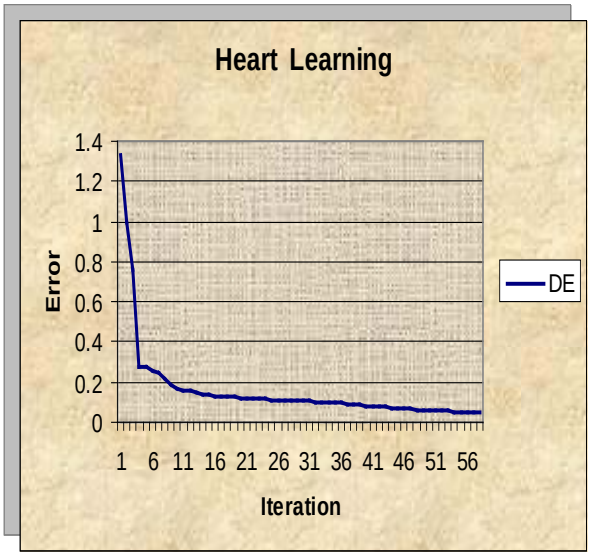

Figure 4.7: DE Convergence of Heart dataset

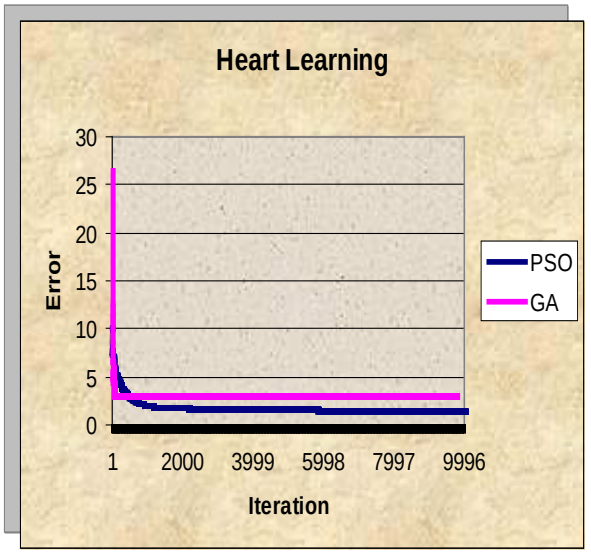

Figure 4.8: PSO,GA Convergence of Heart dataset

For Heart learning, all algorithms converge using the maximum number of pre-specified iteration. DENN takes 16 second to converge at minimum error of 0.048925 at 58 iterations .PSONN takes 170 second to converge at minimum error of 1.4639 while minimum error for GANN is 3.00 at 10000 iterations. Shows that GANN result is better than compared to PSONN and DENN, GANN= 92.83%. PSONN=89.56% , DENN=85.50%,

However, DENN convergence is faster at 58 iteration. Compared PSONN convergence is at 818 iteration and 10000 iteration in GANN. DENN , PSONN significantly reduces the error with minimum iterations compared to GANN (Figure 4.3).

4.5 Comparison DENN ,PSONN and GANN

This analysis is carried out to compare the results between PSONN and GANN. To do this, the learning patterns for all algorithms are compared using all Four datasets. The comparative correct classification percentage for all datasets is Shown in Figure 4.9

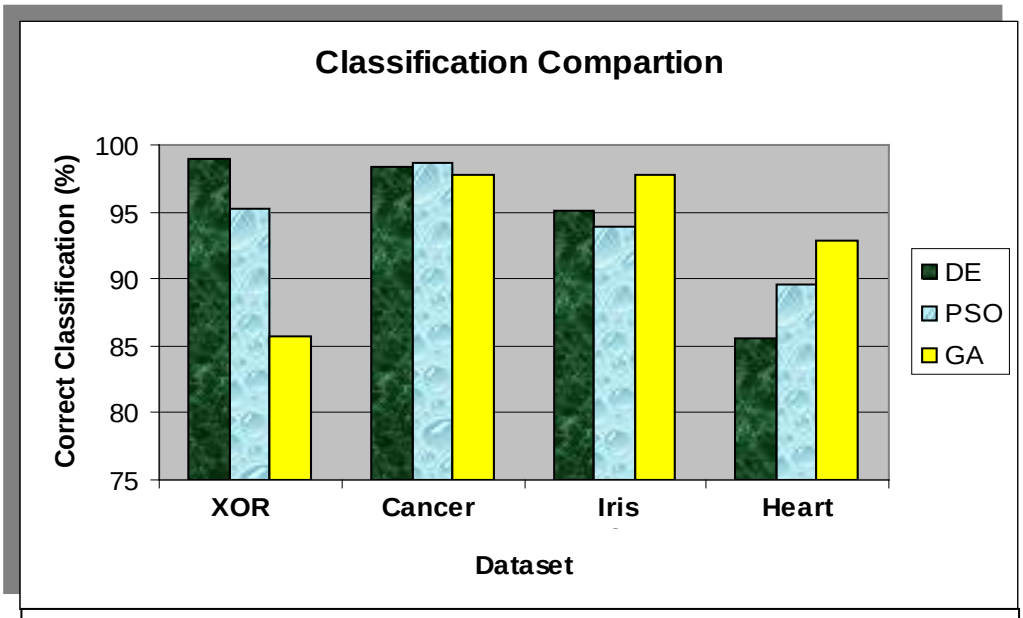

Figure 4.9: Comparative of correct classification percentage DENN,PSONN and GANN

For XOR dataset, the results show that DENN has better results on convergence time and correct classification percentage. DENN converges in a short time with high correct classification percentage. For Cancer dataset, PSONN classification results are better than DENN and GANN although both algorithms do not converge to the solution within specified minimum error, but it shows that at this time, GANN classification results are better than PSONN. But in terms of convergence time, it shows that PSONN is better than DENN, and GANN significantly reduces the error with minimum iterations. For overall performance, the experiments show that PSONN produces feasible results in terms of convergence time and classification percentage.

4.6 Discussion

In this study,a differential evolution (DE) training algorithm was used to train feedforward.term for network weights and biases (MSEREG) was included,the performance of the differential evolution training algorithm appeared to be comparable to that of the gradient based methods. Due to its scalability,it is evident

that the DE method can be applied to the training of enormous neural networks.Based on the results, it is clear that DENN is better than PSONN and GANN in term of convergence time.

The main advantages of differential evolution are:

1- no major restrictions apply to the error function,e .g., non-differentiable transfer functions may be used.
2- There are no major restrictions on the regularization methods.
3- Convergence to a global minimum can be expected.
4- Easy tuning of the algorithm parameters (mainly the size of population).

The most interesting properties are properties (1) and (2). Due to the nature of differential evolution, there are no special restrictions on the performance or regularization functions, and thus, new approaches to regularization and performance can be studied. Property (3) permits the use of differential evolution in validation of networks trained with gradient methods.

It seems that differential evolution algorithm provides new interesting topics for the training of feed-forward neural networks,such as, why some local optimum sare more attractive to differential evolution and why differential evolution finds a global minimum for some problem configurations when gradient-based methods do not,and finally what is the effect of a selected performance function on the optimum found by DE? These problems will be addressed in future work.

While In PSONN, network architecture and selection of network parameters for the dataset influence the convergence and the performance of network learning. By reducing the hidden nodes, it minimizes the number of weight (position) and also reduces the problem dimension. In this study, there are several times that PSONN runs with reduced

Number of hidden nodes, and the results have proven that the learning becomes faster. However, to have fair comparison, Kolmogorov theorem is chosen in this

study and both algorithms need to use the same network architecture. Choosing PSONN parameters also depend on the problem and dataset to be optimized. This parameter can be adjusted to achieve better optimization. But to have better comparison in this study, the same parameters for all four datasets have been used. For GANN, learning rate and momentum rate which critical for standard learning is provided by GA algorithm with a set of weight. This is to ensure the convergence time is faster with better results. Although Standard BP learning becomes faster based on parameters provided by GA, the overall process including parameters selection in GA takes much time compared to overall process in PSONN.

4.7 Summary

This chapter discusses the experimental result and analysis that has been Carried out in this study. The DE is successfully applied in neural network and has been tested using XOR, Cancer, Heart and Iris datasets. The analysis is done by comparing the results for each dataset produced by DENN, PSONN and GANN. Based on the analyses, it shows that DE is successfully applied in neural network and produced better result compared to PSONN, GANN.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter discusses the summary of the works that have been done to achieve the objective of the study. This chapter also discusses the suggestion for future work.

5.1 Discussion

As mentioned in Chapter 2, feed forward ANN is widely used as a learning algorithm in NN for many applications. However, feed forward learning depends on several parameters such as learning rate and momentum rate, and has some limitations such as slow rates of convergence and converges to local minima. Due to this, PSO or GA has been used to obtain optimal parameter value and weight for BP learning. However, in this study, a latest optimization algorithm called DE is applied in feed forward neural network to improve neural network learning mechanism.

5.2 Summary of the Work

The contributions of the study can be summarized are follows.

1. DENN algorithm converges faster with better and acceptable classification derapmoc to PSONN and GANN. DE effectively improves NN learning due to its simple calculation compared to PSO or GA with complex selection, crossover and mutation calculation.
2. Differential evolution is a small and simple mathematical model of a big and naturally complex process of evolution. So, it is easy and efficient.
3. The success of differential evolution resides the manner of the trial individual creation.
4. Summing all the aforesaid, I emphasize the three keys to DE success:
    1. Spontaneous self-adaptability.
    2. Diversity control.
    3. Continuous improvement.
5. Implementation of GA as parameter tuning to standard algorithm does not really give improvement to the convergence rate.
1. The network architecture and selection of network parameters are critical for both algorithms.

5.3 Future Work

Optimization is a procedure of finding and comparing feasible solutions until no better solution can be found. Solutions are termed good or bad in terms of an objective, which is often the cost of fabrication, amount of harmful gases, efficiency of a process, product reliability, or other factors (Deb, 2001). Most of the real world problems involve more than one objective, making the multiple conflicting objectives interesting to solve. Classical optimization methods are inconvenient to solve multi objective optimization problems, as they could at best find one solution in one simulation run.

As the real world problems involve the simulation and optimization of multiple objectives, results and solutions of these problems are conceptually different from single Objective function problems. In multi objective optimization, there may not exist a solution that is best with respect to all objectives. Instead, there are equally good, which are known as pare to optimal solutions. A pare to optimal set of solution is such that when we go from any one point to another in the set, at least one objective function improves and at least one other worsens (Yee et al., 2003). Neither of the solution dominates over each other and all the sets of decision variables on the pareto are equally good.

However, Evolutionary algorithms (EAs) can find multiple optimal solutions in one single simulation run due to their population-based search approach. Thus, EAs are ideally suited for multi- bjective optimization problems. A detailed account of multi-objective optimization using evolutionary Agorithms and some of the applications using genetic algorithms can be found in literature (Deb, 2001; Rajesh et al., 2000; Rajesh et al., 2001; Oh et al., 2002).

Based on the experiments, several suggestions can be implemented for future work and these include:

1. Develop DENN to be more public with high speed through the usage of NN weights and passing these weights directly to MSE.
2. Due to time constraints, the programs are implemented for training purposes. Therefore, further experiments need to be executed for validating the network structure on testing dataset for generalization and verification purposes.

REFERENCES

Al-kazemi B. and Mohan C.K. (2002). Training Feedforward Neural Network Using Multi-phase Particle Swarm Optimization. *Proceeding of the 9th International Conference on Neural Information Processing*. New York.

Ben Kröse, Patrick van der Smagt (1996). *An Introduction to Neural Networks*. The University of Amsterdam.

Bishop C.M. (1995). *Neural Networks for Pattern Recognition*. Chapter 7, pp.253-294. Oxford University Press.

Beasley D. and Heitkpetter J. (Eds.), "The Hitchhikers Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ)," *USENET: comp.ai.genetic*, 1997.

Bonabeau, E., Dorigo, M and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial System*. Oxford University Press. New York.

Charytoniuk, W. and Chen, M.S (2000). Neural Network Design for Short-term Load Forecasting. *International Conference on Electric Utility Deregulation and Restructuring and Power Technologies 2000*. 4-7 April 2000. City University, London, 554-561.

Cho, T-H., Conners, R.w. and Araman, P.A. (1991). Fast Back-Propagation Learning Using Steep Activation Functions and Automatic Weight Reinitialization. *Proceeding of Decision Aiding For Complex Systems Conference*. 13-16 October 1991. Charlottesville, VA USA: IEE, 1587-1592.

Cramer,N. L. "A representation for the adaptive generation of simple sequential programs," *In Proceedings of an International Conference on Genetic Algorithms and the Applications*, pp. 24-26, 1985.

Deb, K. (2001), Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons Limited, New York.

De Jong, K. A., Fogel, D. B. and Schwefel, H.-P. "A history of evolutionary computation," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford, UK: IOP & Oxford University Press,1997, pp. A2.3:1-11.

Dickmanns,D. J., Schmidhuber, and Winklhofer, A. "Der genetische Algorithmus: Eine Implementierung in Prolog.," Institut für Informatik,Technische Universität München, München 1987.

De Jong, K. A. ,Fogel,D. B. and Schwefel, H.-P. "A history of evolutionary computation," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford, UK: IOP & Oxford University Press, 1997, pp. A2.3:1-11.

Eberhart, R. and Shi, Y. (2001). Particle Swarm Optimization: Developments, Application and Resources. *IEEE*: 81-86.

Ennedy J. and Eberhart R. (2001). *Swarm Intelligence*. Inc., San Francisco, CA: Morgan Kaufmann Publishers Inc.

Fnaiech, F., Abid, S., Ellala, N. and Cheriet, M. (2002). A Comparative Study of Fast Neural Network Learning Algorithms. IEEE International Conference on Systems, Man and Cybernetics 2002. IEEE SMC. 24-29.

Fogel, L. J., Owens, A. J. and Walsh,M. J. *Artificial intelligence through simulated evolution*. New York: John Wiley and Sons, Inc., 1966.

Fogel, L.J., Owens A.J., and Walse, M.J. *Artificial intelligence through simulated evolution*. New York: John Wiley and Sons, Inc., 1966.

Feoktistov, V. (2006). Differential Evolution In Search of Solutions. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA.

Ferguson, D. (2004). *Particle Swarm*. University of Victoria, Canada.

Fnaiech, F., Abid, S., Ellala, N. and Cheriet, M. (2002). A Comparative Study of Fast Neural Network Learning Algorithms. *IEEE International Conference on Systems, Man and Cybernetics 2002*. *IEEE SMC*. 24-29.

Holland J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*: The University of Michigan Press, 1975.

Haykin, S. (1999). *Neural Network. A Comprehensive Foundation*. 2nd Ed: Prentice Hall.

Hussein A. Abbass, Ruhul Sarker, and Charles Newton. A pareto differential evolution approach to vector optimization problems. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC2001, Seoul, Korea, IEEE Press, 2001.

Ilonen,J, Kamarainen,J.I and Lampinen,J.(2003), Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters, v.17 n.1, p.93-105.

Jarmo Ilonen, Joni-Kristian Kamarainen, and Jouni Lampinen. Differential evolution training algorithm for feed-forward neural networks. Neural Processing Letters, 17(1):93–105, 2003.

Jones M.T (2005). *AI Application Programming*. 2nd Ed. Hingham, Massachusetts: Charles River Media Inc.

Koza,J. R. , Banzhaf,W. K. , Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M.H. Garzon, D. E. Goldberg, H. Iba, and R. L. Riolo, "Genetic Programming1998: Proceedings of the Third Annual Conference," *Evolutionary Computation, IEEE Transactions on*, vol. 3, pp. 159-161, 1999.

Koza J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

Kim, G-H., Yoon, J-E., An, S-H., Cho, H-H. and Kang, K-I. (2004). Neural Network Model Incorporating A Genetic Algorithm in Estimating Construction Cost. Building and Environment. 39(11): 1333-1340.

Luger, George F., (2002). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 4th ed.. Addison-Wesley/Pearson Education Limited, Harlow, England. 417-473.

Limsombunchai, M., Clemes and Weng, A. (2000). Consumer Choice Prediction: Artificial Neural Networks versus Logistic Model. Lincoln University, New Zealand.

Lee, J. S., Lee, S., Chang, S. and Ahn, B. H. (2005). A Comparison of GA and PSO for Excess Return Evaluation in Stock Markets. *Springer-Verlag*: 221-230.

Nenortaite, J. and Simutis, R. (2004). Stock Trading System Based on the Particle Swarm Optimization Algorithm. *Springer-Verlag*: 843-850.

Oh P.P., Rangaiah,G.P. and A.K. Ray (2002), " Simulation and Multi-objective Optimization of an Industrial Hydrogen Plant Based on Refinery Off-gas" .Industrial and Engineering Chemistry Research, 41, 2248-2261.

Friedberg,R. M. "A Learning Machine: Part I," *IBM J. Research and Development*, vol. 2, pp. 2-13, 1958.

Friedberg, R. M. , Dunham, B. and North, J. H. "A Learning Machine: Part II," *IBM J. Research and Development*,, vol. 3, pp. 183-191, 1959.

Rumelhart, D.E. and McClelland, J.L. (1986). *Parallel Distributed Processing: Explorations in The Microstructure of Cognition*. Vol 1. MIT press, Cambridge, MA.Rainer Storn, Kenneth Price: Differential Evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Global Optimization, 11, 1997 341-359

Randall S. S. and Naheel A. S. (2001). Data Mining Using a Genetic Algorithm-Trained Neural Network. *International Journal of Intelligent Systems in Accounting, Finance & Management*.10, 201–210.

Rajesh, J.K., Gupta, S.K. , Rangaiah, G.P. and Ray, A.K. (2000), "Multi-objective Optimization of Steam Reformer Performance Using Genetic Algorithm" . Industrial and Engineering Chemistry Research, 39, 707-717.

Rajesh, J.K., Gupta,S.K. , Rangaiah,G.P. and Ray, A.K. (2001), "Multi-objective Optimization of Industrial Hydrogen Plants" . Chemical Engineering Science, 56, 999.

Shi, Y. (2004). Particle Swarm Optimization. *IEEE Neural Network Society:* 8-13.

Siti Mariyam Hj. Shamsuddin (2004). *Lecture Note Advanced Artificial Intelligence: Number of Hidden Neurons*. Universiti Teknologi Malaysia. Unpublished.

Song, M. P. and Gu, G.C. (2004). Research on Particle Swarm Optimization: A Review. *IEEE*: 2236-2241.

Storn,R. and Price,K.: Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces,Technical Report TR-95- 012,International Computer Science Institute,Berkeley,CA,USA http://www.icsi.berkeley.edu/ techreports/1995.abstracts/tr-95-012.html),1995.

Van den Bergh, F. (1999). Particle Swarm Weight Initialization In Multi-Layer Perceptron Artificial Neural Networks. *Accepted for ICAI*. Durban, South Africa.

Van den Bergh, F. and Engelbrecht, A. P. (2000). Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal*. (26):84-90.

Wyeth ,G., Buskey, G. and Roberts , J. (2002) . Flight Control Using an Artificial Neural Network. Computer Science and Electrical Engineering University of Queensland. Australia.

Wyeth, G., Buskey, G. and Roberts, J. (2000). *Flight Control Using an Artificial Neural Network*. University of Queensland.

Yao X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*. 4: 203--222.

Yao, X. (1999). 'Evolving artificial neural networks', *Proceedings of the IEEE.* vol. 87, no. 9: 1423 -1447.

Yee, A.K.Y., A.K. Ray, and G.P. Rangaiah (2003),"Multiobjective optimization of an industrial styrene reactor" . Computers & Chemical Engineering, 27, 111-130. Zhang, C., Shao, H. and Li, Y. (2000). Particle Swarm Optimization for Evolving Artificial Neural Network. *IEEE*: 2487-2490.

Zweiri, Y.H., Whidborne, J.F. and Sceviratne, L.D. (2002). A Three-term Backpropagation Algorithm. *Neurocomputing*. 50: 305-318.

APPENDIX A

Normalized XOR Data

Input		Output
First input	**Second input**	
0	0	0
1	0	1
0	1	1
1	1	0

APPENDIX B

Normalized Cancer Dataset

Input									Output
Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.333	0.333	0.444	0.667	1.000	0.222	0.111	0.000	0
0.200	0.000	0.000	0.000	0.111	0.200	0.222	0.000	0.000	0
0.500	0.778	0.778	0.000	0.222	0.400	0.222	0.667	0.000	0
0.300	0.000	0.000	0.222	0.111	0.100	0.222	0.000	0.000	0
0.700	1.000	1.000	0.778	0.667	1.000	0.889	0.667	0.000	1
0.000	0.000	0.000	0.000	0.111	1.000	0.222	0.000	0.000	0
0.100	0.000	0.111	0.000	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.444	0
0.300	0.111	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.222	0.222	0.222	0.111	0.300	0.333	0.333	0.000	1
0.000	0.000	0.000	0.000	0.111	0.300	0.222	0.000	0.000	0
0.700	0.667	0.444	1.000	0.667	0.900	0.444	0.444	0.333	1
0.600	0.333	0.556	0.333	0.556	0.100	0.333	0.222	0.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.900	0.667	0.667	0.556	0.333	1.000	0.333	0.000	0.111	1
0.500	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.600	0.222	0.111	1.000	0.444	1.000	0.444	0.333	0.333	1
0.900	0.444	0.444	0.222	0.556	0.700	0.667	1.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.700	0.333	0.444	0.000	0.111	0.000	0.667	0.222	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.111	0.222	0.333	0.111	0.700	0.222	0.556	0.000	1

0.200	0.111	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.222	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.900	0.667	0.667	0.222	0.778	0.500	0.667	0.333	0.222	1
0.100	0.000	0.000	0.111	0.111	0.100	0.222	0.000	0.000	0
0.200	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.900	1.000	1.000	0.778	0.556	0.100	0.778	0.889	0.000	1
0.500	0.111	0.000	0.000	0.000	0.100	0.667	0.000	0.000	0
0.400	0.333	0.333	0.889	0.111	1.000	0.444	0.556	0.000	1
0.100	0.444	0.222	0.222	0.556	0.700	0.667	0.444	0.000	1
0.500	0.556	0.556	0.889	0.556	0.000	0.667	0.778	0.000	0
0.900	0.333	0.222	0.000	0.222	0.300	0.556	0.444	0.111	1
0.500	1.000	1.000	0.111	0.778	1.000	0.667	0.222	0.222	1
0.400	0.556	0.444	0.556	1.000	0.100	0.222	0.000	0.000	1
0.900	1.000	1.000	0.333	0.778	0.100	0.778	1.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.111	0
0.200	0.667	0.667	0.333	0.333	0.900	0.333	0.778	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.222	0.111	0.100	0.222	0.000	0.000	0
0.600	0.778	0.667	0.111	0.333	0.800	0.222	0.778	0.111	1
0.800	0.444	0.778	0.000	0.111	0.300	0.111	0.000	0.444	1
0.400	0.222	0.222	0.333	0.111	0.400	0.222	0.333	0.000	1
0.900	0.222	0.556	0.111	0.222	0.500	0.333	1.000	0.111	1
0.400	0.444	0.444	0.778	1.000	0.800	0.667	0.222	0.667	1
0.900	0.444	0.444	0.556	0.778	0.800	0.667	0.000	0.000	1
0.900	0.556	0.556	0.222	0.333	0.500	0.222	0.556	0.000	1
0.700	1.000	1.000	0.000	0.222	0.600	0.222	0.889	0.000	1
0.700	0.111	0.333	0.000	0.444	0.100	0.444	0.333	0.333	1
0.400	0.111	0.222	0.000	0.556	1.000	0.444	0.000	0.000	1
0.800	0.444	0.444	0.111	0.111	0.200	0.444	0.000	0.000	1
0.400	0.222	0.444	0.444	0.222	0.300	0.333	1.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.200	0.111	0.000	0.000	0
0.800	1.000	1.000	0.000	1.000	0.800	0.222	0.222	0.000	1
0.500	0.222	0.333	0.000	0.444	0.200	0.222	0.889	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.900	0.333	0.111	0.000	0.222	0.200	0.333	0.222	1.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.222	0.333	0.000	0.778	1.000	0.333	0.889	0.000	1
0.700	0.222	0.778	0.222	0.333	0.900	0.778	0.889	0.778	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.111	0.000	0
0.400	0.000	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.500	1.000	0.111	0.778	1.000	0.200	0.667	0.778	1.000	1

0.000	0.222	0.222	0.111	0.111	0.100	0.667	0.111	0.000	0
0.800	0.333	0.444	1.000	0.556	1.000	0.333	0.778	0.000	1
0.900	0.556	0.333	0.000	0.222	0.400	0.222	0.111	0.222	1
0.000	0.000	0.111	0.000	0.111	0.200	0.333	0.111	0.000	0
0.000	0.000	0.333	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.222	0.000	0.111	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.300	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.222	0.100	0.111	0.000	0.000	0
0.100	0.111	0.111	0.000	0.000	0.100	0.667	0.000	0.000	0
0.300	0.000	0.000	0.111	0.111	0.100	0.111	0.000	0.000	0
0.400	0.111	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.200	0.667	0.000	0.000	0
0.200	0.444	0.667	0.778	0.778	0.900	0.667	1.000	0.667	1
0.400	1.000	0.556	0.000	1.000	0.400	0.333	1.000	1.000	1
0.200	0.222	0.556	0.333	0.444	0.800	0.333	0.333	0.000	1
0.200	0.556	0.556	0.556	0.444	1.000	0.556	0.778	0.222	1
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.111	0.222	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.000	0.000	0.111	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.111	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.800	0.556	0.889	0.111	1.000	0.600	0.111	0.889	1.000	1
0.600	0.444	0.556	1.000	0.444	1.000	0.667	0.889	0.333	1
0.900	0.222	0.444	0.000	1.000	0.500	0.222	1.000	0.111	1
0.100	0.222	0.333	0.333	0.111	0.500	0.111	0.444	0.000	1
0.300	0.000	0.111	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.111	0.222	0.000	0.556	0.300	0.667	0.000	0.000	1
0.900	1.000	1.000	1.000	1.000	0.100	0.778	0.778	0.778	1
0.600	0.222	0.333	0.333	0.222	0.300	0.222	0.111	0.667	1
0.900	1.000	1.000	0.778	0.111	1.000	0.333	0.000	0.000	1
0.000	0.556	0.778	1.000	0.778	1.000	0.444	0.667	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.222	0.000	0
0.500	0.444	0.333	0.333	0.222	0.900	0.667	0.778	0.222	1
0.000	0.222	0.000	0.111	0.111	0.200	0.444	0.222	0.111	0
0.700	0.556	0.333	0.222	0.444	0.900	0.222	0.000	0.000	1
0.900	0.222	0.222	1.000	0.111	1.000	0.667	0.222	0.222	1
0.900	1.000	1.000	0.222	1.000	0.800	0.778	0.000	0.000	1
0.200	0.222	0.111	0.000	0.111	0.300	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.500	0.000	0.000	0.000	0
0.700	0.222	0.222	0.000	0.111	0.200	0.222	0.111	0.000	0
0.300	0.444	0.444	1.000	0.333	1.000	0.667	0.444	0.778	1

0.000	0.000	0.000	0.000	0.333	0.300	0.000	0.000	0.000	0
0.200	0.111	0.000	0.000	0.111	0.200	0.222	0.000	0.000	0
0.000	0.000	0.111	0.111	0.111	0.100	0.222	0.000	0.000	0
0.300	0.111	0.000	0.000	0.111	0.200	0.222	0.000	0.000	0
0.900	1.000	1.000	0.111	1.000	1.000	0.444	0.222	0.222	1
0.400	0.222	0.444	0.000	0.778	1.000	0.444	0.222	0.000	1
0.400	0.333	0.556	0.667	0.889	0.700	0.778	1.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.600	0.444	0.222	0.667	0.333	1.000	0.667	0.444	0.444	1
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.222	0.444	0.333	0.444	1.000	0.000	0.556	0.111	1
0.000	0.000	0.000	0.000	1.000	0.100	0.000	0.000	0.000	0
0.400	0.000	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	1.000	0.778	1.000	0.778	1.000	0.222	0.556	0.222	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.111	0.000	0
0.200	0.000	0.000	0.000	0.222	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.200	0.222	0.222	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.000	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.800	0.444	0.444	0.333	0.333	0.500	0.333	0.222	0.222	1
0.000	0.000	0.000	0.000	0.111	0.500	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.222	0.000	0.111	0.000	0.111	0.000	0.000	0
0.200	0.333	0.444	0.111	0.556	0.800	0.333	0.000	0.000	1
0.000	0.000	0.000	0.000	0.222	0.200	0.111	0.000	0.000	0
0.200	0.000	0.000	0.222	0.778	0.100	0.444	0.778	0.000	0
0.700	0.778	0.667	0.333	1.000	1.000	0.667	0.778	0.667	1
0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.600	0.111	0.333	0.000	0.556	1.000	0.444	0.333	0.222	1
0.900	1.000	0.778	0.556	0.333	0.500	0.778	1.000	0.000	1
0.300	0.000	0.000	0.000	0.111	0.300	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.444	0.444	0.556	0.222	1.000	0.222	0.000	0.000	1
0.000	0.111	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.111	0.000	0.222	0.000	0.000	0.000	0.000	0
0.800	0.889	1.000	0.222	0.556	1.000	0.667	1.000	0.556	1
0.900	0.667	0.667	0.333	0.444	1.000	0.444	0.667	0.111	1
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.111	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.111	0.000	0.300	0.000	0.000	0.667	0

0.400	0.000	0.000	0.000	0.111	0.000	0.222	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.200	0.222	0.111	0.000	0
0.400	0.556	0.667	0.778	0.778	1.000	0.222	1.000	0.222	1
0.900	0.778	1.000	1.000	0.556	0.100	0.222	0.000	1.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.111	0.000	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.500	1.000	1.000	1.000	0.778	1.000	1.000	1.000	0.667	1
0.700	0.556	0.444	0.333	0.222	1.000	0.556	0.000	0.000	1
0.400	0.778	0.667	0.667	1.000	1.000	0.444	0.667	0.000	1
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	1.000	1.000	0.222	0.778	0.100	0.444	1.000	0.222	1
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.222	0.222	0.222	0.556	1.000	0.222	0.000	0.000	1
0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.500	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.778	0.778	0.778	0.444	1.000	0.667	0.778	0.000	1
0.700	0.667	0.556	0.333	0.333	1.000	0.444	0.000	0.000	1
0.100	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.000	0.444	0.778	0.556	0.444	0.800	0.667	1.000	0.000	1
0.900	0.444	0.556	1.000	0.556	1.000	0.667	0.667	1.000	1
0.400	0.778	0.333	1.000	0.444	0.800	0.889	1.000	0.000	1
0.000	0.111	0.222	0.000	0.111	0.100	0.222	0.000	0.000	0
0.900	1.000	1.000	0.778	0.556	0.800	0.667	1.000	0.000	1
0.600	0.444	1.000	1.000	1.000	1.000	0.333	1.000	0.222	1
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.333	0.333	0.444	0.333	0.700	0.667	0.778	0.111	0
0.400	0.000	0.000	0.333	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.800	0.667	0.667	0.444	0.444	1.000	0.667	0.778	0.222	1
0.900	0.778	0.778	0.333	1.000	1.000	0.778	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	1.000	1.000	0.889	0.556	1.000	0.667	1.000	0.444	1
0.900	1.000	0.889	0.222	0.667	0.500	0.222	0.444	0.000	1
0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.400	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0

0.700	1.000	1.000	1.000	0.444	1.000	0.778	1.000	0.556	1
0.700	1.000	0.778	0.778	0.333	0.800	0.667	0.667	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.900	1.000	1.000	1.000	0.667	1.000	0.667	1.000	0.333	1
0.900	1.000	1.000	1.000	0.222	1.000	1.000	0.556	0.000	1
0.700	0.667	0.778	0.667	0.444	0.500	0.444	1.000	0.111	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.500	1.000	0.667	0.667	0.556	0.400	0.778	1.000	0.111	1
0.500	0.000	0.222	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.111	0.111	0.100	0.222	0.000	0.000	0
0.900	0.556	0.333	0.222	1.000	1.000	0.889	1.000	0.000	1
0.300	0.000	0.000	0.222	0.000	0.500	0.111	0.000	0.000	1
0.600	0.444	0.556	0.222	0.222	0.800	0.667	0.333	0.000	1
0.900	0.444	0.444	0.556	0.222	1.000	0.667	0.889	0.111	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.900	0.444	0.667	0.333	0.333	1.000	0.778	0.889	0.000	1
0.700	0.889	0.889	0.444	0.222	0.500	0.667	0.667	0.000	1
0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.900	1.000	1.000	0.222	1.000	1.000	0.889	1.000	0.000	1
0.600	0.333	0.667	0.333	0.222	0.700	0.667	0.556	0.000	1
0.500	0.778	0.667	0.444	0.556	0.800	0.778	0.889	0.111	1
0.700	0.333	0.556	0.222	0.222	0.100	0.333	0.222	0.000	0
0.900	0.333	0.444	0.444	0.444	1.000	0.333	0.000	0.000	1
0.200	0.222	0.111	0.000	0.222	0.100	0.222	0.556	0.000	0
0.200	0.000	0.333	0.000	0.111	0.000	0.222	0.000	0.000	0
0.900	0.778	0.778	0.111	0.778	1.000	0.333	0.778	1.000	1
0.800	0.778	0.778	0.444	0.556	0.200	0.333	1.000	0.333	1
0.700	1.000	1.000	0.778	0.556	0.900	0.222	1.000	1.000	1
0.900	0.333	0.222	0.111	0.222	1.000	0.444	0.222	0.111	1
0.400	0.000	0.222	0.222	0.111	0.200	0.111	0.222	0.000	0
0.200	0.000	0.000	0.222	0.000	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.500	0.444	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.000	0.111	0.111	0.200	0.222	0.000	0.000	0
0.700	1.000	1.000	0.778	0.444	1.000	0.667	0.778	0.000	1
0.700	0.333	0.333	0.000	0.111	0.900	0.222	0.222	0.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.556	0.000	0
0.200	0.000	0.000	0.000	0.111	0.000	0.222	0.000	0.000	0
0.000	0.111	0.111	0.000	0.111	0.100	0.000	0.000	0.000	0
0.900	0.333	0.333	1.000	0.111	1.000	0.444	0.222	0.222	1
0.500	0.222	0.222	0.444	0.222	1.000	0.222	0.444	0.222	0
0.500	1.000	1.000	0.111	0.778	1.000	0.667	0.222	0.222	1
0.800	1.000	1.000	0.000	1.000	0.800	0.222	0.222	0.000	1
0.400	0.556	0.556	0.111	0.333	1.000	0.222	0.556	0.000	1

0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.667	0.667	0.000	0.444	0.800	0.222	0.333	0.000	0
0.900	0.444	0.778	1.000	0.222	1.000	0.444	0.000	0.222	1
0.400	1.000	1.000	0.556	1.000	1.000	1.000	0.556	0.444	1
0.700	0.778	0.889	0.333	0.444	1.000	0.667	0.778	0.000	1
0.900	0.333	0.333	1.000	0.556	1.000	0.444	0.444	0.000	1
0.600	0.889	0.333	1.000	1.000	0.300	0.444	0.222	0.222	1
0.400	0.000	0.333	0.000	0.111	0.100	0.222	0.111	0.000	0
0.900	1.000	0.556	0.222	0.222	1.000	0.333	0.222	0.111	1
0.200	0.222	0.444	0.111	0.222	1.000	0.667	0.000	0.000	1
0.900	0.778	0.778	0.111	0.222	0.400	0.778	0.667	0.778	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.333	0.667	0.000	0.222	1.000	0.222	0.889	0.111	1
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.222	0.444	0.111	0.222	1.000	0.667	0.000	0.000	1
0.600	0.111	0.333	0.000	0.222	0.400	0.222	0.222	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.111	0.000	0
0.200	0.000	0.222	0.000	0.111	0.000	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.900	0.444	0.667	0.222	0.222	0.700	0.222	0.222	0.778	1
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.111	0.111	0.100	0.222	0.000	0.000	0
0.000	0.333	0.222	1.000	0.333	1.000	0.444	0.556	0.000	1
0.900	0.333	0.556	0.000	0.111	1.000	0.444	0.222	0.000	1
0.600	0.333	0.444	1.000	0.111	1.000	0.222	0.778	0.111	1
0.700	1.000	1.000	1.000	0.778	1.000	1.000	0.667	0.222	1
0.900	1.000	1.000	1.000	1.000	1.000	0.333	1.000	1.000	1
0.200	0.000	0.000	0.000	0.222	0.100	0.111	0.000	0.000	0
0.500	0.000	0.222	0.000	0.333	0.500	0.444	1.000	0.000	1
0.400	0.556	0.556	0.778	0.556	1.000	0.333	1.000	0.333	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.778	0.778	0.000	0.111	0.000	0.556	1.000	0.000	1
0.900	0.333	0.333	0.556	0.111	1.000	0.111	0.222	0.000	1
0.000	0.000	0.000	0.000	0.111	0.000	0.111	0.000	0.000	0
0.400	0.444	0.667	0.778	0.556	1.000	0.667	0.333	0.000	1
0.400	0.222	0.333	0.222	0.333	0.500	0.333	0.667	0.000	0
0.400	0.333	0.222	0.000	0.111	0.000	0.111	0.222	0.000	0
0.700	0.111	0.000	0.000	0.444	0.100	0.000	0.000	0.000	0
0.800	0.000	0.111	0.556	0.333	1.000	0.667	0.667	0.111	1
0.700	0.333	1.000	0.444	0.333	0.400	0.667	1.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0

0.900	1.000	1.000	0.667	0.889	1.000	0.667	1.000	1.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.222	0.333	0.889	0.222	1.000	0.222	0.222	0.000	1
0.900	0.778	0.333	0.333	0.333	1.000	0.222	1.000	0.333	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.600	0.778	0.667	0.556	0.333	0.300	0.778	0.778	0.333	1
0.200	0.000	0.000	0.000	0.111	0.500	0.444	0.000	0.000	0
0.100	0.000	0.000	0.000	0.222	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.700	0.556	0.333	1.000	1.000	0.100	0.222	0.444	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.300	0.556	0.444	0.556	0.667	0.000	0.333	0.889	0.000	0
0.400	0.444	0.444	0.111	0.444	1.000	0.333	0.222	0.000	1
0.500	0.778	0.667	0.778	0.556	0.800	0.778	0.889	0.000	1
0.000	0.000	0.000	0.000	0.444	0.100	0.222	0.000	0.000	0
0.300	0.333	0.333	0.333	0.556	0.500	0.667	0.222	0.000	0
0.600	0.556	0.222	0.111	0.444	1.000	0.667	0.333	0.556	1
0.200	0.000	0.000	0.000	0.111	0.000	0.222	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.333	0.556	1.000	0.111	1.000	0.333	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.111	0.111	0.000	0.111	0.100	0.111	0.222	0.000	0
0.900	0.000	0.000	0.000	0.111	1.000	0.444	0.333	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.700	1.000	0.222	0.111	0.556	0.400	0.222	1.000	0.000	1
0.900	0.333	0.556	0.333	0.444	1.000	0.667	0.000	0.000	1
0.900	0.333	0.667	0.111	0.111	0.800	0.556	0.000	0.000	1
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.111	0
0.400	0.111	0.111	0.111	0.111	0.100	0.111	0.111	0.000	0
0.400	0.333	0.556	0.556	0.333	1.000	0.333	0.222	0.000	1
0.700	0.556	0.667	0.222	0.222	1.000	0.222	0.333	0.111	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.500	0.444	0.444	0.778	0.333	1.000	0.222	0.333	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.700	0.444	0.444	0.444	0.111	1.000	0.333	0.222	0.000	1
0.900	0.222	0.222	0.000	0.111	1.000	0.667	0.556	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.600	0.556	0.333	0.778	1.000	1.000	0.889	0.444	0.222	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.111	0.111	0.111	0.222	0.100	0.000	0.222	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.000	0.222	0.000	0

0.200	0.333	0.333	1.000	0.444	0.100	0.222	0.222	0.000	1
0.300	0.111	0.222	0.444	0.222	0.800	0.667	0.556	0.000	1
0.400	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.333	0.444	0.222	0.667	0.300	0.333	0.556	0.000	0
0.100	0.667	1.000	1.000	0.667	1.000	0.333	0.889	0.333	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.222	0.100	0.111	0.111	0.000	0
0.400	0.222	0.222	0.000	0.222	0.300	0.222	0.222	0.222	1
0.700	1.000	1.000	0.667	1.000	1.000	0.667	0.222	0.778	1
0.700	1.000	0.444	0.222	0.778	0.400	0.333	1.000	0.222	1
0.900	0.222	0.444	0.333	0.222	0.700	0.222	0.444	0.222	1
0.500	1.000	1.000	1.000	1.000	1.000	0.778	1.000	1.000	1
0.200	1.000	0.222	1.000	0.556	1.000	0.444	0.000	0.333	1
0.200	0.111	0.111	0.000	0.333	0.300	0.111	0.000	0.000	0
0.300	0.333	0.333	0.111	0.111	0.300	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.500	1.000	1.000	1.000	0.778	1.000	0.667	1.000	0.667	1
0.400	0.778	0.778	1.000	0.444	1.000	0.778	1.000	0.222	1
0.000	0.000	0.222	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.222	0.000	0.000	0.100	0.111	0.000	0.000	0
0.300	0.222	0.111	0.000	0.222	0.100	0.111	0.000	0.000	0
0.000	0.000	0.222	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.111	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.333	0.222	0.100	0.111	0.111	0.000	0
0.400	0.222	0.333	0.000	0.333	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.900	0.556	0.222	0.556	0.333	1.000	0.667	0.778	0.333	1
0.200	0.111	0.111	0.111	0.111	0.100	0.222	0.111	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.222	0.111	0.111	0.222	0.100	0.000	0.111	0.222	0
0.600	0.556	0.556	0.222	0.111	1.000	0.667	0.000	0.000	1
0.400	0.222	0.222	0.111	0.222	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.111	0.000	0
0.400	0.000	0.000	0.000	0.222	0.200	0.111	0.111	0.000	0
0.000	0.000	0.000	0.111	0.111	0.100	0.111	0.000	0.000	0
0.900	0.778	0.667	0.333	0.222	1.000	0.667	0.889	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0

0.000	0.111	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.111	0.000	0.000	0.111	0.100	0.111	0.111	0.000	0
0.000	0.111	0.222	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	1.000	0.778	0.667	0.556	0.900	0.889	0.222	0.778	1
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.222	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.400	0.000	0.000	0.000	0
0.000	0.111	0.000	0.222	0.111	0.100	0.000	0.111	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.111	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.222	0.111	0.111	0.111	0.200	0.222	0.000	0.000	0
0.200	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.000	0.111	0.000	0.000	0
0.900	1.000	1.000	0.556	0.778	0.400	0.778	0.444	0.000	1
0.400	0.000	0.111	0.000	0.111	0.100	0.222	0.000	0.000	0
0.700	0.444	0.556	0.111	0.222	1.000	0.556	0.556	0.000	1
0.200	0.222	0.111	0.556	0.222	0.300	0.222	0.444	0.000	0
0.700	0.667	0.778	0.444	1.000	1.000	0.667	0.111	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.111	0.111	0.111	0.111	0.200	0.222	0.111	0.111	0
0.100	0.222	0.000	0.000	0.444	0.100	0.000	0.000	0.000	0
0.200	0.111	0.111	0.222	0.111	0.300	0.222	0.000	0.000	0
0.900	1.000	1.000	0.667	1.000	1.000	0.778	0.111	0.000	1
0.300	0.222	0.222	0.000	0.111	0.100	0.222	0.222	0.000	0
0.400	0.000	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.800	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	1
0.400	0.222	0.556	0.000	0.111	0.100	0.000	0.000	0.000	0
0.700	0.667	0.778	0.111	0.333	0.200	0.444	1.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.222	0.000	0.000	0.111	0.100	0.111	0.111	0.000	0
0.400	0.000	0.000	0.222	0.333	0.100	0.222	0.111	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.111	0.000	0
0.200	0.111	0.111	0.222	0.111	0.100	0.000	0.000	0.000	0
0.500	0.889	0.667	0.444	0.444	0.800	0.333	0.111	0.000	0
0.900	0.778	1.000	0.000	0.222	1.000	0.444	0.000	0.000	1
0.900	1.000	1.000	0.000	0.556	0.100	0.111	0.778	0.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.222	0.222	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0

0.900	0.333	0.222	1.000	0.333	1.000	1.000	0.000	0.000	1
0.400	0.111	0.111	0.333	0.111	0.400	0.000	0.000	0.000	0
0.000	0.000	0.000	0.222	0.111	0.300	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.200	0.000	0.000	0.000	0
0.400	0.000	0.000	0.556	0.222	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.400	0.667	0.889	0.778	0.556	1.000	0.778	1.000	0.000	1
0.300	0.000	0.000	0.222	0.000	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.300	0.444	0.444	0.778	0.556	1.000	1.000	0.667	0.000	1
0.100	0.222	0.000	0.000	0.222	0.100	0.000	0.000	0.000	0
0.900	0.111	0.111	0.000	0.111	0.600	0.000	0.000	0.111	1
0.900	0.556	0.444	0.778	0.444	1.000	0.778	0.556	0.000	1
0.700	0.778	0.889	0.556	0.556	0.300	1.000	1.000	0.000	1
0.400	0.000	0.111	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.222	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.500	0.000	0.000	0.000	0
0.500	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.111	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.900	0.889	0.778	0.667	0.556	0.400	0.667	1.000	0.222	1
0.900	0.556	0.556	0.111	0.333	1.000	0.889	0.667	0.000	1
0.500	0.556	0.556	0.444	0.333	1.000	0.667	0.556	0.111	1
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.500	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.500	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.111	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.111	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.778	0.667	1.000	0.333	1.000	0.667	0.444	0.000	1
0.400	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.400	0.222	0.111	0.333	0.111	0.100	0.000	0.000	0.000	0
0.800	1.000	1.000	1.000	1.000	0.500	1.000	1.000	1.000	1
0.700	0.667	0.778	0.444	0.444	1.000	0.889	1.000	0.000	1
0.400	0.000	0.111	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.222	0.000	0.300	0.000	0.000	0.000	0

0.200	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.900	1.000	1.000	1.000	0.556	1.000	0.778	0.000	0.444	1
0.200	0.556	0.333	1.000	0.222	0.300	0.222	0.333	0.000	1
0.500	0.222	0.111	0.000	0.222	0.400	0.333	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.778	0.889	0.333	0.222	1.000	0.667	0.000	0.000	1
0.300	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.400	1.000	1.000	1.000	0.556	1.000	0.556	0.444	0.111	1
0.400	0.000	0.111	1.000	0.333	0.500	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.300	0.111	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.500	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.111	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.222	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.700	1.000	1.000	1.000	0.667	0.500	0.333	0.778	0.667	1
0.000	0.000	0.000	0.000	0.111	0.400	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.500	0.556	0.667	1.000	0.222	1.000	0.778	1.000	0.111	1
0.300	1.000	0.333	0.667	0.222	1.000	0.889	1.000	0.000	1
0.000	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.200	0.000	0.111	0.111	0.111	0.100	0.000	0.000	0.000	0
0.300	0.667	0.778	0.222	0.333	1.000	0.889	0.000	0.000	1
0.000	0.000	0.000	0.000	0.222	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.222	0.100	0.000	0.000	0.000	0
0.900	0.333	0.444	0.333	0.222	0.500	0.667	0.222	0.000	1
0.600	0.444	0.556	1.000	0.333	1.000	0.444	0.222	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.111	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.500	0.000	0.222	0.111	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.600	0.333	0.333	0.222	0.333	1.000	0.556	0.889	0.000	1
0.300	0.111	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0

0.000	0.000	0.000	0.000	0.000	0.100	0.222	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.222	0.111	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.111	0.000	0.111	0.100	0.222	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.500	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.200	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.222	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.100	0.000	0.222	0.111	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.500	1.000	1.000	1.000	0.333	1.000	0.667	1.000	0.000	1
0.100	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.000	0.100	0.000	0.000	0.000	0
0.600	0.778	0.222	0.667	0.333	0.500	0.667	0.778	0.111	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.111	0.111	0.111	0.111	0.100	0.333	0.111	0.000	0
0.300	0.333	0.111	0.000	0.111	0.500	0.111	0.000	0.111	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.222	0.000	0.000	0.111	0.100	0.333	0.778	0.000	0
0.400	0.111	0.111	0.111	0.000	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.111	0.000	0
0.400	0.667	1.000	1.000	0.444	1.000	1.000	1.000	0.000	1
0.200	0.000	0.111	0.000	0.111	0.100	0.222	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.300	0.111	0.000	0.000	0
0.700	0.333	0.333	0.000	0.556	1.000	0.111	0.444	0.111	1
0.900	1.000	0.778	1.000	0.556	0.500	1.000	0.222	0.000	1
0.700	1.000	0.333	0.333	0.778	1.000	0.778	0.111	0.000	1
0.600	0.556	1.000	0.444	0.222	1.000	0.889	1.000	0.111	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.900	0.889	0.667	0.222	0.333	0.200	0.667	0.667	0.000	1
0.400	0.000	0.111	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0

0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	0
0.400	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.667	1.000	0.556	0.444	1.000	0.667	0.444	0.000	1
0.500	1.000	0.444	0.444	0.333	1.000	0.556	1.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.556	0.222	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.700	1.000	1.000	1.000	0.556	1.000	1.000	1.000	0.000	1
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.111	0.000	0
0.800	0.778	0.778	0.889	0.556	0.300	0.333	0.000	0.000	1
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	1.000	0.778	0.444	0.333	0.100	1.000	0.000	0.000	1
0.100	0.444	0.667	0.556	0.333	1.000	0.667	0.556	0.000	1
0.900	0.222	0.333	0.444	0.222	1.000	0.333	0.000	0.000	1
0.400	0.000	0.111	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.778	0.556	0.222	0.333	1.000	0.667	0.000	0.000	1
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.222	0.000	0.111	0.100	0.222	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.111	0.333	0.000	0.000	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.333	0.556	0.778	0.333	0.100	0.778	1.000	0.000	0
0.400	0.222	0.111	0.778	0.444	1.000	0.778	0.000	0.111	1
0.900	0.444	1.000	0.222	0.444	0.800	0.667	0.778	0.222	0
0.300	0.000	0.000	0.111	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.900	0.333	0.222	1.000	0.222	1.000	0.667	0.000	0.111	0
0.400	1.000	1.000	1.000	0.444	0.200	0.778	0.444	0.000	1
0.700	1.000	1.000	1.000	0.556	1.000	1.000	1.000	1.000	0
0.100	0.222	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.100	0.000	0.000	0.000	0.000	0.100	0.111	0.000	0.000	1
0.300	0.000	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.500	0.222	0.222	0.222	0.222	0.200	0.556	0.000	0.000	0
0.600	0.000	0.111	0.222	0.111	0.100	0.111	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0

0.400	0.000	0.000	0.111	0.000	0.100	0.111	0.000	0.000	1
0.200	0.000	0.222	0.000	0.222	0.400	0.000	0.000	0.000	0
0.300	0.556	0.556	0.444	0.667	0.600	0.667	0.667	0.222	0
0.100	0.000	0.000	0.000	0.111	0.500	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.500	0.111	0.222	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.700	0.667	0.333	0.333	0.444	0.300	0.444	1.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.333	0.000	0.111	0.100	0.000	0.000	0.000	1
0.900	1.000	0.667	0.778	0.667	0.100	1.000	1.000	0.222	0
0.300	0.111	0.333	0.222	0.111	0.200	0.111	0.000	0.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.400	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.300	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.000	0.111	0.111	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.400	1.000	1.000	1.000	1.000	0.200	1.000	1.000	1.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.200	0.000	0.000	0.111	0.222	0.400	0.000	0.000	0.000	1
0.000	0.111	0.000	0.222	0.111	0.100	0.111	0.000	0.000	1
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.111	0.000	1
0.300	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.400	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	1
0.400	0.333	0.444	0.000	0.778	0.100	0.222	0.556	0.000	1
0.600	0.778	0.778	0.667	0.222	1.000	0.667	0.111	0.222	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	1
1.000	0.000	0.222	0.000	0.111	0.100	0.111	0.000	0.000	1
0.000	0.000	0.222	0.000	0.111	0.100	0.111	0.000	0.000	0
0.200	0.000	0.000	0.222	0.111	0.100	0.111	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	0.111	0.111	0.111	0.111	0.100	0.000	0.000	0.111	1
0.200	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	1
0.400	0.667	0.333	0.000	0.556	0.100	0.667	1.000	0.222	0
0.400	1.000	1.000	0.778	0.444	0.500	0.667	1.000	0.000	0

0.200	1.000	0.667	0.778	0.444	0.800	0.667	0.333	0.000	1
0.200	0.111	0.000	0.111	0.111	0.100	0.222	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.222	0.000	0.000	1
0.400	0.222	0.111	0.000	0.222	0.100	0.000	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.000	0
0.300	0.000	0.333	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.111	0.000	0.111	0.100	0.111	0.000	0.000	0
0.400	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.900	1.000	1.000	1.000	0.444	1.000	1.000	1.000	0.667	0
0.400	1.000	1.000	1.000	0.333	1.000	0.444	0.556	0.222	0
0.400	0.000	0.000	0.000	0.111	0.100	0.222	0.111	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	1
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.222	0.000	0
0.300	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.000	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.778	0
0.000	0.000	0.000	0.222	0.111	0.100	0.000	0.000	0.000	0
0.400	1.000	1.000	0.444	0.333	0.500	0.333	0.333	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.200	0.000	0.000	0.000	0.111	0.100	0.111	0.000	0.111	0
0.200	0.000	0.000	0.000	0.222	0.200	0.000	0.000	0.000	0
0.100	0.000	0.000	0.000	0.111	0.100	0.000	0.000	0.000	0
0.400	1.000	1.000	0.222	0.667	0.300	0.778	1.000	0.111	0
0.300	0.778	0.556	0.333	0.222	0.400	1.000	0.556	0.000	1
0.300	0.778	0.778	0.444	0.333	0.500	1.000	0.333	0.000	1

Cancer output reference:

0 = Benign
1 = Malignant

APPENDIX C

Normalized Iris Dataset

Input pattern				Output pattern		
Petal_width	Petal_length	Sepal_width	Sepal_length	Species_name		
0.306	0.708	0.085	0.042	1	0	0
0.139	0.583	0.102	0.042	1	0	0
0.139	0.417	0.068	0.000	1	0	0
0.000	0.417	0.017	0.000	1	0	0
0.417	0.833	0.034	0.042	1	0	0
0.389	1.000	0.085	0.125	1	0	0
0.306	0.792	0.051	0.125	1	0	0
0.222	0.625	0.068	0.083	1	0	0
0.389	0.750	0.119	0.083	1	0	0
0.222	0.750	0.085	0.083	1	0	0
0.306	0.583	0.119	0.042	1	0	0
0.222	0.708	0.085	0.125	1	0	0
0.083	0.667	0.000	0.042	1	0	0
0.222	0.542	0.119	0.167	1	0	0
0.139	0.583	0.153	0.042	1	0	0
0.194	0.417	0.102	0.042	1	0	0
0.194	0.583	0.102	0.125	1	0	0
0.250	0.625	0.085	0.042	1	0	0
0.250	0.583	0.068	0.042	1	0	0
0.111	0.500	0.102	0.042	1	0	0
0.139	0.458	0.102	0.042	1	0	0
0.306	0.583	0.085	0.125	1	0	0
0.250	0.875	0.085	0.000	1	0	0
0.333	0.917	0.068	0.042	1	0	0
0.167	0.458	0.085	0.000	1	0	0
0.194	0.500	0.034	0.042	1	0	0
0.333	0.625	0.051	0.042	1	0	0

0.167	0.458	0.085	0.000	1	0	0
0.028	0.417	0.051	0.042	1	0	0
0.222	0.583	0.085	0.042	1	0	0
0.194	0.625	0.051	0.083	1	0	0
0.056	0.125	0.051	0.083	1	0	0
0.028	0.500	0.051	0.042	1	0	0
0.194	0.625	0.102	0.208	1	0	0
0.222	0.750	0.153	0.125	1	0	0
0.139	0.417	0.068	0.083	1	0	0
0.222	0.750	0.102	0.042	1	0	0
0.083	0.500	0.068	0.042	1	0	0
0.278	0.708	0.085	0.042	1	0	0
0.194	0.542	0.068	0.042	1	0	0
0.222	0.625	0.068	0.042	1	0	0
0.167	0.417	0.068	0.042	1	0	0
0.111	0.500	0.051	0.042	1	0	0
0.083	0.458	0.085	0.042	1	0	0
0.194	0.667	0.068	0.042	1	0	0
0.306	0.792	0.119	0.125	1	0	0
0.083	0.583	0.068	0.083	1	0	0
0.194	0.583	0.085	0.042	1	0	0
0.028	0.375	0.068	0.042	1	0	0
0.167	0.458	0.085	0.000	1	0	0
0.194	0.000	0.424	0.375	0	1	0
0.444	0.417	0.542	0.583	0	1	0
0.472	0.083	0.508	0.375	0	1	0
0.500	0.375	0.627	0.542	0	1	0
0.361	0.375	0.441	0.500	0	1	0
0.667	0.458	0.576	0.542	0	1	0
0.361	0.417	0.593	0.583	0	1	0
0.417	0.292	0.525	0.375	0	1	0
0.528	0.083	0.593	0.583	0	1	0
0.361	0.208	0.492	0.417	0	1	0
0.444	0.500	0.644	0.708	0	1	0
0.500	0.333	0.508	0.500	0	1	0
0.556	0.208	0.661	0.583	0	1	0
0.500	0.333	0.627	0.458	0	1	0
0.583	0.375	0.559	0.500	0	1	0
0.639	0.417	0.576	0.542	0	1	0
0.694	0.333	0.644	0.542	0	1	0
0.667	0.417	0.678	0.667	0	1	0
0.472	0.375	0.593	0.583	0	1	0
0.389	0.250	0.424	0.375	0	1	0
0.333	0.167	0.475	0.417	0	1	0
0.333	0.167	0.458	0.375	0	1	0
0.417	0.292	0.492	0.458	0	1	0

0.472	0.292	0.695	0.625	0	1	0
0.306	0.417	0.593	0.583	0	1	0
0.472	0.583	0.593	0.625	0	1	0
0.667	0.458	0.627	0.583	0	1	0
0.556	0.125	0.576	0.500	0	1	0
0.361	0.417	0.525	0.500	0	1	0
0.333	0.208	0.508	0.500	0	1	0
0.333	0.250	0.576	0.458	0	1	0
0.500	0.417	0.610	0.542	0	1	0
0.417	0.250	0.508	0.458	0	1	0
0.194	0.125	0.390	0.375	0	1	0
0.361	0.292	0.542	0.500	0	1	0
0.389	0.417	0.542	0.458	0	1	0
0.389	0.375	0.542	0.500	0	1	0
0.528	0.375	0.559	0.500	0	1	0
0.222	0.208	0.339	0.417	0	1	0
0.389	0.333	0.525	0.500	0	1	0
0.750	0.500	0.627	0.542	0	1	0
0.583	0.500	0.593	0.583	0	1	0
0.722	0.458	0.661	0.583	0	1	0
0.333	0.125	0.508	0.500	0	1	0
0.611	0.333	0.610	0.583	0	1	0
0.389	0.333	0.593	0.500	0	1	0
0.556	0.542	0.627	0.625	0	1	0
0.167	0.167	0.390	0.375	0	1	0
0.639	0.375	0.610	0.500	0	1	0
0.250	0.292	0.492	0.542	0	1	0
0.611	0.500	0.695	0.792	0	0	1
0.583	0.292	0.729	0.750	0	0	1
0.694	0.417	0.763	0.833	0	0	1
0.389	0.208	0.678	0.792	0	0	1
0.417	0.333	0.695	0.958	0	0	1
0.583	0.500	0.729	0.917	0	0	1
0.611	0.417	0.763	0.708	0	0	1
0.944	0.750	0.966	0.875	0	0	1
0.944	0.250	1.000	0.917	0	0	1
0.472	0.083	0.678	0.583	0	0	1
0.722	0.500	0.797	0.917	0	0	1
0.361	0.333	0.661	0.792	0	0	1
0.944	0.333	0.966	0.792	0	0	1
0.556	0.292	0.661	0.708	0	0	1
0.667	0.542	0.797	0.833	0	0	1
0.806	0.500	0.847	0.708	0	0	1
0.528	0.333	0.644	0.708	0	0	1
0.500	0.417	0.661	0.708	0	0	1
0.583	0.333	0.780	0.833	0	0	1

0.806	0.417	0.814	0.625	0	0	1
0.861	0.333	0.864	0.750	0	0	1
1.000	0.750	0.915	0.792	0	0	1
0.583	0.333	0.780	0.875	0	0	1
0.556	0.333	0.695	0.583	0	0	1
0.500	0.250	0.780	0.542	0	0	1
0.944	0.417	0.864	0.917	0	0	1
0.556	0.583	0.780	0.958	0	0	1
0.583	0.458	0.763	0.708	0	0	1
0.472	0.417	0.644	0.708	0	0	1
0.722	0.458	0.746	0.833	0	0	1
0.667	0.458	0.780	0.958	0	0	1
0.722	0.458	0.695	0.917	0	0	1
0.417	0.292	0.695	0.750	0	0	1
0.694	0.500	0.831	0.917	0	0	1
0.667	0.542	0.797	1.000	0	0	1
0.667	0.417	0.712	0.917	0	0	1
0.556	0.208	0.678	0.750	0	0	1
0.611	0.417	0.712	0.792	0	0	1
0.528	0.583	0.746	0.917	0	0	1
0.444	0.417	0.695	0.708	0	0	1
0.556	0.542	0.847	1.000	0	0	1
0.417	0.292	0.695	0.750	0	0	1
0.778	0.417	0.831	0.833	0	0	1
0.556	0.375	0.780	0.708	0	0	1
0.611	0.417	0.814	0.875	0	0	1
0.917	0.417	0.949	0.833	0	0	1
0.167	0.208	0.593	0.667	0	0	1
0.833	0.375	0.898	0.708	0	0	1
0.667	0.208	0.814	0.708	0	0	1
0.806	0.667	0.864	1.000	0	0	1

Iris output pattern reference:

1-0-0 = Setosa

0-1-0 = Versicolor

0-0-1 = Vergininca

APPENDIX D

Normalized Heart Dataset

Input																						Output
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1
1	1	0	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	1

1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0

1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0

1	1	1	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	1	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	1	1	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0
1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1	1	0

0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0

Heart output reference:

0 = normal
1 = abnormal