# ADAPTING AND ENHANCING MUSSELS WANDERING OPTIMIZATION ALGORITHM FOR SUPERVISED TRAINING OF NEURAL NETWORKS

**AHMED A. A. ABUSNAINA**

**UNIVERSITI SAINS MALAYSIA**

**2015**

# ADAPTING AND ENHANCING MUSSELS WANDERING OPTIMIZATION ALGORITHM FOR SUPERVISED TRAINING OF NEURAL NETWORKS

**By**

**AHMED A. A. ABUSNAINA**

**Thesis submitted in fulfillment of the requirements of the degree of Doctor of Philosophy**

**April 2015**

﴿ يَا أَيُّهَا الَّذِينَ آمَنُوا اسْتَعِينُوا بِالصَّبْرِ وَ الصَّلَاةِ إِنَّ اللَّهَ مَعَ الصَّابِرِينَ﴾

(سورة البقرة ، آية : 153)

*To the soul of my brother*

*Salman*

## ACKNOWLEDGMENT

First and foremost, my sincere glorifications and adorations go to almighty Allah for his guidance, protection and strength over me to complete my PhD research study. I ask Allah to guide me always to the straight path.

Words alone cannot express the love and how grateful I am to my beloved parents. Thank you my mum and my dad for your encouragements, continuous support, patience, and for always being there for me. Your continuous pray “doaa” at day and night has made me strong in this life. Life would not have been as colorful without my brothers and sisters, thank you all.

My great thanks and gratitude go to my supervisor and mentor Prof. Rosni Abdullah. Her valuable advices, comments and feedback have enabled me to finish this thesis. My grateful and thanks are continued to my co-supervisor Dr. Ali Kattan for his support especially the presentation of this thesis. I have learned a great deal from their unique perspective on research, I will forever remain thankful for both of you.

My appreciation goes to the Universiti Sains Malaysia for its partial financial support. I would like to thank many people around me for their encouragements, especially Muhnnad AbuHashem, Muhammad Shehab, Ahmad Alsalibi, Muhammad Shambour, Salah Salem and Basem Alijla.

*Ahmed Abusnaina*

# TABLE OF CONTENTS

# LIST OF TABLES

**Page**

# LIST OF FIGURES

**Page**

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **$ACO_{\mathbb{R}}$** | ACO for continuous optimization |
| **ABC** | Artificial Bee Colony |
| **ACO** | Ant Colony Optimization |
| **ANN** | Artificial Neural Network. |
| **BP** | Back-propagation |
| **CACO** | Continuous-ACO |
| **CEP** | Classification Error Percentage |
| **CPSO** | Cooperative Particle Swarm Optimization |
| **DE** | Differential Evolution |
| **DPSO** | Discrete Particle Swarm Optimization |
| **E-MWO** | Enhanced-Mussels Wandering Optimization |
| **FP** | Floating point |
| **ES** | Evolutionary Strategy |
| **GA** | Genetic Algorithm |
| **GD** | Gradient-descent |
| **GE** | Genetic Evolution |
| **GP** | Genetic Programming |
| **GSO** | Group Search Optimizer |
| **HS** | Harmony Search |
| **HS-BtW** | Harmony Search with Best-to-Worst Ratio |
| **IHS** | Improvised Harmony Search |
| **LM** | Levenberg-Marquardt |
| **LSE** | Least Square Error |
| **MDPSO** | Multi-dimensional Particle Swarm Optimization |
| **MLP** | Multilayer Perceptron |
| **MPPSO** | Multi-phase Particle Swarm Optimization |
| **MSE** | Mean Squared Error |
| **MWO** | Mussels Wandering Optimization |

| | |
|---|---|
| **NN** | Neural Network |
| **NRMSE** | Normalized Root Mean Squared Error |
| **PARDE** | Parallel Differential Evolution |
| **PCCE** | Percentage of Correctly Classified Examples |
| **P-Metaheuristic** | Population-based Metaheuristic |
| **PSO** | Particle Swarm Optimization |
| **RMSE** | Root Mean Squared Error |
| **SNN** | Spiking Neural Network |
| **SPSO** | Stochastic Particle Swarm Optimization |
| **SRM** | Spike-Response Model |
| **$SRM_0$** | Simplified Spike-Response Model |
| **SSE** | Sum Squared Error |
| **XOR** | Exclusive-OR |
| **IF** | Integrate-and-Fire model |
| **IZ** | Izhikevich model |
| **HH** | Hodgkin-Huxely model |

# LIST OF PUBLICATIONS

Ahmed A. Abusnaina, Rosni Abdullah, Ali Kattan (2014) The Application of Mussels Wandering Optimization Algorithm for Spiking Neural Networks Training. 1st International Engineering Conference (IEC2014) On Developments in Civil & Computer Engineering Applications, pp. 197-204.

Ahmed A. Abusnaina, Rosni Abdullah, Ali Kattan (2014) Enhanced MWO Training Algorithm to Improve Classification Accuracy of Artificial Neural Networks. Proceedings of the First International Conference on Soft Computing and Data Mining (SCDM-2014), Springer International Publishing, pp. 183-194. [ISI]

Ahmed A. Abusnaina, Rosni Abdullah, (2014) Spiking Neuron Models: A Review. International Journal of Digital Content Technology and its Applications (JDCTA), Vol. 8, No. 3, pp. 14-21.

Ahmed A. Abusnaina, Rosni Abdullah (2013) Mussels Wandering Optimization Algorithm based training of Artificial Neural Networks for Pattern Classification. In: Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013, pp. 78-85.

# ADAPTASI DAN PENINGKATAN ALGORITMA PENGOPTIMUMAN PERAYAUAN KUPANG UNTUK LATIHAN PENYELIAAN BAGI RANGKAIAN NEURAL

## ABSTRAK

Membangunkan kaedah latihan yang cekap untuk Rangkaian Neural (NN) dalam mencapai kejituan yang tinggi adalah satu cabaran. Tambahan pula, latihan NN masih lagi memerlukan masa yang lama. Algoritma Pengoptimuman Perayauan Kupang (MWO) ialah satu algoritma pengoptimuman metaheuristik yang baru dan telah diinspirasikan secara ekologi oleh tingkah laku pegerakan kupang. Objektif utama bagi tesis ini adalah untuk mencapai prestasi yang terbaik dalam penumpuan masa latihan dan ketepatan pengelasan untuk pengelasan corak dengan mengusulkan kaedah latihan penyeliaan yang baru untuk Rangkaian Neural Buatan (ANN) yang berasaskan penggunaan algoritma MWO. Mempertingkatkan prestasi, terutamanya dalam kejituan pengelasan yang membawa kepada perkenalan versi MWO yang telah di adaptasi; dikenali sebagai algoritma Peningkatan-MWO (E-MWO). Kedua-dua algoritma asal dan algoritma MWO yang telah ditingkatkan, digunakan untuk latihan penyeliaan bagi Rangkaian Pakuan Neural (SNN). Kebaikan bagi kaedah-kaedah yang telah dicadangkan telah disahkan secara empirik dengan menggunakan permasalahan piawaian sebagai penanda aras, manakala perbandingan telah dilakukan terhadap kaedah-kaedah latihan yang lain. Keputusan telah menunjukkan kaedah yang berasaskan MWO adalah dalam purata 12 kali lebih cepat berbanding kaedah yang lain bagi masa latihan. Ketika pengelasan kejituan, MWO adalah setanding dengan kaedah-kaedah yang lain dengan purata kejituan pada 80.18%, manakala E-MWO telah mengatasi kaedah-kaedah yang lain dalam 3 permasalahan dan setara dengan kaedah-kaedah yang lain dalam 5 permasalahan yang seterusnya dengan menggunakan ANN dengan purata kejituan pada 86.0%. E-MWO telah mengatasi kaedah-kaedah lain dalam pengelasan 6 permasalahan daripada 8 permasalahan dengan menggunakan SNN sebagai ejen pengelasan.

# ADAPTING AND ENHANCING MUSSELS WANDERING OPTIMIZATION ALGORITHM FOR SUPERVISED TRAINING OF NEURAL NETWORKS


## ABSTRACT

Developing efficient training method for Neural Networks (NN) in terms of high accuracy is a challenge. In addition, training NN is still highly-time consuming. The Mussels Wandering Optimization (MWO) is a recent metaheuristic optimization algorithm inspired ecologically by mussels movement behavior. The major objective of this thesis is to achieve better performance in terms of convergence training time and classification accuracy for pattern classification by proposing new supervised training methods for Artificial Neural Networks (ANN) based on the MWO algorithm. Increasing the performance, especially in terms of classification accuracy led to an adapted version of the MWO; known as Enhanced-MWO (E-MWO) algorithm. Both the original and the enhanced MWO algorithms are then applied for supervised training of Spiking Neural Networks (SNN). The merits of the proposed methods are validated empirically by using a set of benchmark problems, whereas comparisons are conducted against other common rival training methods. Results show that the MWO-based methods are in average 12 folds faster than other rival methods in terms of training time. In terms of classification accuracy, the MWO is on par with other methods with average accuracy of 80.18%, while the E-MWO outperforms other rival methods in 3 problems significantly, and on par with the other methods in classifying the rest 5 problems using ANN with average accuracy of 86.0%. The E-MWO outperforms other methods significantly in classifying 6 problems out of 8 problems using SNN as a classifier.

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background

Neural Network (NN) is considered as a simplified mathematical approximation of biological neural networks in terms of structure and function (Haykin, 1999). NN has been used to perform a variety of tasks such as identification of objects and patterns, making decisions based on prior experiences and prediction of future events based on past experience (Lin, 2007; Bennett et al., 2013; Dhar et al., 2010). NNs are being used in many fields such as medicine, chemistry, gaming, engineering, industry and banking (Krenker, Bester, & Kos, 2011).

Neural networks are categorized according to their computational units (neurons) to three generations (Maass, 1997; Ghosh-Dastidar & Adeli, 2009b; Ghosh-Dastidar & Adeli, 2009c): (1) Networks based on McCulloch and Pitts neurons as computational units; the neurons have only digital inputs/outputs, (2) Networks based on computational units that apply an activation function with a continuous set of possible inputs/outputs values, this generation is formally known as Artificial Neural Network (ANN), (3) Networks which employ spiking neurons as computational units, these networks consider the precise firing times of neurons for information coding which is considered more realistic to the real biological neurons, this generation is called Spiking Neural Networks (SNN).

The ability to discover nonlinear relationships among inputs and outputs, data driven, and robustness for missing or inaccurate data are the main attributes of NN (Moon, 2012; Li & Ma, 2010; Bhattacharyya & Kim, 2010; Zhang, 2000). Basically the most

challenging aspects of NN are: the mechanism of learning (training algorithms) that adjusts the neurons' weights values to minimize the error function, and the mechanism of information flow that depends on the NN structure (Ghosh-Dastidar & Adeli, 2009a; Suraweera & Ranasinghe, 2008).

Generally there are mainly two paradigms for supervised training of NN: Gradient-descent (GD) and Population-based Metaheuristic (P-Metaheuristic). GD paradigm performs local search near the starting point of the initial solution, it uses the error gradient to descend the error surface. The GD paradigm has some shortcomings such as slow convergence and local minima (Silva, Pacifico, & Ludermir, 2011; Huang, Zhu, & Siew, 2004).

P-Metaheuristic training algorithms are proposed to overcome the weaknesses of GD algorithms (Kattan & Abdullah, 2011; Karaboga, Akay, & Ozturk, 2007). The key concept of P-Metaheuristic methods is generating multiple solutions to find the optimum (or near the optimum) solution and performing global search for the entire search space.

The P-Metaheuristic algorithms are inspired from various aspects in nature. Several algorithms are inspired from the biological processes in the living creatures, while other methods are inspired from social interactions among animals such as those shown in Fig. 1.1. Algorithms used for NN training includes Genetic algorithm (GA) (Dorsey, Johnson, & Mayer 1994; Gao, Lei, & He, 2005), Particle Swarm Optimization (PSO) (Van Wyk & Engelbrecht, 2010; Yu, Wang, & Xi, 2008), Artificial Colony Optimization (ACO) (Mavrovouniotis & Yang, 2014; Wei, 2007), Artificial Bee Colony (ABC) (Karaboga, Akay, & Ozturk, 2007), and Group Search Optimizer (GSO) (He, Wu, & Saunders, 2009a; He, Wu, & Saunders, 2009b).

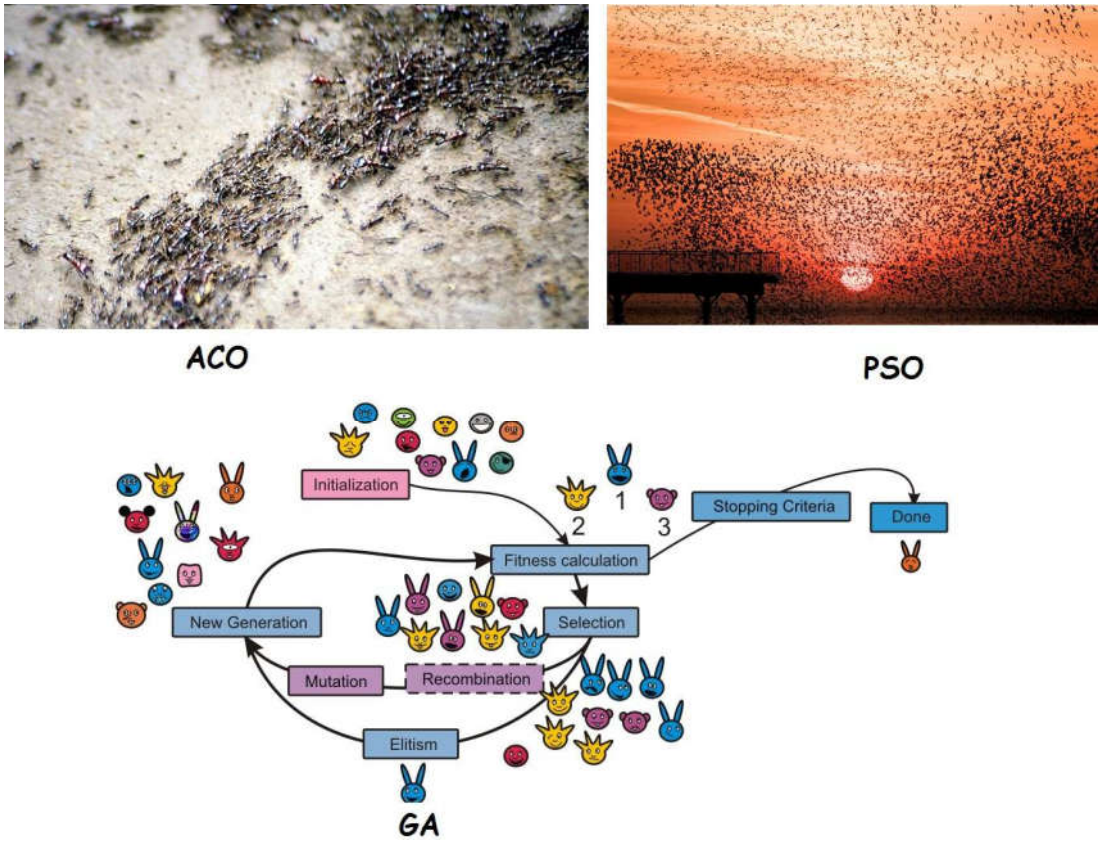


**Fig. 1.1 – Some of nature-inspired P-Metaheuristic algorithms.**

## 1.2 Research Problems and Motivations

Pattern classification is one of the main applications of NN, it is widely used in many fields such as medical diagnosis, biology, engineering, etc. Classification accuracy is the most commonly employed metric for evaluating classifiers and summarizes the overall performance (Da Silva et al., 2015). Developing efficient NN training method with high accuracy is a challenge (Mahmoud et al., 2013; Da Costa et al., 2004). In addition, it is highly-time consuming (Kiranyaz et al., 2009). The overall problem is training the NN (including ANN and SNN) in low (or reasonable) convergence time and gaining high classification accuracy.

The GD algorithms for training NN (e.g. Back-Propagation) have several drawbacks such as local minima, training oscillation, and long computational time (Silva,

Pacifico, & Ludermir, 2011; Huang, Zhu, & Siew, 2004). P-Metaheuristic might be more efficient by utilizing the exploration ability for the whole search space and obtaining an acceptable solution (Gilli & Winker, 2008; Manjarres et al., 2013). However, the majority of P-Metaheuristic algorithms for training NN were based on using simple logical operations (e.g. the classical XOR problem). Most of these algorithms were unable to generalize its superiority against others and number of compared optimization methods is too low to draw more general conclusions (Piotrowski et al., 2014; Kattan & Abdullah, 2011). Moreover, P-Metaheuristic algorithms might have higher computational cost (in terms of time and resources) and more complex structures (Song et al., 2012; Gilli & Winker, 2008). In addition, certain training algorithms are suitable for some type of classification problems (e.g. balanced-class or binary-class) only, also some algorithms cannot provide high classification accuracy (Su, Jhang, & Hou, 2008).

The Mussels Wandering Optimization (MWO) algorithm is one of the most recent and novel P-Metaheuristic optimization algorithms (An et al., 2013). The MWO is inspired ecologically by mussels' movement behaviour, whereas the stochastic decision and Le´vy walk are two techniques used to formulate a landscape-level of mussels' distribution.

The MWO is chosen to train the NN because of its ability to tackle the hard optimization problems. The MWO algorithm is highly dependent on stochastic (non-deterministic) processes; stochastic methods are more appropriate to tackle NP-hard problems (i.e. problems that have no known solutions in polynomial time) (Alijla et al., 2014). The MWO is simple as it depends only on primitive mathematics operations, and its parameters can be adjusted to fit any optimization problem.

Up to the existing knowledge and by reviewing the literature, the MWO has never been used for training NN. By applying, adapting, and enhancing the MWO-based algorithms, the diverse engineering, medical and scientific applications performance that use ANN and/or SNN might be improved and their use would become more efficient and reliable.

### 1.3 Research Questions

This research aims to answer and address the following research questions:

How the NN can be trained in short convergence time?

How the MWO can be applied for supervised training of NN?

How to increase the performance of the NN in terms of accuracy and time?

### 1.4 Research Objectives

This research has the following objectives:

1) To introduce a new method for supervised training of ANN by applying the MWO algorithm in order to reduce the training convergence time.

2) To enhance the MWO algorithm to achieve better performance for NN in terms of classification accuracy and time.

3) To investigate the application of the original MWO and the enhanced MWO algorithms for supervised training of SNN. The SNN usage as a classifier should satisfy the realistic behavior of biological neurons such as temporal encoding, synapse delay and information processing.

## 1.5 Research Scope

This research focuses on applying the MWO algorithm and its enhanced version (i.e. the E-MWO algorithm) for supervised training of fixed structure, fully connected, with single hidden layer of feed-forward Neural Network such as the network shown in Fig. 1.2. The proposed methods will address the supervised training of ANN and SNN.

As neural network has many applications, pattern classification is chosen as the application domain by considering several benchmark classification problems. For each benchmark problem, a network structure is designed in terms of number of neurons in each layer. Number of layers, neuron activation-function and network topology are fixed for all benchmarks and throughout all experiments. The proposed methods are compared against other existing training paradigms e.g. P-Metaheuristic and GD, whereas all experiments are performed by using the same computer.

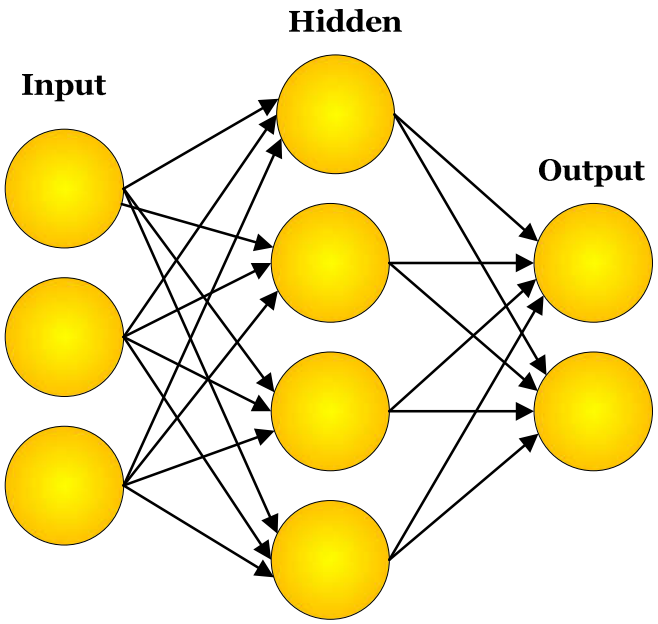


**Fig. 1.2 – Feed-forward Neural Network.**

## 1.6 Overview of Methodology

The research objectives will be achieved through four main phases. The first phase is reviewing the literature and identifying the research gap. Then one of main application of NN will be selected. Benchmark classification problems will be then determined and manipulated.

The second phase will consider the application of the MWO algorithm for supervised training of ANN. The performance of applying the MWO will be evaluated according to convergence training time and classification accuracy. The third phase will address the enhancements of the MWO algorithm. Then the E-MWO will be applied for training the NN. The performance of the E-MWO will be compared against the original MWO and other rival methods.

The final phase will tackle the supervised training of SNN in order to perform pattern classification. Firstly, SNN-based classifier will be designed, and then the MWO and the E-MWO algorithms will be applied to train the SNN. The overall performance of SNN for pattern classification will be evaluated according to convergence training time and classification accuracy, and will be compared against other rival methods.

## 1.7 Thesis Organization

This thesis is presented in seven chapters and organized as follows: Chapter Two presents the fundamental background of NN covering both generations; ANN and SNN, along with their common training methods. In addition, chapter two introduces the basics of the MWO algorithm. Chapter Three demonstrates the conducted research methodology and its main phases, describes the benchmarking problems and

the pre-processing manipulation for the datasets and illustrates the software and hardware specifications.

Chapter Four introduces the application of the MWO for training the ANN. Chapter Five proposes the enhancements that made for the MWO to achieve better performance. Chapter Six demonstrates how the SNN is used to for pattern classification, as well as it introduces the application of the MWO and the E-MWO for training the SNN. Chapter Seven gives an overall summary for the conducted research, the research findings and the feasible future works.

# CHAPTER TWO

# LITERATURE REVIEW

## 2.1 Overview

The work on neural networks goes back to the middle of 20th century, when McCulloch and Pitts (1943) created a physiological-neuron model with digital inputs and outputs which was motivated by a desire to understand the brain and to emulate some of its functions. The *perceptron* was developed as the next model of the neuron by Rosenblatt (1958), which can deal with continuous inputs and outputs that form the basis for the second generation of NN i.e. ANN. A desire for more realistic model that can mimic the biological neuron especially in its ability to process temporal information is the basis for the latest NN generation i.e. SNN by Maas (1997).

Since this research is based on the supervised training of NN, this chapter is divided into two parts. The first part presents the fundamentals of ANN and reviews its supervised training methods. The second part presents the fundamentals of SNN and reviews its supervised training methods. Fig. 2.1 shows the NN taxonomy, whereas related aspects of the research are shaded.

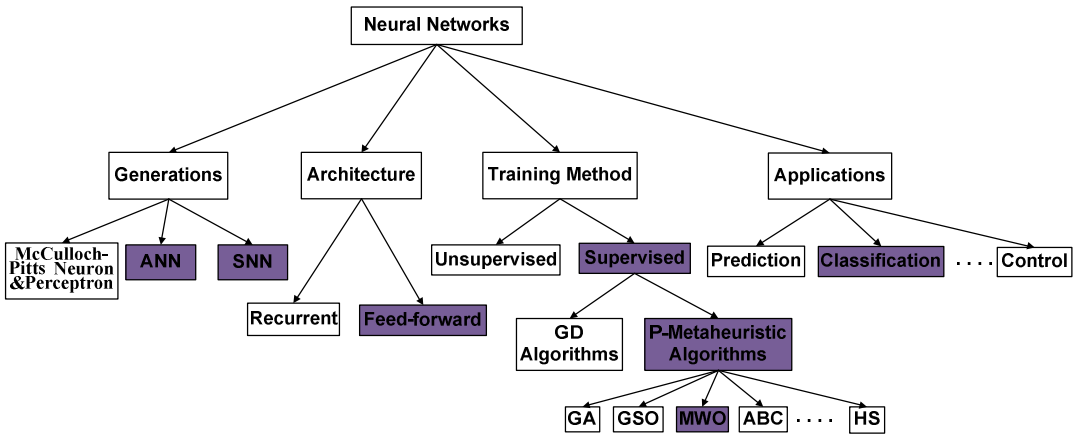


**Fig. 2.1 – Neural Networks Taxonomy.** (Shaded boxes denote to the research scope).

## 2.2 Artificial Neural Networks

The ANN is an interconnected group of processing units “artificial neurons” via a series of adjusted weights; these neurons use a mathematical model for information processing to accomplish a variety of tasks such as identification of objects and patterns, making decisions based on prior experiences and prediction of future events based on past experience (Lin, 2007; Bennett et al., 2013; Dhar et al., 2010).

ANNs are biologically inspired computer programs that can mimic the way in which the human brain processes information, it can be applied to determine a nonlinear relationship between set of factors by iterative training of data obtained from the environment (Agatonovic-Kustrin & Beresford, 2000; Ibrić et el., 2003).

The artificial neuron can learn from its given information and adapt to new facts using previous experience (Mahmud, Arafat, & Zuhori, 2012). As the biological neuron gather input signals from other sources, integrate them, performs a nonlinear process, and then outputs the result (Doreswamy & Vastrad, 2013). Fig. 2.2 shows a schematic of biological neuron versus artificial neuron.

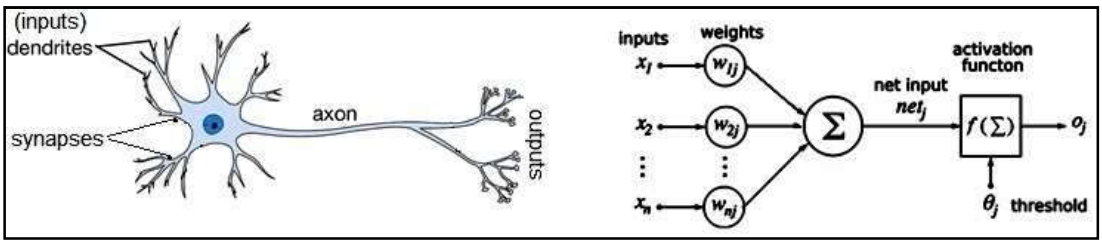


**Fig. 2.2 – Biological neuron Vs. artificial neuron.**

The ability to find out complex and nonlinear relationships among inputs and outputs is one of the main key characteristics of ANN, which they considered as universal functional approximators with arbitrary accuracy (Bhattacharyya & Kim, 2010; Zhang, 2000). In addition, ANN is data driven self-adaptive methods; it can be

adapted to any kind of data without any explicit specification for the underlying model, ANN also has robustness for missing or inaccurate data (Moon, 2012; Li & Ma, 2010; Bhattacharyya & Kim, 2010; Zhang, 2000). In case of hardware implementation, the ANN is able to work in parallel with input variables and therefore handle large sets of data rapidly. Also it can tolerate to any faults for any damage that might happen for the network-weights connections (Tehrani & Khodayar, 2010; Haykin, 1999).

Generally, Neural Networks distinguished by: (1) Architecture (topology); how the neurons connected to each other, (2) Learning method (training); the method of determining the network connections' weights', and (3) Activation function; the mathematical representation model of individual neurons (Fausett, 1994).

### 2.2.1 ANN Architecture

The architecture "structure" of NN deals with the connection way between neurons, as their inputs and outputs signals will be propagated to form the network. It plays a fundamental role in its functionality and performance (Fiesler & Beale, 1996). In general, NN is classified according to their connection way as feed-forward networks or recurrent networks (Zou, Han, & So, 2009; Bose, 2007; Du & Swamy, 2006).

The data in feed-forward networks flow from input nodes to output nodes strictly in forward direction (Negnevitsky, 2005; Kasabov, 1998). The processing of data can extend over multiple layers and units, but no feedback connections are allowed. The most popular network of such architecture is Multilayer Perceptron (MLP) which it is compatible with most training algorithm. However, the desired number of layers and number of neurons in each layer is an issue and sometimes selected by a trial-and-error process (Wilamowski, 2009).

### 2.2.2 Mathematical Representation of Neurons

The ANN derives its power from the nonlinear processing of its neurons (Samarasinghe, 2007). The neurons apply a linear/nonlinear activation function to the weighted sum of its input in order to limit the output of a neuron to some finite value (Negnevitsky, 2005). Usually, the activation function is same for all network neurons (De Blasio et al., 2012; Monjezi, Hesami, & Khandelwal, 2011). Various activation functions are given in Fig. 2.3.

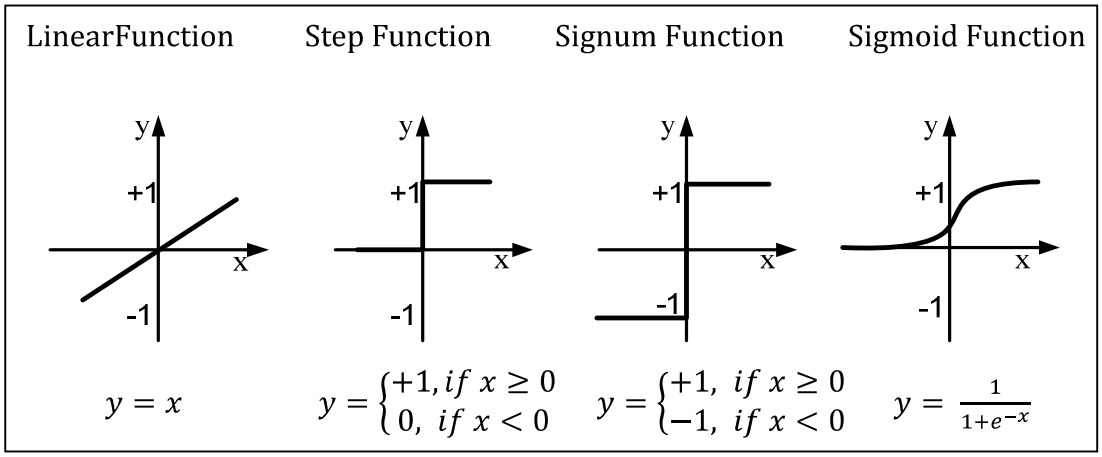


**Fig. 2.3 – Popular activation functions.**

### 2.2.3 Applications of ANN

ANN has many applications that used through wide variety of fields such as biology, psychology, medicine, marketing, computer vision, artificial intelligence and remote sensing. This subsection gives a brief overview about these applications.

Pattern classification is a very common application of ANN. Pattern classification is considered as the most frequently encountered decision making task of human activity (Afshar, Mosleh, & Kheyrandish, 2013; Zhang, 2000). Pattern classification is assigning an object to a predefined class; based on a number of feature attributes related to that object. It shows how machines can observe the environment and learn

to distinguish patterns and making reasonable decisions about the patterns categories (Basu, Bhattacharyya, & Kim, 2010; Zhang, 2000; Jain, Duin, & Mao, 2000).

Classifying the bacteria depending on its protein sequence was performed by using MLP and back-propagation algorithm (Banerjee et al., 2013), classifying protein based on their junctions was conducted using the ANN (Ogawa et al., 2013). Psychosocial risk factors were used as input neurons, and then ANN used to classify the pain intensity after the onset of treatment (Hallner & Hasenbring, 2004).

Shivakumar and Vijaya (2013) developed a system to categorize the human emotions depending on facial expressions, emotion class assignment is done by applying the extracted blocks as inputs to a feed-forward ANN trained by back-propagation algorithm. High-speed video for measurement of startle eye-blinks as a new augmentative modality is used as biometric security. Then, the NN used as classifier to the extracted features (Lovelace et al., 2009). ANN have been used in finance such as portfolio management, credit rating and predicting bankruptcy, forecasting exchange rates, predicting stock values, inflation and cash forecasting to accomplish a consistent decision-making process using reliable and scientific approaches (Li & Ma, 2010). Three-layer architecture of neural network has been used to learn predicting volatility of selected stocks (Fong et al., 2005).

### 2.3 Fundamentals of ANN Training

The training process of ANN (also it refers to learning) deals with adjusting and altering the weights and/or structure of the network depending on a specific training algorithm, in order to perform the required functionality (Zou, Han, & So, 2009; Heaton, 2008; Negnevitsky, 2005; Kasabov, 1998; Rojas, 1996). The training can be classified into two major categories: supervised training is a common method used

for training the feed-forward NN, while unsupervised training is commonly used for training the recurrent NN.

Supervised training involves providing the network with the training patterns set along with the anticipated target outputs (Negnevitsky, 2005). The training dataset is fed to the network repetitively in order to determine its outputs; the objective of this process generally is to minimize the ANN error between the actual and the desired outputs (Zou, Han, & So, 2009). Accordingly, the training is performed until the network learns to associate each input pattern to its corresponding desired output as illustrated in Fig. 2.4.

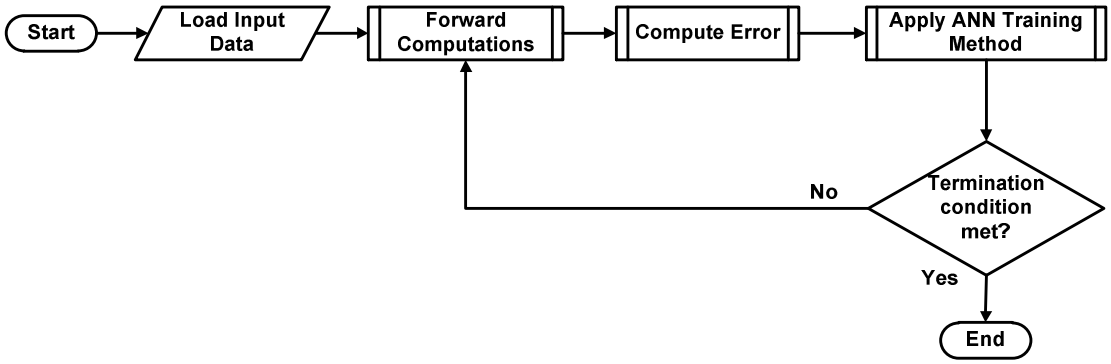


**Fig. 2.4 – Supervised Training of ANN.**

### 2.3.1 Error Calculations

The basic idea behind the ANN training is to minimize the error value between the actual and desired outputs by adjusting the weights between the network elements (Si, Hazra, & Jana, 2012). The error function is used in ANN to evaluate the performance of the network during the training process.

Several error functions have been used in the literature to perform the error calculations. The common used error function to report the network performance is the Sum Squared Error (SSE) (Abbasi & Mahlooji, 2012; Kattan, Abdullah, & Geem, 2011). However, other error functions are still used such as Mean Squared

Error (MSE) and Root Mean Squared Error (RMSE). Table 2.1 gives examples for popular and recent studies that use each of the error function in pattern classification. These error functions are dependent on the network application and design (Kattan, Abdullah, & Geem, 2011). It can be noted that some of studies use a combination of two error functions, e.g. Normalized Root Mean Squared Error (NRMSE) and CEP used by Yaghini et al. (2013), NRMSE and PCCE used by Rocha et al. (2007).

**Table 2.1 – Error calculation formulas.***

| Error Function | Formula | Description | Used by[1] |
|---|---|---|---|
| Sum Squared Error (SSE) | $SSE = \sum_{p=1}^{NP}\sum_{i=1}^{NO}\left(d_i^p - y_i^p\right)^2$ | Measures the sum squared difference between the desired and the actual output. | Zailah et al., 2013; Kulluk et al., 2012 ; Kattan, et al., 2010; Dorsey et al., 1994 |
| Least Square Error (LSE) | $LSE = \frac{1}{2}\sum_{p=1}^{NP}\sum_{i=1}^{NO}\left(d_i^p - y_i^p\right)^2$ | Measures the SSE, then multiplied by a factor of 0.5. | Wei, 2007; Svozil et al.,1997; Rumelhart, et al., 1986 |
| Mean Squared Error (MSE) | $MSE = \frac{1}{NP\times NO}\sum_{p=1}^{NP}\sum_{i=1}^{NO}\left(d_i^p - y_i^p\right)^2$ | Measures the SSE, then multiplied by a factor depends on the problem dimension itself. | Gonzalez & Vazquez, 2013 ; Widrow et al., 2013; Costa et al., 2009 |
| Root Mean Squared Error (RMSE) | $RMSE = \sqrt{\frac{1}{NP\times NO}\sum_{p=1}^{NP}\sum_{i=1}^{NO}\left(d_i^p - y_i^p\right)^2}$ | Returns the root square of the MSE. | Yaghini, et al., 2013; Mendes et al., 2002 |
| Normalized Root Mean Squared Error (NRMSE) | $NRMSE = \frac{RMSE}{\frac{\left(\sum_{p=1}^{NP}\sum_{i=1}^{NO}(d_i^p)\right)}{NP}} \times 100\%$ | The RMSE error value is squashed to the range [0, 1]. | Yaghini et al., 2013; Rocha et al., 2007 |
| Squared Error Percentage (SEP) | $SEP = 100\frac{y_{max} - y_{min}}{NP \times NO}\sum_{p=1}^{NP}\sum_{i=1}^{NO}\left(d_i^p - y_i^p\right)^2$ | Measures the SSE, then it multiplied by a factor depend on the max & min outputs and the problem dimension. | Alba & Chicano 2004; Islam et al., 2009 ; Ahmad et al., 2010 |
| Percentage of Correctly Classified Examples (PCCE) | $\varphi(p) = \begin{cases} 1, if\ \overrightarrow{d_p} = \overrightarrow{y_p} \\ 0, otherwise \end{cases}$ $PCCE = \frac{\sum_{i=1}^{NP}\varphi(p)}{NP} \times 100\%$ | Measures the percent of correctly classified patterns. | Rocha et al., 2007. |
| Classification Error Percentage (CEP) | $\varphi(p) = \begin{cases} 1, if\ \overrightarrow{d_p} \neq \overrightarrow{y_p} \\ 0, otherwise \end{cases}$ $CEP = \frac{\sum_{i=1}^{NP}\varphi(p)}{NP} \times 100\%$ | Measures the percent of incorrectly classified patterns. | Kattan & Abdullah, 2013; Yaghini et al., 2013; Ahmad et al., 2010 |

*where NP : total number of patterns. NO: number of output neurons. $d_i^p$:desired $i^{th}$ output of $p^{th}$ pattern. $y_i^p$:actual $i^{th}$ output of $p^{th}$ pattern. $\overrightarrow{d_p}$:desired output vector of $p^{th}$ pattern. $\overrightarrow{y_p}$:actual output vector of $p^{th}$ pattern.

[1] These are just examples for some of existed research uses the mentioned error calculations.

### 2.3.2 Termination Condition

The ANN training is an iterative process; it keeps running until a certain condition is satisfied. This condition (it can be called termination condition or termination criterion) depends on the training method and the ANN application. Some of training algorithms depend on the error function value, for example the training process stops when the error difference between the last two iterations is less than small value e.g. $error\ difference \leq 10^{-6}$, or the error satisfies a predefined value e.g. $error \leq 10^{-4}$.

Other algorithms depend on the number of iterations consumed by the training process e.g. *iteration number* = 10,000 (Samarasinghe, 2007; Du & Swamy, 2006; Haykin, 1999). Some of P-Metaheuristic algorithms may include a certain criterion depends on the dynamic quality measure of the solutions, e.g. Best-to-Worst ratio: $BtW_{termination} < 0.99$ in HS-BtW (Kattan & Abdullah, 2011). A combination of the aforementioned termination conditions also could be used as a termination criterion, e.g. *iteration number* and *error difference* (Ahmad, Abdullah, & Alghamdi, 2009).

### 2.3.3 Common Issues in ANN Training

There are several issues involved in supervised training of ANN. These are finding a globally optimal solution that avoids the local minima, converging to an optimal solution in a reasonable period of time, and validating the neural network by testing it against over-fitting and generalization ability (Dhar et al., 2010). The following subsections discuss the over-fitting, the under-fitting and the local-minima issues, respectively.

- ***Over-fitting***

Over-fitting problem (it also referred to over-training) is a phenomenon when the ANN loses its generalization ability to classify new patterns different from the

training patterns (Kasabov, 1998; Hassoun, 1995; Bishop, 1995; Emanuelsson, Nielsen, & Heijne, 1999). Even though the purpose of training is to reduce the error value as much as possible, but lessening it beyond a certain point might lead to overtraining (Maier & Dandy, 2000) as shown in Fig. 2.5.

- ***Under-fitting***

Under-fitting (under-training) is a case when the ANN incapable to capture the essential features of inputs to distinguish each pattern and map it to the correct class (Samarasinghe, 2007). In this case, the ANN model considered as very general and does not have the ability to recognize the underlying patterns in the dataset (Zhang, et al., 2008). Under-fitting is a result when small number of hidden neurons used in the hidden layer, so the network error is high (Mjalli, Al-Asheh & Alfadala, 2007).

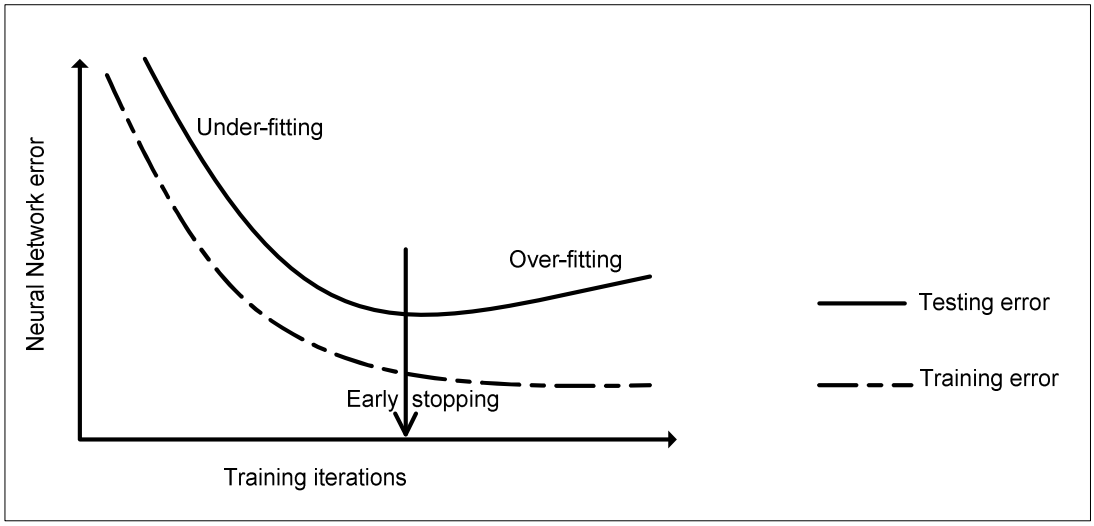


**Fig. 2.5 – Over and under fitting.**

- ***Local-minima***

The ANN training can be seen as searching for global minima on the error surface through the weight space (Sprinkhuizen-Kuyper & Boers, 1999). Local minima can be defined as the minimum closest point of a function compared to all its neighbours over a limited range of its parameters, whereas the global minimum is far from this

point (Negnevitsky, 2005; Ytlcetiirk et al., 1999) as illustrated in Fig. 2.6. If the training algorithm falls in local minima, the desired behaviour of the ANN may never be realized. Random weight initialization and using global optimization are suggested solutions for local minima (Atakulreka & Sutivong, 2007).

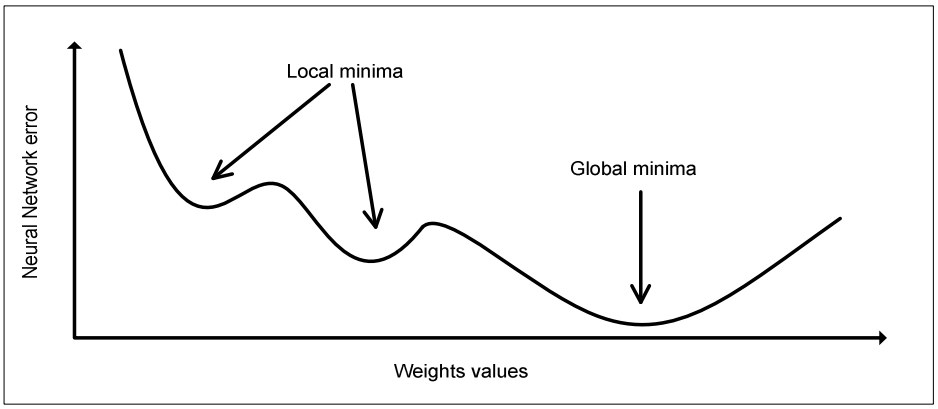


**Fig. 2.6 – Local minima problem.**

## 2.4 Existing Algorithms for Supervised Training of ANN

The supervised training methods of ANN fall into two main paradigms: Gradient-descent (GD) and Population-based Metaheuristic (P-Metaheuristic) training algorithms. GD paradigm uses the error gradient to descend the error surface. The derivative of the error function is computed in order to adjust the network weights. GD suffers from convergence slowness and local minima (Silva, Pacifico, & Ludermir, 2011; Huang, Zhu, & Siew, 2004). P-Metaheuristic training algorithms which depend on global optimization methods overcome the disadvantages of GD algorithms (Karaboga, Akay, & Ozturk, 2007; Kattan & Abdullah, 2011).

Other paradigms than GD and P-Metaheuristic exist in literature with the aim of training the ANN for the purpose of classification. Hybrid methods that combine the GD and P-Metaheuristic optimization have been proposed aiming to improve upon GD algorithms, such as GA with BP (Alba & Chicano, 2004) and PSO with BP

(Yaghini et al., 2013). However, the advantages of those methods are arguable (Bullinaria & AlYahya, 2014). Algorithms that use single-based metaheuristic are also utilized for training ANN, such as Simulated Annealing (Liang, 2007).

Cantu-Paz and Kamath (2005) have concluded that simple methods perform well and often better than more complex ones; networks trained with simple BP or simple GA were competitive with the most complex methods tested on classification problems by them. Therefore, other paradigms rather than GD or P-Metaheuristic are rarely used. Thus, detailed discussion of them is beyond the scope of this research.

### 2.4.1 Gradient-descent Paradigm

GD paradigm was the first proposed scheme to train the ANN. The algorithms that falls under this scheme utilize the directed search in which weights are always updated in such a way that minimizes the network error surface (Ince, et al. 2010). Typically, in GD paradigm the algorithms tends to minimize the error between the expected output and the generated network output values across large number of iterations (Joutsijoki et al., 2014; Mitra, Das, & Hayashi, 2011).

Such algorithms have several negative aspects such as dependency on a learning rate parameter, slowing down by an order of magnitude for every extra (hidden) layer added, the necessity of using only differentiable activation functions and the sensitivity to the initial values of the network parameters e.g. weight initialization (Gutiérrez & Hervás-Martínez, 2011; Yaghini, Khoshraftar, & Fallahi, 2013). The possibility of getting stuck in local minima is considered one of the main drawbacks of GD paradigm as the search is biased towards the locally optimal solution near the starting point (Ahmadi, Karamouz, & Moridi, 2010; Liang, 2007).

The most well-known algorithms from the category of GD paradigm are Back-Propagation (BP), Levenberg-Marquardt (LM), Conjugate Gradient and Quasi-Newton. A review of GD algorithms could be found in (Seiffert, 2006). A detailed discussion for each of GD algorithms is beyond the scope of this work. Since the BP and the LM are considered the most popular methods; the following subsections will introduce these methods.

### 2.4.1 (a) Back-Propagation

BP algorithm is widely used method for training feed-forward ANN, which has been used in many applications. BP has the great advantage of simple implementation (Che, Chiang, & Che, 2011). Basically, the standard BP was proposed by Rumelhart et al. (1986). Conceptually, the BP starts by applying the data to the network and performing the forward pass, and then it calculates the gradient error function of the outputs. After that, the error signal propagated in backward direction to hidden and input layers in order to adjust the network weights (backward pass). The network keeps training by the BP repeatedly until the total error falls to some predetermined very small value, then the training process is stopped.

The BP has several drawbacks; it depends on the initialization parameters and starting point (initial values of connection weights) in the solution domain. Therefore, the algorithm may be simply trapped in local minima problem. As well as, the BP suffers from the slow convergence for complex problems (Ahmad et al., 2010; Kathirvalavakumar & Thangavel, 2006). Finally, it cannot work with all neuron transfer functions; it has the necessity of using differentiable activation function (Kulluk, Ozbakir, & Baykasoglu, 2012).

Different variations of BP have been developed in order to overcome its drawbacks. These enhancements include: using BP with variable learning rate (Jacobs, 1988), speeding up the BP by adding the momentum factor (Miniani & Williams, 1990; Kamiyama et al., 1992), different weight initialization techniques to avoid local minima (Nguyen & Widrow, 1990; Fernández-Redondo & Hernández-Espinosa, 2001). The BP still has recent improvements, such as Yu et al. (2002), Kathirvalavakumar & Thangavel (2006) and Alejo et al. (2012).

### 2.4.1 (b) Levenberg-Marquardt

The Levenberg-Marquardt (LM) algorithm was developed by both Levenberg (Levenberg, 1944) and Marquardt (Marquardt, 1963), it is an iterative technique that finds the minimum of a multivariate function. LM provides a numerical solution to the problem of minimizing a nonlinear function (Liu, 2010; Yu & Wilamowski, 2011; Deossa et al., 2011). The LM algorithm has been used to train the ANN (Hagan & Menhaj, 1994; Liu, 2010; Wilamowski & Yu, 2010).

The LM combines the property of local convergence near a minimum point from Gauss-Newton method and the property of consistent error decrease that's provided by gradient-descent. Whenever the difference between the current solution and the correct solution is large; the algorithm behaves like a gradient-descent method (slow), however, it behaves like a Gauss-Newton method when the difference from correct solution is small (Liu, 2010; Ampazis & Perantonis, 2000). Some researchers argued that the LM outperformed the BP in terms of training time and accuracy (He, Sepehri, & Unbehauen, 2001; Ampazis, & Perantonis, 2000; Liu, 2010; Bascil & Temurtas, 2011). However, it is considered suitable and efficient algorithm for training small or median sized of ANN (Deossa et al., 2011).

The major drawback of LM algorithm is its large memory requirements, which arises from the demand to calculate the Jacobian matrix of the error function and the need to invert matrices with dimensions equal to the number of the NN weights'. The LM does not guarantee the convergence to the global minimum, since it depends on GD method. The LM suffers from local minima problem and the necessity of using differentiable activation function since it uses the second-order derivative (Yu & Wilamowski, 2011; Ampazis & Perantonis, 2000).

### 2.4.2 The P-Metaheuristic Paradigm

The P-Metaheuristic algorithms depend on global optimization and stochastic techniques. The P-Metaheuristic algorithms are employed for supervised training of ANN to overcome the drawbacks of GD learning algorithms (Karaboga, Akay, & Ozturk, 2007; Kattan & Abdullah, 2011). The P-Metaheuristic algorithms are inspired from various aspects in the real world such as biological processes in the living creatures or from social interactions among animals.

The training process of ANN deals with adjusting the weights and/or structure of the network depending on a specific training algorithm (Cantu-Paz & Kamath, 2005). The ANN weights search space is considered as continuous optimization problem, because it is high-dimensional and multimodal, also it could be corrupted by noises or missing data (He, Wu, & Saunders, 2009a; Karaboga, Akay, & Ozturk, 2007). The training is process of searching for suitable weights' values in the search space. The search progresses to new solutions by recombining or considering elements from different solutions in the population as shown in Fig. 2.7.

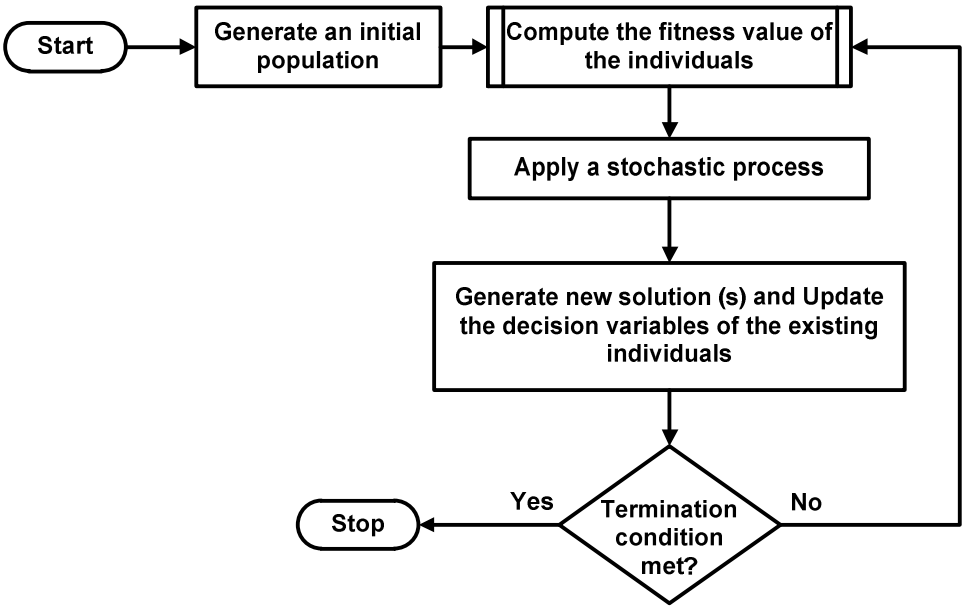


**Fig. 2.7 – General concept of the P-Metaheuristic algorithms.**

P-Metaheuristic might be more efficient by its exploration ability of the search space and obtaining an acceptable solution (Gilli & Winker, 2008; Manjarres et al., 2013). Unlike GD paradigm which considers the connection weights only to train the ANN. P-Metaheuristic algorithms can be used in all aspects of neural networks. P-Metaheuristic methods can be also used to find the suitable ANN structure, e.g. finding suitable number of hidden layer and finding suitable number of nodes in the hidden layer. Some P-Metaheuristic training methods evolve both the ANN structure and connection weights simultaneously. On the contrary to the GD methods, the algorithms in P-Metaheuristic can work on different and multiple regions of the solution space for the same problem simultaneously via a set of individuals that form the whole population (Crainic & Toulouse, 2010).

Training process of ANN is considered as a complex optimization problem (Chau, 2006) which it has been handled by Meta-heuristics algorithms. Nevertheless, most of these training algorithms were based on using the classical XOR problem, such as the method proposed by Karaboga et al. (2007). Hence, most of these algorithms

were unable to generalize its superiority against others (Kattan & Abdullah, 2011), because the training dataset size of XOR problem is too small. Furthermore, the XOR problem does not have local minima (Hamey, 1998). In addition, P-Metaheuristic algorithms might have higher computational cost and more complex structures (Song et al., 2012; Gilli & Winker, 2008), whereas the performance often depends on the algorithm settings and the problem characteristics.

The integration between the concept of supervised learning of the ANN illustrated earlier in Fig. 2.4 and the general concept of P-Metaheuristic algorithms given in Fig. 2.7 form the complete picture of the P-Metaheuristic paradigm for supervised training of ANN. The P-Metaheuristic paradigm has three essential issues that must be addressed for its application in supervised training: how the ANN can be represented through the algorithm, the fitness function, and when the algorithm should be terminated (Kattan, Abdullah, & Geem, 2011).

Two schemes are being used to represent the neural network through the P-Metaheuristic: vector-based scheme and matrix-based scheme (He, Wu, & Saunders, 2009b; Kattan, Abdullah, & Salam, 2010; Fish et al., 2004).

Given an ANN structure that has N of neurons, an upper triangular matrix W that represents the connection weights of the N neurons has a dimension N×N used in matrix-based scheme (Manrique, Rios, & Rodriguez-Paton, 2006), the element $w_{ij}$ of a weight matrix W refers to the connection strength from neuron $i$ to neuron $j$. By using Vector-based scheme, only the existed connections between any two neurons are included to form the vector of network weights (He, Wu & Saunders, 2009b; Kattan, Abdullah & Salam, 2010). The two schemes are illustrated in Fig. 2.8.

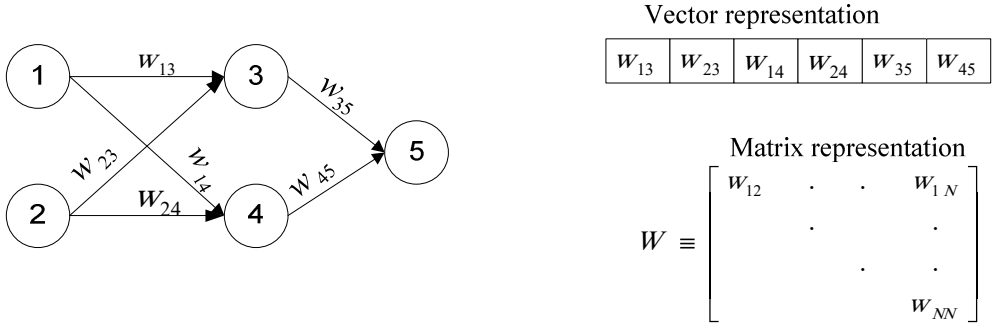


**Fig. 2.8 – Matrix Vs. Vector representation for ANN structure.**

### 2.4.2 (a) Genetic Algorithm

The Genetic Algorithm (GA) was firstly put forward and proposed by Holland (Holland, 1975). GA generates a sequence of candidate solutions to the underlying optimization problem by using a set of genetically inspired stochastic transition operators to transform the candidate solutions into a new offspring population (Nikolaev & Jacobson, 2010). The three most popular transition operators for generating new offspring are selection, crossover and mutation (Alba & Chicano, 2004; Lee & Geem, 2005). Selection or reproduction is the process of survival-of-the-fittest selection. Crossover is the partial swapping between two parent strings to produce a new offspring strings. Mutation is the occasional random inversion of bit values that generates non-recursive offspring.

The GA uses a population of strings, whereas these strings are often called chromosomes. As the GA is a global search optimizer, each string in the population is evaluated depending on the objective function, strings that have higher fitness have more chance to produce new solutions (Reeves, 2010; Hassoun, 1995).

The randomly initialized of chromosomes (population of strings) that represent the problem search space must be encoded first in order to be manipulated in GA (Alba & Chicano, 2004; Montana & Davis, 1989). Two encoding schemes existed: binary-