LOGIC PROGRAMMING IN RADIAL BASIS FUNCTION NEURAL NETWORKS

NAWAF HAMADNEH

UNIVERSITI SAINS MALAYSIA

2013

LOGIC PROGRAMMING IN RADIAL BASIS FUNCTION NEURAL NETWORKS

by

NAWAF HAMADNEH

Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosphy

November 2013

ACKNOWLEDGEMENTS

In the Name of Allah, the Beneficent, the Merciful

I ask Allah to have mercy on my father, who passed away a few months before my viva. I would like to say thanks my mother for here moral support. Without them, I may never have gotten to where I am today. I would like to express my deepest gratitude to my supervisor, Dr. Saratha Sathasivam, for all continuous guidance and advices given to me during my research work and writing up of this thesis. I would like to express my appreciation to my co-supervisor Assoc. Prof. Ong Hong Choon for giving me the encouragement and guidance during my research work and writing up of this thesis. I would like to thank Universiti Sains Malaysia, USM Graduate Assistant, for sponsoring me to pursue this Ph.D degree and thanks to the School of Mathematical Sciences for the financial support provided for publications. Special thanks to my wife and my daughters (Lamar and Sara) for their constantly support and sacrifice during my Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IPS Institut Pengajian Siswazah

PPSM Pusat Pengajian Sains Matematik

USM Universiti Sains Malaysia

ANNs: Artificial Neural Networks

AI: Artificial Intelligence

RBFNNs: Radial Basis Function Neural Networks

RRBFNNs: Recurrent Radial Basis Function Neural Networks

RBFs: Radial Basis Functions

MLPs: Multilayered Perceptron Neural Networks

SVM: Support Vector Machine

SOM: Self-Organizing Maps

SBC: Schwarz Bayesian Criterion

PSO algorithm: Particle Swarm Optimization algorithm

GA: Genetic algorithm

EBP algorithm: Error Back-Propagation algorithm

PPA: Prey-Predator Algorithm

TS: Training Set

DNF: Disjunctive Normal Form

CNF: Conjunction Normal Form

QBF: Quantified Boolean Formula

SAT: Satisfiability problem

MSE: Mean Square Error

RMSE: Root Mean Square Error

LOG-RBFNNs: Logic programming in Radial Basis Function Neural Networks

EBP-RBFNNs: EBP algorithm in RBFNNs

DPLL algorithm: Putnam Logemann Loveland algorithm

TSP: Traveling Salesman Problem

NP-problems: Nondeterministic polynomial time Problems

BH-RBFNNs method: The method of selecting the best number of hidden neurons in RBFNNs

LIST OF SYMBOLS

$\mathbb{R}$: The set of real numbers

$\mathbb{N}$: The set of natural numbers

w_{ij}: The output weight between the hidden neuron i and the output neuron j

w'_{ij}: The input weight between the input neuron i and the hidden neuron j

φ: Gaussian function

σ_i: The width of hidden neuron i

c_i: The center of hidden neuron i

$\|.\|$: Euclidean norm

T_P: Single-step operator

$\boldsymbol{T}$: True

$\bot$: False

$\wedge$: Conjunction

$\vee$: Disjunction

$\neg$: Negation

$\longleftarrow$: Implication

$\exists$: There exist

$\forall$: For all

PENGATURCARAAN LOGIK DALAM RANGKAIAN NEURAL FUNGSI ASAS RADIAL

ABSTRAK

Dalam tesis ini, saya telah membentuk teknik-teknik baru untuk mewakili pengaturcaraan logik dalam rangkaian neural fungsi asas radial. Dua teknik telah dibangunkan. Teknik yang pertama ialah untuk mengekod pengaturcaraan logik dalam rangkaian neural fungsi asas radial. Teknik yang kedua ialah untuk mewakili pengaturcaraan logik dengan mengira pengendali langkah pengaturcaraan logik dalam rangkaian neural fungsi asas radial. Saya menggunakan pelbagai jenis algoritma pengoptimuman untuk meningkatkan prestasi rangkaian neural. Saya menggunakan tiga teknik yang berbeza untuk meningkatkan keupayaan ramalan rangkaian neural. Teknik-teknik ini ialah: tidak ada latihan teknik, separuh teknik latihan dan teknik latihan penuh. Dalam tesis ini, saya telah membangunkan satu kaedah baru untuk menentukan bilangan yang terbaik daripada neuron tersembunyi dalam rangkaian neural fungsi asas radial. Untuk melakukan saya telah menggunakan fungsi ralat min punca kuasa dua dan kriteria Schwarz Bayesan sebagai pemilihan model. Saya menggunakan set data sebenar saiz yang berbeza dalam keputusan pengiraan. Analisis menunjukkan bahawa prestasi algoritma pengoptimuman sekumpulan zarah dan algoritma pemangsa mangsa yang lebih baik untuk digunakan dalam latihan rangkaian. Dalam tesis ini juga, saya telah membangunkan satu teknik baru untuk mendapatkan pengaturcaraan logik dari fungsi asas jejarian rangkaian neural. Untuk berbuat demikian, kita perlu mewujudkan asas jejarian fungsi rangkaian neural yang mewakili

bentuk normal sambung tiga (3 CNF) pengaturcaraan logik dengan menggunakan zarah sekumpulan pengoptimuman algoritma. Berikutan ini, kami telah melaksanakan keputusan kami untuk mewakili litar elektronik dalam asas jejarian fungsi rangkaian neural. Di samping itu, kami telah berjaya rangkaian neural fungsi asas radial untuk menyelesaikan masalah perjalanan jurujual, dan juga dalam Sistem Berasaskan Pengetahuan. Akhir sekali, kaedah baru tersebut adalah sesuai untuk digunakan dalam sains komputer, kecerdasan buatan, sains fizikal, dan beberapa masalah seperti masalah masa polinomial.

LOGIC PROGRAMMING IN RADIAL BASIS FUNCTION NEURAL NETWORKS

ABSTRACT

In this thesis, I established new techniques to represent logic programming in radial basis function neural networks. Two techniques were developed. The first technique is to encode the logic programming in radial basis function neural networks. The second technique is to compute the single step operator of logic programming in radial basis function neural networks. I used different types of optimization algorithms to improve the performance of the neural networks. I used three different techniques for improving the predictive capability of the neural networks. These techniques are: no-training technique, half training technique and full training technique. In this thesis, I established a new method for determining the best number of the hidden neurons in radial basis function neural networks. To do that I used the root mean square error function and Schwarz bayesian criterion as model selection criteria. I used real data sets of different sizes in the computational results. The analysis revealed that performance of particle swarm optimization algorithm and Prey predator algorithm are better to use in training the networks. In this thesis also, I developed a new technique to extract the logic programming from radial basis function neural networks. To do that, I established the radial basis function neural networks which represent the three conjunctive normal form (3-CNF) logic programming. Following this, I implemented the results to represent the electronic circuits in the radial basis function neural networks. In addition, I used successfully the radial basis function neural networks

to solve traveling salesman problem, and also in Knowledge-Based Systems. Finally, the new methods are suitable to use in computer science, artificial intelligence, physical sciences, and some problems such as nondeterministic polynomial time problems (NP-problems).

CHAPTER 1

INTRODUCTION

This thesis introduces new approaches to integrate logic programming in radial basis function neural networks. In this chapter, we review the artificial neural networks, artificial intelligent and related work. Beside this, I provide reasons for the study of neural-symbolic integration and the structure of this thesis.

1.1 INTRODUCTION

1.1.1 Artificial neural networks and artificial intelligent

The human brain is a biological neural network, containing about 100 billion neurons (Gopal, 2002). Each neuron is composed of a cell body (soma), dendrites that receive input, and axons that send output (Jain et al., 1996; Samardak et al., 2009), as shown in figure 1.1. Synapses separate axons from dendrites, are used to transfer information between neurons. Neurons "fire" signals through axons (output) when thresholds Synapse's exceeded. Artificial Neural Networks (ANNs) simulate neural networks found in nature, such as the human brain. The term artificial is used to distinguish artificial neural networks (ANNs) from their biological counterparts.

Intelligence in human is the ability to think and understand instead of doing things by instinct or automatically. Marvin Lee Minsky (Minsky, 1969) concisely defines artificial intelligence (AI) as "The science of making machines do things that would require intelligence if done by men". There are many topics of AI such as

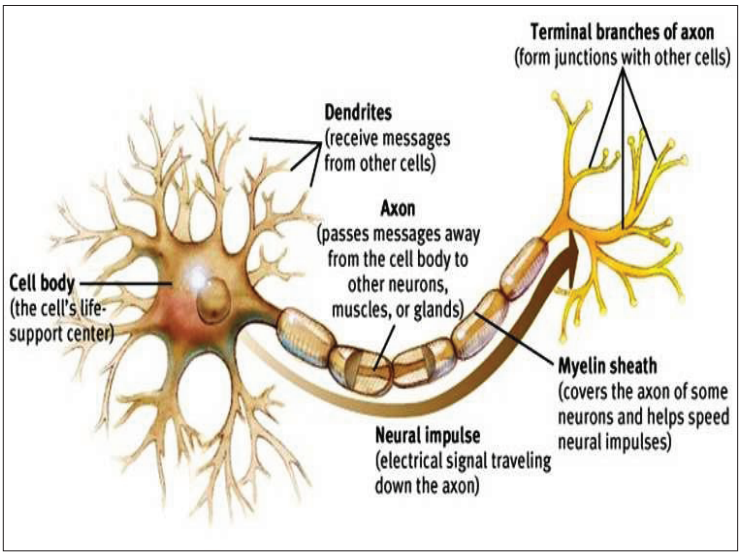

Figure 1.1: A structure of biological neuron

Neural-symbolic systems (Garcez et al., 2008), since the Neural-symbolic systems are a realization of symbolic processes within artificial neural networks (McCarthy, 1956, 1963; Haugeland, 1989; Negnevitsky, 2005). So, ANNs are a type of AI. Some questions can be asked in AI; How to get computers to make decisions? How to get computers to do correction / instruction? Furthermore, practical reasoning should be at the center of neural-symbolic integration research (see e.g.(Bose and Liang, 1996; Borges et al., 2007).

The artificial neural network is one of the AI applications. It has recently been used widely to model human interesting activities in many scopes. The discipline of neural networks is well developed with wide applications in almost all areas of science. A neural network is a black box that has clearly learned the internal relations of unknown systems, and include some mathematical and graphical models. There are two types of ANNs, feedforward neural-network, and feedback neural-network. ANNs are interconnected groups of artificial neurons that use a mathematical model for information processing based on a connectionist approach to computation. In other

words, ANNs are made of interconnecting artificial neurons, which may share some properties of biological neural networks. ANNs have received particular attention of researchers because of their ability to analyze complex nonlinear data sets (Hopfield, 1982; Moody and Darken, 1989). All neural networks perform essentially the same function, mainly vector mapping. It consists on a large number of simple processing elements called neurons. Each neuron is connected to other neurons by direct communication links, each with an associated weight. The main difference between ANNs and normal computational function is that ANNs process information in parallel, rather than, as with conventional computing, each task being broken down into discrete subtasks and processed sequentially. This provides ANN's with a level of flexibility in data processing that other computers and software simply do not have.

1.1.2 History of artificial neural networks

The basic neuron model is the McCulloch-Pitts neural model or linear threshold gate, which proposed by McCulloch and Pitts (1943). The flows of the neuron in the McCulloch-Pitts neural model are similar to the processes involved in the biological neurons. The McCulloch-Pitts neural model computes a weighted sum of its inputs from the other neurons. The following equation represents the model.

$$sum = \sum_{i=1}^{N} w_i x_i \tag{1.1}$$

$$y = \begin{cases} 1 & \text{if } sum > 0 \\ 0 & \text{if } sum < 0 \end{cases} \tag{1.2}$$

where x_i is an input point, w_i is a synaptic weight, and y is a binary output data point. N is the dimension of the input vector $x = \{x_1, x_2, \ldots, x_N\}$.

Warren McCulloch and Walter Pitts show that simple logical connectives such as conjunction, disjunction and negation can easily be encoded using binary threshold units.

Hebb (1949) developed the first learning rule, that is if two neurons are active in the same time, then the weight (the strength) between them should be increased. After that, many researchers (Rosenblatt, 1958; Minsky, 1969) showed that, the learning algorithm which was used in the perceptron is more powerful than the Hebb rule. Hopfield (1982) proposed a recurrent neural network, called Hopfield model, with symmetric synaptic connection (Sathasivam et al., 2011). Kohonen (1982) proposed a novel structure of ANNs, namely self-organizing maps (SOM) which use a one of the two-dimensional lattice structures. The first support vector machine (SVM) was invented by Boser et al. (1992). A historical criticism about neural networks was raised by McCarthy (1988). He referred to neural networks as having a "propositional fixation". Since then, several approaches have dealt with first-order logic in neural networks (Browne and Sun, 2001; Bader and Hitzler, 2004).

1.2 Related work

Towell and Shavlik (1993) shed light into ANNs by combining symbols. They use background knowledge in the form of propositional rules and encode these rules in multilayered perceptron neural networks (MLPs). Hölldobler and Kalinke (1994) proposed a new method to encode propositional logic programs by feedforward neural

networks with threshold activation functions. The method is called the core method, where the method is to represent logic programs by representing their associated immediate consequence operators. The idea behind the core method is computing or approximating the single step function within feedforward neural networks (Hölldobler and Kalinke, 1994; Hitzler et al., 2004). A general approximation theorem is of central importance for their approach, because of Funahashi's theorem (Funahashi, 1989). Since the Funahashi's theorem states that: Every continuous function on a compact subset of R can be uniformly approximated by MLPs with sigmoid units in the hidden layer (Funahashi, 1989). If the output layer of a network is connected to its input layer, then these recurrent networks allow for an iteration of the single step function leading to a stable state. Therefore, they show that the logic programming can be approximated arbitrarily well by MLPs, but do not specify any means for actually constructing them. In this thesis, I used radial basis function neural networks (RBFNNs) to represent logic programming, with the possibility of extracting logic programming from the neural networks.

Hölldobler and Kalinke (1994) also observe that these neural networks can be cast into a recurrent networks in order to simulate the iterative behavior of the T_P operator (Hölldobler and Kalinke, 1994). Hölldobler et al. (1999) study first-order logic programs and how to approximate their single-step operators by feedforward neural networks, and also reported in (Hitzler et al., 2004). The results from (Hitzler et al., 2004) guarantee the existence of recurrent neural networks with a feedforward network to approximate the a first-order logic programming. But it is by no means obvious how the rule extraction techniques can be generalized such that first-order rules are extracted from these networks. Blair et al. (1999) showed the intimate relationship between logic programming and dynamical systems related to self-similarity and chaos

theory. Garcez et al. (2002) extend the work by Hölldobler et al. (1999) to cover networks with sigmoidal activation functions. In this thesis, I used the Gaussian function as an activation function in RBFNNs.

The general approach to integrate logic programming in ANNs is illustrated in figure 1.2. The work is started with logic programming and next embed the logic programming in a neural network. The neural network can be then trained to determine the parameters. Finally, the acquired knowledge is extracted from the neural system. In this thesis, I used radial basis function neural networks (RBFNNs) to integrate logic programming in ANNs.

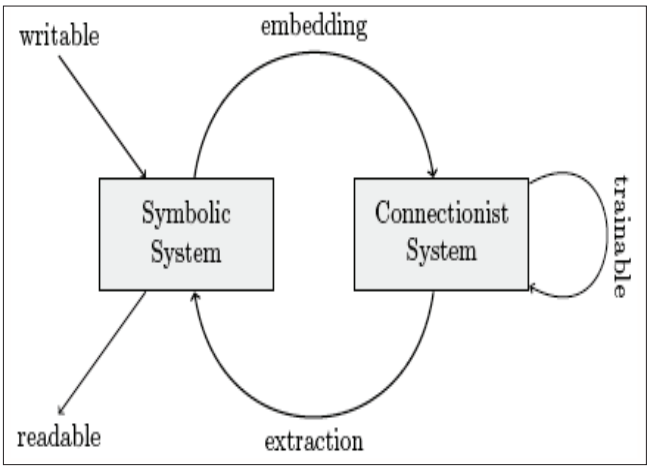

Figure 1.2: The neural symbolic cyclic

The most commonly used method for selecting the best number of hidden neurons in RBFNNs is by trial- and error (Maier and Dandy, 2000). In this method, the number of hidden neurons is systematically increased or decreased until the network with the best performance is found. The performance of the neural networks can be estimated by evaluating its testing data result using some goodness of fit measure, such as the root mean squared error (RMSE). In this thesis, I established a new algorithm using to determine the best number of the hidden neurons in RBFNNs.

1.3 Significance of research and problem statement

Several approaches that have been proposed for an integrated neural-symbolic system have made significant progress in the recent past. The question that arose naturally is how symbolic knowledge can be represented and dealt effectively within artificial neural networks. Recently, this question has been the fundamental challenge in computer science (Valiant, 2003). Our approach described the relation between RBFNNs and logic programming. Several researchers provide practical reasoning and explanation capabilities to the models integrated with logic programming and neural networks (Lloyd, 1987; Hölldobler and Kalinke, 1994; Bader and Hitzler, 2004).

MLP (Towell and Shavlik, 1993; Hölldobler and Kalinke, 1994; Hammer and Hitzler, 2007) is widely used neural network for an integrated neural symbolic system. In this thesis, I begun to examine the use of RBFNNs for an integrated neural symbolic system due to a number of drawbacks of MLP. MLPs sometimes don't converge or take too long to train to be useful in real-time applications (Tang and Kak, 2002). Although a MLP produces decision surfaces that effectively separate training examples of different classes, this does not necessarily result in the most plausible or robust classifier. The decision surfaces of MLP may not take on any intuitive shapes because regions of the input space not occupied by training data are classified arbitrarily, not according to proximity to training data. In addition, MLP have no mechanism to detect that a case to be classified has fallen into a region with no training data. This is a serious drawback since the power system operates within a wide range of system and fault conditions.

There are three basic elements in a RBFNN model (model). Since all these

elements can change independently to other, there is

i. Neural mode (model): is a structure use to solve a certain problem. At the beginning of the RBFNN building process, it is important to clearly define the criteria by which the performance of the network will judged as they can have a significant impact on the network architecture. Performance criteria include training speed and processing speed. In order to maximize processing speed, it is desirable to keep the number of the parameters as small as possible. For this, I established a new method for architecture selection used to determine the best number of the hidden neurons in RBFNNs. One of the main advantages of using a RBFNN over other approaches is that it does not require training the network with all possible combinations of input data. As well as also, the designing an integrated system using as RBFNN can bring benefits to their advantages. In addition, one of the advantages of RBFNNs, compared to MLP, is the possibility of choosing suitable parameters for the hidden neurons without having to perform a non-linear optimization of the network parameters (Ghodsi et al., 2003).

ii. Error function: it provides an evaluation about the quality of the model. In this thesis, I used the root mean square error (RMSE) to quantify the difference between actual values imposed by a neural network and the target values. RMSE incorporates both the standard deviation of the estimator and its bias. RMSE is suitable to compare forecasting errors of different models(Hyndman and Koehler, 2006).

iii. Neural Learning: The task of the neural learning algorithm is to obtain the parameters (the centers, the widths, and the output weights) of the neural model. Finding the action that optimizes (that is, maximizes or minimizes) the value

of an objective function, is the interest in determining the output level that maximizes profits. So, the goal may be to find the combination of parameters, which minimizes the cost of producing a desired level of output. There are many algorithms for solving optimization problems such as GA, PSO algorithm, EBP algorithm algorithm, and PPA. This study focuses on the use of optimization algorithms to find the combination of parameters that minimizes the objective functions in RBFNNs.

1.4 Objectives of research

The objective of this study is to introduce a mechanism using symbolic knowledge in radial basis function neural network. Ultimately, the goal is to produce biologically motivated models with integrated logic programming in radial basis function neural network. Two important areas in neural networks are optimization and validation. On the optimization front, efforts are directed towards building networks that are fast, small and efficient. On the validation front, the networks need to be functionally correct.

The objectives of this thesis are as follows:

i. To integrate the logic programming in the radial basis function neural networks. To do so, we proposed new techniques to embed logic programming in RBFNNs. We proposed a new equation to generate the training data set of logic programming. Then, we encode the logic programming in RBFNNs by using our new techniques. Finally, we trained these initial neural networks to determine its parameters.

ii. To use different types of learning algorithms for neural networks (neural learning algorithms). The reasons behind is to determine the values of a set of parameters in order to enhance the performance of the neural networks.

iii. To modify a new optimization algorithm (Prey predator algorithm) to use in training RBFNNs in order to improve the performance of the neural networks.

iv. To establish a new method for architecture selection. I established a new method for selecting the best number of the hidden neurons in RBFNNs, I called it 'BH-RBFNNs method'. To find the solution, I used PPA and then compared the results with grid search method.

v. To implement the results in many real life data.

1.5 Methodology

The action plan for this study are as follows:

i. Theory refinement-finding new theory or hypothesis related to logic programming and radial basis function neural networks.

ii. Developing a theoretical framework to embed logic programming in radial basis function neural networks.

iii. Implementation of theory in programming manner to verify the hypothesis developed for logic programming in radial basis function neural networks with computer simulation.

iv. Developing a new method for determining the best number of the hidden neurons in RBFNNs, and also modified the prey predator algorithm to train radial basis

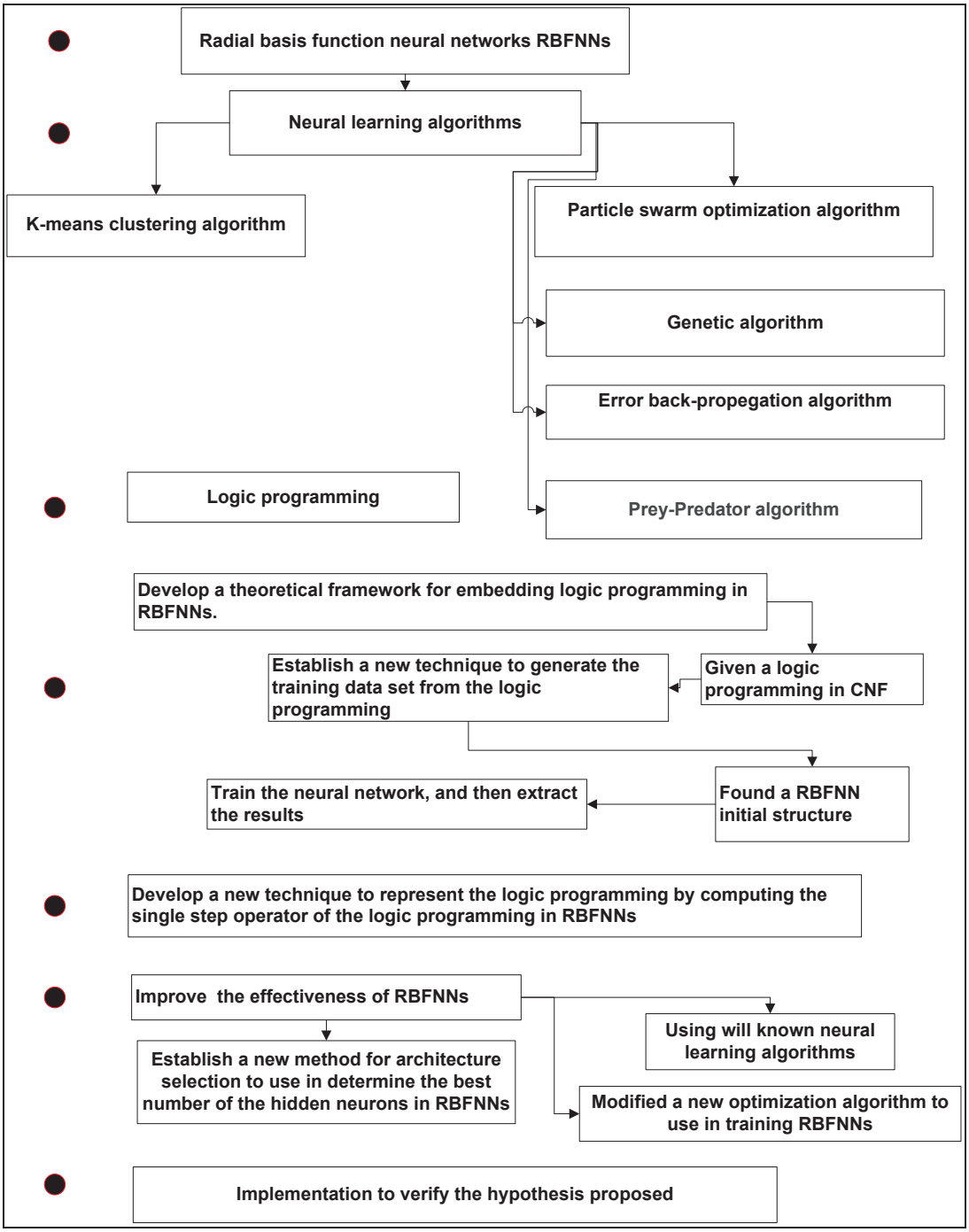

Figure 1.3: Research methodology flowchart

function neural networks.

Methodology of this thesis is explained in detail in Figure 1.3.

1.6 Organization of thesis

The rest of this thesis is organized as follows. In Chapter 2, I provide background on radial basis function network, and training algorithms. In Chapter 3, I provide background on logic programming. In Chapter 4, I describe the methods which I established to represent logic programming in radial basis function neural networks.

The first objective of Chapter 5 is to modified the prey predator algorithm to train the RBFNNs. The second objective of Chapter 5 is establish A new method for architecture selection, using to determine the best number of the hidden neurons in RBFNNs. The data sets which I have used are logic programming data sets, Iris data set, Thyroid data set, Breast Cancer Wisconsin (Original) data set, German Credit data set, Contraceptive Method Choice data set, Balance Scale data set, Haberman/s Survival data set, Hepatitis data set, Seeds data set, and Fertility data set. In addition, I showed that using full training technique in the training is better than half training.

In Chapter 6, I established new methods to represent Conjunctive normal form-logic programming in RBFNNs, to solve Quantified boolean formula QBF problems in RBFNNs, and to represent finite higher order logic programming in RBFNNs. Chapter 7 is about representing electronic circuits in RBFNNs, solving traveling salesman problem in RBFNNs, and representing the Knowledge Based Systems in RBFNNs. Finally, Chapter 8 gives the summary and suggestions for future research.

CHAPTER 2

RADIAL BASIS FUNCTION NEURAL NETWORKS

In this chapter, I review one of the most popular feed forward neural networks (radial basis function network) and their associated techniques. In addition, I surveyed some of optimization algorithms and K-means cluster algorithm.

2.1 Introduction

There are two main topologies in neural network according to their connectivity (Müller et al., 1995). The first one is feed forward networks, such as radial basis function neural network (Lowe, 1989; Garcez et al., 2002), if the neurons belonging to the same layer receive inputs from neurons of the previous layer and send their only to neurons of the next layer. The second one is the feedback networks,, such as Hopfield neural network (Hopfield, 1982), if the neurons belonging to the same layer send their output to neurons of the next and previous layers. The training involves adjusting the weights on the interconnections in the network until the error which is the difference between the actual output and the target output is small. Unsupervised learning mechanisms differ from supervised learning, where the differentiable characteristic of supervised ANNs lies in the inclusion of "a teacher" in their learning process, while unsupervised networks do not have "a teacher" in the training data set. Supervised learning is based on a direct comparison between the actual output and the target output. In other words, it is formulated as the minimization of an error function such

as the fitness function which is the mean absolute percentage error between the actual output and the target output summed over all available data. Unsupervised learning is solely based on the correlations among input data. No information on correct output is available for learning. The essence of a learning algorithm is the learning rule, i.e., a weight updating rule, which determines how connection weights are changed, for example, Hebbian rule. After the network is trained, it can be used to solve the problem at hand by merely presenting the inputs to the network. The corresponding value of the output is found virtually instantaneously as the inputs are propagated through the network. The success of ANNs lies in the fact that they can be trained using raw data, and in some problem domains (Garcez et al., 2002; Zhang and Sun, 2009). The generalization from the raw data made during the learning process turns out to be highly adequate for the problem at hand, even if the training data contains some noise. Note that, raw data are the data input for processing. ANNs can be applied to incomplete and fragmented data sets and also can understand and analyze incomplete data and nonlinear data. ANNs allow for "local" validation and prediction studies that would be costly and less effective using traditional methods.

Particle swarm optimization (PSO) algorithm is one of the most widely used algorithms to find the optimal values in order to optimize the expectation as a function in the neural networks. Several fields in engineering and computer science used PSO algorithm (Nenortaite and Butleris, 2008; Shi et al., 2001; Marinke et al., 2005). Genetic algorithm (GA) is also one of the well known and widely used metaheuristic solution algorithms for optimization problems (Ustun, 2007, Mishra and Patnaik, 2009). It has been used to solve many real problems, such as traveling salesman problem. The Error Back-Propagation (EBP) algorithm has been known to be using the gradient descent technique(Looney, 1997, Noman et al., 2009). Prey-Predator

algorithm (PPA) is a new metaheuristic algorithm (Tilahun, 2013). In the following sections, I will give an overview about RBFNNs and optimization algorithms.

2.2 Radial Basis Function Neural Network

A radial basis function neural network (RBFNN) is a neural network that was first used in 1989 (Moody and Darken, 1989; Lowe, 1989). It is addressed by viewing the design as an approximation problem in a high dimensional space. RBFNNs typically have three layers: an input layer, a hidden layer with a non-linear activation function and a linear output layer, as shown in figure 2.1.

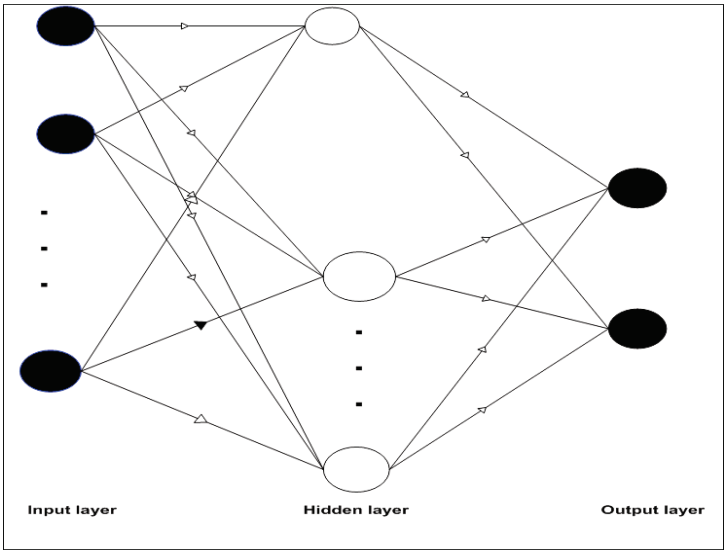

Figure 2.1: Structure of a radial basis function neural network with two output neurons

The input layer is made up of source neurons that connect the network to its environment. The hidden layer receives the input information, followed by certain decomposition, extraction, and transformation steps to generate the output data. The neurons in the hidden layer are associated with centers and widths that determine the behavioral structure of the network. The output layer provides the response of

the network to the activation pattern of the input layer that serves as a summation unit. Nevertheless, RBFNN is different from the other networks, possessing several distinctive features. Owing to its universal approximation ability, more compact topology and faster learning speed, it has attracted much attention, and it has been widely applied in many science and engineering fields (Lu and Ye, 2007; Kang and Jin, 2010; Zayandehroodi et al., 2010; Idri et al., 2010). The name, RBFNN, comes from the fact that the radial basis functions are radially symmetric. The radial basis functions are used in the hidden layer. Pursuant to that, each hidden neuron is allocated to respond to each of sub-spaces of the input regions, formed by the clusters of training samples (Vakil-Baghmisheh and Pavešić, 2004; Noman et al., 2009; Xiaobin, 2009).

NEURONS- The neurons are the processing units in a neural network. They also generate outputs according to the activation function. A neuron is linked to other neurons by variables, called the weights. Each layer in a neural network consists of a set of neurons.

RBFNN HAS PARAMETERS-The parameters of a RBFNN are the centers, the widths, the constant input weights, and the output weights, as shown in figure 2.2. The hidden layer parameters are the centers, and the widths. The output weights linked between hidden layer and output layer. The determination of the parameters, including the centers, the widths and the output weights is an important task (Vakil-Baghmisheh and Pavešić, 2004). So, the problem in RBFNNs is how to set the parameters in order to minimize an error criterion. In addition, the number of the hidden neurons is very important because it affects the network complexity and the generalizing capability of the network. Finally, the bias values which are added to each output are determined in the RBFNNs training procedure. In RBFNNs each neuron in the model inserts into the

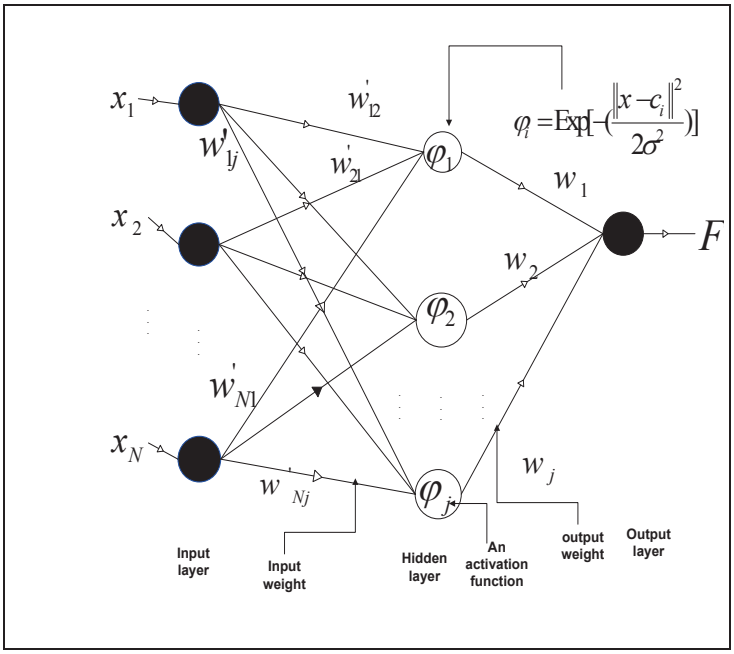

Figure 2.2: Structure of a radial basis function neural network

state space an open basin of attraction around its center, thus introducing a stable fixed point.

LEARNING PROCESSES - Learning at the hidden layer and the links between hidden layer and output layer, is commonly configured as the problem of finding these clusters and their parameters, as shown in figure 2.3 (Uykan and Koivo, 2005). The first phase in the learning procedure is to determine the hidden parameters (the centers and the widths) (Idri et al., 2010; Looney, 1997). The second phase involves output weights, which can be obtained by using the optimization algorithms. The optimization algorithms which we used in this thesis are PSO algorithm, GA, EBP algorithm, and PPA. The optimization algorithms use some of the different techniques, which are the no-training technique, the half-training technique, and the full-training technique (Schwenker et al., 2001; Uykan and Koivo, 2005).

OUTPUT LAYER - The output neurons represent a map that satisfies the

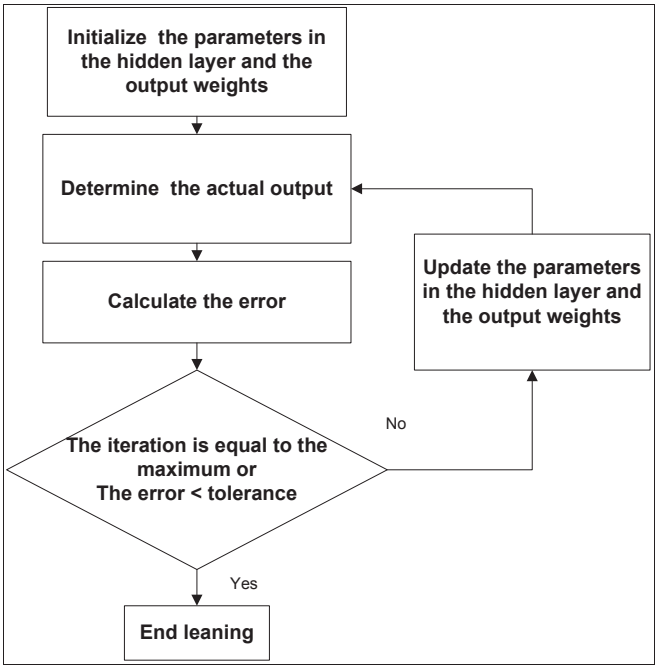

Figure 2.3: The processing on radial basis function neural networks

interpolation condition.

$$F(x_k) = d_k, k = 1, ..., R \tag{2.1}$$

where

- $\{x_k \in \mathbb{R}^N, k = 1, ..., R \in \mathbb{N}\}$ is the input data set.
- $\{d_k \in \mathbb{R}, k = 1, ..., R \in \mathbb{N}\}$ is a corresponding target output data set.
- $\{(x_k, d_k), k = 1, ..., R\}$ is called the training data set.

The interpolating function F has to pass through all the training data points.

Accordingly, the radial basis function neural network technique has the following form

$$F(x_k) = \sum_{i=1}^{j} w_i \varphi_i(x_k) \tag{2.2}$$

where

- $F(x_k)$ is the actual output value of an output neuron, which corresponds to the input value $x_k \in \mathbb{R}^N$. The Output actual value was encoded by the binary values $\{1,0\}$. The truth values represent by one, and the false values represented by zero. The difference between the actual values and the output target values can be obtained by using an error function such as the root mean square error (RMSE).

- w_i is the output weight between the hidden neuron i and the output neuron.

- j is the number of the hidden neurons.

- φ_i is a radial basis function in hidden neuron i. In this thesis the radial basis function used is the Gaussian function.

Accordingly, the following set of simultaneous linear equations (***Aw=D***, ***A*** is a non-singular matrix) can be obtained for the unknown coefficients (the output weights, ***wi, i=1,..., j***) for each output neuron, i.e. the dimensionality of the output is chosen as one. The output weights cab be solved by simple liner algebra (***w=Inverse[A]*D***)

$$\begin{pmatrix} \varphi_{\mathbf{1}}(\mathbf{x_1}) & \varphi_2(x_1) & \cdots & \varphi_j(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_j(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_R) & \varphi_2(x_R) & \cdots & \varphi_j(x_R) \end{pmatrix} \times \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_j \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_R \end{pmatrix} \tag{2.3}$$

where

- $\{x_k, k = 1, ..., R\}$ is the set of the values entered in the input layer.
- $\{w_i, i = 1, ..., j\}$: The output weights.
- $\{d_k, i = 1, ..., R\}$: is the set of target output values.
- $\{(x_k, d_k), k = 1, ..., R\}$ is called the training data set.

TRAINING DATA SET - The training set (Moody and Darken, 1989; Lowe, 1989) in RBFNNs are labeled pairs $\{(x_k, d_k), k = 1, ..., R\}$. It represents associations of a given a finite set of input-output data, where d_k is the output target value which corresponding to the input data x_k. So, to create a RBFNN, we need a training set to use in training the neural networks to calculate the parameters, including the centers, the widths and the output weights. In this thesis, we established a new technique used to calculate and fixed the training data set of the logic programming.

RADIAL BASIS FUNCTIONS - There are many types in radial basis function

such as Gaussian function, Multiquadric function and Polyharmonic Spline function (Ahmed, 2006; Park et al., 2012; Lewis et al., 2002). In order to classify the training data sets into (satisfactory data and unsatisfactory data), we can apply the following Gaussian function in RBFNNs (Idri et al., 2010).

$$\varphi_i(x) = \exp(-(\frac{\|\sum_{j=1}^{N} w'_{ji}x^j - c_i\|^2}{2\sigma_i^2})) \tag{2.4}$$

where

- φ_i is the radial basis function in hidden neuron i.
- w'_{ji} is the constant input weight between the input neuron j and the hidden neuron i.
- c_i and σ_i are the center and the width of the hidden neuron i respectively.
- N is the number of the input neurons.
- $x = (x^1, x^2, ..., x^N) \in \mathbb{R}^N$ is an input value in φ_i.
- $\|.\|$indicates the Euclidean norm on the input space as follows:

$$\|x - c\| = \sqrt{\sum_{m=1}^{N} (x^m - c^m)^2}, \quad x, c \in \mathbb{R}^N \tag{2.5}$$

We can apply the Gaussian function (φ) in RBFNNs, because of the following attributes. This can be observed in Figure 2.4 (Vakil-Baghmisheh and Pavešić, 2004).

- φ is continuous and bounded on real number, as shown in figure 2.4.

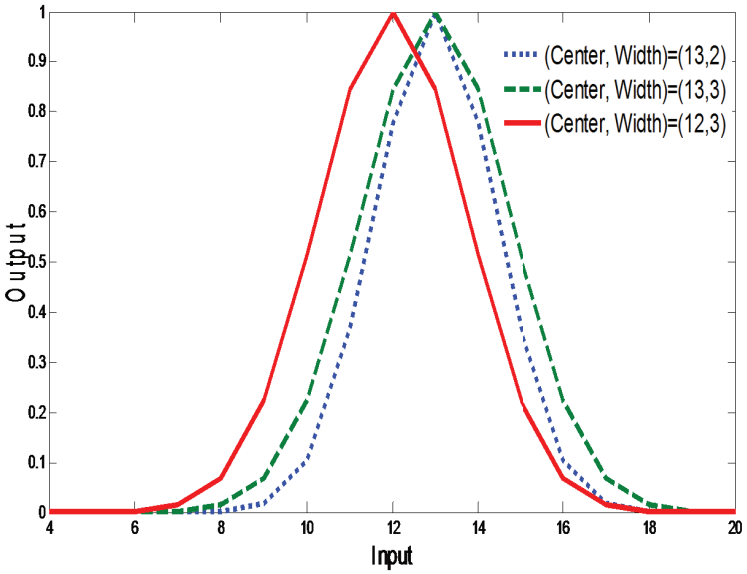

Figure 2.4: Gaussian function curves with different centers and widths.

- φ attains its unique maximum at the center (zero distance).

- φ has considerable values in the close neighborhood of the centers. That is because φ is continuous function and attains its unique maximum at the center.

- φ has negligible values in far distances (where are close to other centers). From figure 2.4, we can see that $\lim_{x\to\sigma^2} \varphi(x) = 0$

- Differentiability. The graph of a differentiable function must have a non-vertical tangent line from each point in its domain, and must be relatively smooth. This

is what has been achieved in φ, as shown in figure 2.4. Accordingly, φ is a differentiable function.

According to above discussion, in order to establish a RBFNN we have to answer the following questions (Sathasivam et al., 2011; Moody and Darken, 1989; Lowe, 1989; Rojas et al., 2000; González et al., 2003).

i. How many neurons will settle in the hidden layer? This question is necessary, because the hidden layer affects the network complexity and the generalizing capability of the network. So, the number of hidden neurons in RBFNNs should be as low as possible to obtain a reliable network. The most commonly used method for selecting the number of hidden neurons is by trial (Maier and Dandy, 2000). In this thesis, we established a new algorithm used to determine the best number of the hidden neurons.

ii. What are the values of the parameters? The networks's prediction errors must be minimized in order to achieve a well fitted neural network.

iii. What function will be used at the hidden neurons? In this thesis we used the Gaussian function as a radial basis function, because of its attributes.

2.3 Learning

Learning methods are used for the flexible neural networks. Two important areas in neural networks are optimization and validation. On the optimization aspect, the efforts are directed towards building networks that are efficient and fast. On the validation aspect, the networks need to be functionally correct. The adaptive neural

network approach was amenable to rule expression. Note that, learning in a neural network is called training. This section covers fundamental issues such as the different types of the training process in RBFNNs, neural learning algorithms, and k-means clustering. Performance aspects of the different learning process are discussed, as well as difficulties encountered in the learning process.

2.3.1 Training processes in radial basis function neural networks

Training methods can be applied to the RBFNN parameters in order to improve the performance of the network. Training process is used to determine the output weights, the centers and the widths of RBFNNs (Vakil-Baghmisheh and Pavešić, 2004; Dhubkarya et al., 2010). The combination of the parameters which minimizes the error function is considered to be a solution of the learning network. One of the most common measures of the training is the root mean square error (RMSE) function, where the error is the distance between the target values and the corresponding actual values, which will also be explained later.

As mentioned, there are three different techniques for training RBFNNs, as shown in table 2.1 (Dhubkarya et al., 2010; Xiaobin, 2009; Noman et al., 2009; Vakil-Baghmisheh and Pavešić, 2004).

- No-training: in this simplest case, all the parameters including the centers, the widths and the output weights, are calculated and fixed (no training is required). This paradigm does not have any practical value, because the number of the centers should be equal to the number of training data.

- Half-training (hybrid learning): this case involves two Phases. The first phase is