

University of Alberta

A Data Clustering Algorithm for Stratified Data Partitioning in Artificial
Neural Network

by

Ajit Kumar Sahoo

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Engineering Management

Department of Mechanical Engineering

©Ajit Kumar Sahoo
Spring 2011
Edmonton, Alberta

Permission is hereby granted to the University of Alberta Libraries to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only. Where the thesis is converted to, or otherwise made available in digital form, the University of Alberta will advise potential users of the thesis of these terms.

The author reserves all other publication and other rights in association with the copyright in the thesis and, except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author's prior written permission.

Examining Committee

Ming J. Zuo, Mechanical Engineering

Amit Kumar, Mechanical Engineering

Yasser Mohamed, Construction Engineering and Management

Dedicated to,

My family for their support

ABSTRACT

The statistical properties of training, validation and test data play an important role in assuring optimal performance in artificial neural networks (ANN). Researchers have proposed randomized data partitioning (RDP) and stratified data partitioning (SDP) methods for partition of input data into training, validation and test datasets. RDP methods based on genetic algorithm (GA) are computationally expensive as the random search space can be in the power of twenty or more for an average sized dataset. For SDP methods, clustering algorithms such as self organizing map (SOM) and fuzzy clustering (FC) are used to form strata. It is assumed that data points in any individual stratum are in close statistical agreement. Reported clustering algorithms are designed to form natural clusters. In the case of large multivariate datasets, some of these natural clusters can be big enough such that the furthest data vectors are statistically far away from the mean. Further, these algorithms are computationally expensive as well. Here a custom design clustering algorithm (CDCA) has been proposed to overcome these shortcomings. Comparisons have been made using three benchmark case studies, one each from classification, function approximation and prediction domain respectively. The proposed CDCA data partitioning method was evaluated in comparison with SOM, FC and GA based data partitioning methods. It was found that the CDCA data partitioning method not only performed well but also reduced the average CPU time.

ACKNOWLEDGEMENT

I would like to thank my supervisor Prof. Ming J. Zuo for his guidance and mentorship throughout my study and research. I would also like to thank him for providing the financial support to pursue this work. I would like to thank the members of my examining committee, Dr. Amit Kumar and Dr. Yasser Mohammad. I specially thank my wife Krishna not only for her overall support but for the excellent proofreading of the thesis which has improved the language in a significant way. Without the blessing of my parents nothing would be possible. In the end, I would like to thank them for their incessant encouragement and unconditional support.

TABLE OF CONTENTS

Chapter 1	INTRODUCTION.....	1
	1.1 Background.....	1
	1.2 Research Objective.....	4
	1.3 Organization of Thesis.....	6
Chapter 2	FUNDAMENTALS OF ANN.....	7
	2.1 Introduction.....	7
	2.2 Biological Neuron vs. Artificial Neuron.....	8
	2.3 A Multi-Neuron Multi-Layer Network.....	12
	2.4 Training of the Network.....	14
	2.5 Backpropagation Algorithm.....	16
	2.6 Data Partitioning in ANN and its Importance..	20
	2.7 Summary.....	22
Chapter 3	LITERATURE REVIEW FOCUSING ON DATA PARTITIONING METHODS.....	23
	3.1 Introduction.....	23
	3.2 Conventional Data Partitioning Methods.....	26
	3.3 Issues with Conventional Data Partitioning Methods.....	29
	3.4 Non-Conventional Data Partitioning Methods..	30
	3.5 Issues with Non-Conventional Data Partitioning Methods.....	32
	3.6 Summary.....	34

Chapter 4	THE PROPOSED AND REPORTED DATA PARTITIONING METHODS FOR COMPARISON.....	35
4.1	Introduction.....	35
4.2	Problem Statement.....	37
4.3	The Proposed CDCA Data Clustering Algorithm.....	39
4.3.1	Motivation.....	40
4.3.2	Selecting a Suitable Distance Metric and Territory Size d	43
4.3.3	Basic Structure of the Proposed Algorithm.....	45
4.3.4	Optimization of territory size d using Silhouette Coefficient SC	50
4.4	Data Sampling Scheme.....	53
4.5	The General Steps of the Proposed Data Partitioning Method.....	57
4.6	Reported Data Partitioning (DP) Algorithms for Comparison.....	58
4.6.1	Self Organizing Map Based Data Partitioning (SOMDP)	59
4.6.2	Fuzzy Clustering Data Based Data Partitioning (FCDP)	61
4.6.3	Genetic Algorithm Based Data Partitioning (GADP)	62
4.7	Summary.....	65
Chapter 5	EVALUATION OF THE PROPOSED DATA PARTITIONING ALGORITHM.....	67
5.1	Introduction.....	67

5.2	Friedman Regression Function Dataset.....	69
5.3	Housing Dataset.....	72
5.4	Ultrasonic Dataset.....	74
5.4.1	Experimental Setup.....	74
5.4.2	Data Collection Process.....	75
5.5	Performance Measures.....	79
5.5.1	Bias and Variance.....	79
5.5.2	CPU Time.....	80
5.6	Network Selection and Training.....	81
5.7	Results and Discussions.....	83
5.7.1	Friedman Regression Function Dataset Results.....	83
5.7.2	Housing Dataset Results.....	87
5.7.3	Ultrasonic Data Results.....	88
5.8	Summary.....	91
Chapter 6	CONCLUSIONS AND FUTURE WORK.....	93
6.1	Conclusions.....	93
6.2	Future Work.....	94

BIBLIOGRAPHY

LIST OF TABLES

Table 2-1	Biological neuron vs. artificial neuron component analogy.....	10
Table 2-2	List of commonly used activation functions.....	12
Table 4-1	Nomenclature of the proposed data clustering algorithm	39
Table 4-2	Subjective interpretation of the silhouette coefficient <i>SC</i>	52
Table 4-3	70%:20%:10% heuristic rules.....	56
Table 4-4	50%:40%:10% heuristic rules.....	57
Table 5-1	Friedman regression dataset feature trends.....	71
Table 5-2	Housing dataset feature trends.....	73
Table 5-3	Definition and formula for statistical features.....	77
Table 5-4	Ultrasonic dataset feature trend.....	78
Table 5-5	Varying hidden neurons and their test error.....	83
Table 5-6	Friedman dataset results for sampling ratios 70:20:10 and 50:40:10.....	86
Table 5-7	Housing dataset results for sampling ratios 70:20:10 and 50:40:10.....	88
Table 5-8	Ultrasonic dataset results for sampling ratios 70:20:10 and 50:40:10.....	90

LIST OF FIGURES

Figure 2-1	Schematic diagram of a biological neuron.....	9
Figure 2-2	Single artificial neuron with multiple input and single output.....	10
Figure 2-3	A three layer multi-neuron multi-layer ANN.....	13
Figure 2-4	Two layer feedforward ANN.....	16
Figure 2-5	A schematic of the data partitioning process.....	21
Figure 4-1	Schematic diagram of SDP data partitioning.....	38
Figure 4-2	Example dataset to illustrate natural clustering.....	40
Figure 4-3	Example dataset to illustrate customized clustering...	42
Figure 4-4	Flowchart of the proposed CDCA clustering algorithm	48
Figure 4-5	Random sampling of 10 data samples into training: 60%, validation: 20% and test: 20%.....	63
Figure 5-1	Ultrasonic experimental setup.....	74

Chapter 1

INTRODUCTION

1.1 Background

With the world population touching the seven billion mark and more and more developing countries getting economically stronger, the demand for exploiting resources is overwhelming. This has stimulated high demand far and wide for example of the industrial products, mining resources, agro and food processing industries, oil and natural gas sectors and so on. At some point, it has been felt that the sustainable demand is exceeding the production of the goods. Such condition has prompted countries to take steps for further industrialization. Industries are now expected to run their machines nonstop to meet these demands. It is quite understandable that such challenges have increased the competition among peers to increase their production and at the same time decrease the per unit cost of the production.

Keeping these goals in mind, engineers started looking for solutions to minimize the downtime of the machines. Conventional periodic maintenance of the mechanical systems has now become condition based maintenance. However, for any machine that has been assembled with hundreds of individual components, the task to predict the actual machine health conditions is not easy. Over the years it has become a challenging field of research to estimate actual machine health conditions early enough to prevent low output or to avoid catastrophic fail-

ure. It is worth mentioning here that critical component failure has far reaching consequences other than production loss and higher downtime for the machines. The recent leakage in Enbridge pipeline, Michigan, US on July 25, 2010 is a blatant example of such a catastrophic failure. Approximately 1 million gallon of crude oil gushed into the Kalamazoo River. The oil spill caused huge environmental damage up to 30 miles downstream through the towns of Marshall, Ceresco, Battle Creek and Augusta (Picard, 2010).

Traditionally, the tools used for monitoring machine health conditions relied on statistical techniques. But it seems that the existing statistical techniques had their limitations and at many instances it was not possible to implement these tools for obtaining real time solutions. Over the years, artificial intelligence (AI) based tools have been filling up this gap although not completely. For the last four decades, researchers have found profound interest in data driven tools like the artificial neural networks (ANNs). The ease with which ANN can map the input data with logical or numerical output has been the core of this overwhelming interest among the researchers. Some of the background work on ANN started in the late 19th and early 20th century (Hagan et al., 1996). These early foundations were mainly focused on the theory of learning without any mathematical approach. The modern foundation of ANN was laid by Frank Rosenblatt and his team in the late 1950s (Rosenblatt, 1958). They proposed the perceptron network and the associated learning rule and demonstrated its ability to do classification. During this time Bernard Widrow and Ted Hoff introduced the adaptive linear neural networks (Widrow and Hoff, 1960). However, further research on ANN

almost stalled because of the unavailability of the powerful computers of today. With the development of more powerful computers in the 1980s, ANN was re-born (Hagan et al., 1996).

For the last three decades, there has been a tremendous progress in ANN research. Numerous new algorithms have been proposed to speed up the learning or training process. Several variant models have been introduced. Significant amount of progress has been made in finding the optimal architecture as well. All these events have facilitated the application of ANN in diversified fields of research like aerospace, automotive, banking, manufacturing, medicine, oil and gas, finance, speech, transportation, fault diagnosis etc to name a few. Where Han et al. (2009), Dede and Sazli (2010), Scanzio et al. (2010) and Warlaumont et al. (2010) successfully implemented ANN in speech recognition problems, Zhang (2006), Wu et al. (2009), Weerasinghe et al. (1998) and Simani and Fantuzzi (2000) applied ANN in fault diagnosis. Around the same time, Kaastra and Boyd (1996), Bodyanskiy and Popov (2006), Poddig and Rehkugler (1996) and Kim and Chun (1998) are credited with successfully using ANN in financial forecasting. The list of such references on other areas are humongous, thus, it is not possible to include them all here.

Research in ANN can be broadly divided into two categories. The first category is mainly focused on the design and architecture of the ANN models and is dominated by researchers from Computing Science and Mathematics. The second category includes researchers not only from all disciplines of engineering, but

also ranges from medicine to economics and the list are endless. The focus in the latter category has been mainly on the implementation of the existing models. Although there are many exceptions, but in general such division holds true to a great extent. Among one of the many challenges faced by the second community is how to prepare the input data available to them. It is customary to mention that the quality of input data affects the performance of the network. Input data preparation in ANN mainly comprises of three steps (Yu et al., 2007):

- (1) data cleaning which entails removing incomplete data, outliers and random noise;
- (2) input and output feature selection; and
- (3) data partitioning into three sub-groups, namely, training, validation and test datasets.

Although considerable amount of study has been carried out on the first two stages (Park et al., 2010; Chen et al., 2009; Yen and Lin, 2000 and He et al., 2008), the third stage has not received adequate attention. In this thesis work, I am focusing on the third stage.

1.2 Research Objective

Traditionally, the task of data partitioning has been accomplished in an arbitrary fashion. Researchers have found that random data partitioning is not an efficient way and it can adversely affect the performance of the network (Maier and Dandy, 1996; Tokar and Johnson, 1999). In recent years new data partitioning algo-

rithms were proposed to overcome this limitation. Existing methods like genetic algorithm (GA), fuzzy clustering (FC) and self organizing map (SOM) were implemented for this task. It was observed that all these algorithms performed better in comparison to the random data partitioning. So far there is no clear conclusion available from which it can be judged that which one of these new methods performs better in comparison to the other. Each of these algorithms has their own limitations. For example, almost every algorithm being computationally extensive in nature needs huge computational time. From my personal experience, it has been observed that for a decent size input data vectors (say, 100 data vectors), the computational time can be in hours. Yu et al. (2007) found that 50-70 percent of the time and effort is exhausted in data preparation especially in complex data analysis projects. Rest of the time goes into network training, validation and testing. Considering the above factors, my main objectives in this thesis work is as follows:

- (1) Study the existing data partitioning algorithms.
- (2) Identify their limitations.
- (3) Propose a data partitioning algorithm to overcome the above discussed limitations.
- (4) Validate the efficacy of the proposed algorithm with the help of benchmark and experimental data.
- (5) Study the computational complexity of the reported algorithm.

We will use a feedforward back propagation (FFBP) network for our investigation. The FFBP network is a natural choice since it is the most commonly used and the most versatile network in terms of application. It has been implemented in classification, function approximation and prediction problems on several occasions. One benchmark data from each of these problem domains will be considered in this investigation. Reported validation metrics like network bias and variance (May et al., 2010) are taken into consideration in this work. CPU time is considered for the computational complexity analysis.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2 presents the fundamentals of ANN and highlights the importance of data partitioning in ANN. Chapter 3 covers the literature review in ANN with a focus on data partitioning. In Chapter 4 the problem is defined in detail and the limitations of the reported algorithms are discussed. Eventually, the proposed data partitioning algorithm is introduced. We also discuss the reported data partitioning algorithms considered for performance comparison in this study. Chapter 5 includes the experimental design, results and discussions. Two Benchmark datasets from the literature and one experimental dataset from the Reliability Research Lab are considered in this investigation. Reported performance measures like bias and variance from May et al. (2010) is considered to present the results. CPU time is considered for computational complexity analysis of the proposed algorithm. Finally in Chapter 6 the conclusions and future scope for the work is presented.

Chapter 2

FUNDAMENTALS OF ANN

2.1 Introduction

Artificial neural network (ANN) has its inspiration based in the functioning of biological neural network (BNN). When scientists began to understand the functioning of BNN, they discovered that the brain is composed of some 10^{11} biological neurons interconnected to each other. Such a complex network helps us in decision making and cognitive thinking helping us in memorization and learning new things. The next obvious question was whether it is possible to realize such a network assembled with artificial neurons. Some of the background work on ANN started in the late 19th and early 20th century (Hagan et al., 1996). But the mathematical foundation of ANN was laid by Frank Rosenblatt in the late 1950s (Rosenblatt, 1958). Research on ANN almost touched the dead end in the 60s and 70s because of the unavailability of powerful computers. With the advent of more powerful computers in the 1980s, the rebirth of ANN became a possibility. Since then there has been tremendous progress in ANN research. Application of ANN has been reported since in almost every field of research thereafter.

In this chapter, relevant insights on the fundamentals of ANN have been presented. Importance of data partitioning in ANN application has also been highlighted. Sections to follow are organized as indicated: Section 2.2 is started by

making a comparison between biological neuron vs. artificial neuron. Section 2.3 gives a basic idea of a multi-neuron multi-layer ANN architecture. In Section 2.4 the need of training in ANN is explained. Section 2.5 gives an explanation of the backpropagation algorithm. In Section 2.6 the problem of data partitioning and its importance in ANN research is discussed. Eventually the chapter is concluded by presenting a summary in Section 2.7.

2.2 Biological Neuron vs. Artificial Neuron

Although the inspiration for constructing ANN has its genesis in the working of BNN in the human brain, it is not easy to compare the two on the same plane. This is because the functioning of billions of biological neurons in a single network is a very complex process. It has not been possible for the researchers to decode their working so far. However, we can compare the basic constituents of BNN and ANN and examine the similarities between them.

Biological neuron essentially consists of three components: (i) the dendrites (for receiving signals), (ii) the soma (cell body) and (iii) the axons (for sending signals). Figure 2-1 shows the schematic diagram of a biological neuron. The cell body receive the signals though the dendrites. The soma where the nucleus is located processes the incoming signals. The axon is the long fiber that carries the output signals to the other neurons (Hagan et al., 1996).

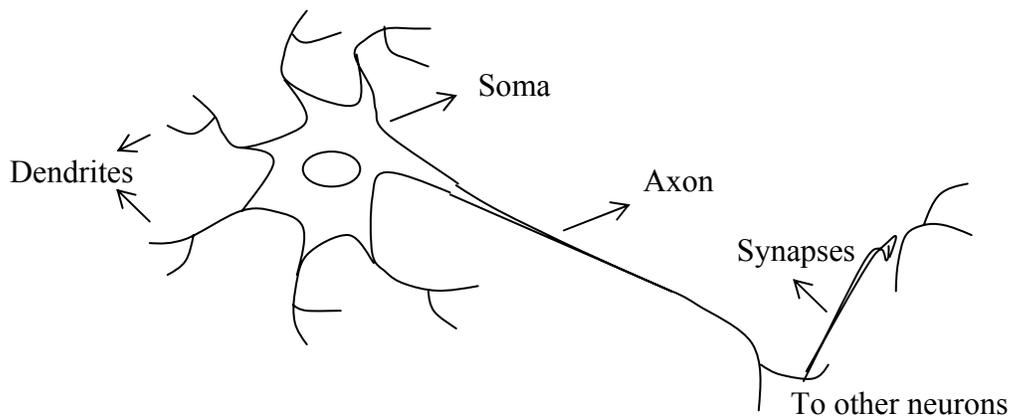


Figure 2-1 Schematic diagram of a biological neuron (Hagan et al., 1996)

The contact between axon of one neuron and the dendrites of another neuron is called synapses. The strength of individual synapses is decided by a complex chemical process. It keeps changing during the learning process throughout human life.

Artificial neuron is considered as a much simpler version than its counterpart biological neuron. However, there is a direct analogy between the individual components of an artificial neuron and a biological neuron. As illustrated in Table 2-1, artificial neuron also consists of three principal constituents which greatly resemble their counterparts in a biological neuron. Figure 2-2 shows an artificial neuron with multiple inputs and a single output. The outputs of the other neurons in the network become the input to this particular neuron. The output of this neuron in turn becomes the input to the other neurons in the network. The inputs are connected to this neuron in question by connections each of which has a certain

weight. Similar to the biological neurons, the artificial neuron goes through learning or training process in which it learns to perform a task. At the end of this training, each of these connection weights is adjusted to a level at which the output of the network matches as closely as possible to the desired output (Dhaka and Singh, 2007).

Table 2-1 Biological neuron vs. artificial neuron component analogy

Biological neuron	Artificial neuron
Soma (cell body)	Processing unit
Axon, Dendrites	Connection
Synaptic strength	Weight

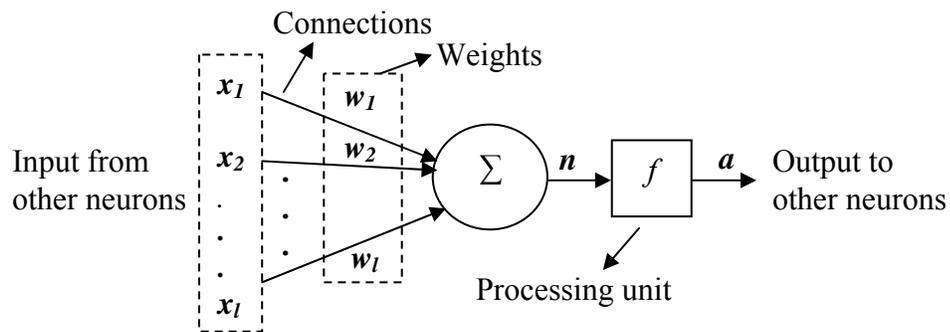


Figure 2-2 Single artificial neuron with multiple input and single output

The functioning of artificial neurons relies on pure mathematics and the principle of optimization. The output of the neuron is a function of the weighted sum of the inputs. The function is called the activation function or transfer function which can be either linear or non-linear. Table 2-2 lists some of the com-

monly used activation functions where a represents the neuron output and n represents the weighted sum of the input. Each of these functions is used under a different condition. For instance, the competitive activation function is used in self organizing feature maps (Kohonen, 2001) which are based on unsupervised training. Let us assume that the input element to be represented as $\mathbf{x}=[x_1, \dots, x_l]^T \in \mathbb{R}^l$. Next suppose the connections are weighted by weight vector $\mathbf{w}=[w_1, \dots, w_l]^T \in \mathbb{R}^l$. Then the input to the processing unit n is a weighted sum of the input element as follows:

$$n = w_1 x_1 + w_2 x_2 + \dots + w_l x_l$$

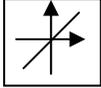
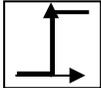
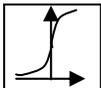
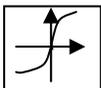
And the output of the neuron y can be expressed as

$$y = f(n) = f(w_1 x_1 + w_2 x_2 + \dots + w_l x_l)$$

where $f(\cdot)$ is the activation function

During the training process, the weight parameters are adjusted to the closest values at which the neuron will give the closest desired output. This is achievable by implementing optimization techniques which will be discussed later in this chapter. The actual output also depends on the type of activation function used. For example, for a non-linear mapping, a non-linear activation function is needed and vice versa. Depending on the complexity of the task, a single neuron may not be sufficient, thus, there is a need for multiple neurons to be working in multiple layers. An example of such a network is given in the next section.

Table 2-2 List of commonly used activation functions

Name of activation function	Icon	MATLAB function	Input/output relation
Linear		purelin	$a = n$
Hard limit		hardlim	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$
Log-sigmoid		logsig	$a = \frac{1}{1 + e^{-n}}$
Hyperbolic tangent sigmoid		tansig	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$
Competitive		compet	$a = 1$ neuron with max n $a = 0$ all other neurons

2.3 A Multi-Neuron Multi-Layer Network

In practice, one neuron may not be sufficient for the purpose desired, thus, we need multiple neurons operating in parallel, called a layer of neurons. Further, there can be multiple layers depending on the complexity of the desired mapping. Figure 2-3 shows a typical multiple neuron three layer network. Here, the first and second layers are called the hidden layers, while the third layer is the output layer. It is also very common practice for some researchers to include the input layer to count the total number of layers (Chong and Žak, 2008). In that case the three layer network would actually become a four layer network. But since there is no real processing of data in the input layer, we will stick to the former convention (Hagan et al., 1996).

Let us assume, the input vector to be represented as $\mathbf{x}=[x_1, \dots, x_l]^T \in \mathbb{R}^l$. In the example network we have S^1 neurons in the first hidden layer, S^2 neurons in the second hidden layer and S^3 neurons in the output layer. As shown in Figure 2-3, each of the elements in input vector $\mathbf{x}=[x_1, \dots, x_l]^T \in \mathbb{R}^l$ has a weighted connection with each of the S^1 neurons in the first hidden layer. The weighted connection between u -th element of the input vector and the v -th neuron of the first hidden layer is represented as $w_{v,u}^1$, where $u=1, \dots, l$ and $v=1, \dots, S^1$. The weighted sums to the S^1 neurons of the first hidden layer are represented by $n_v^1 = (x_1 w_{v,1}^1 + \dots + x_l w_{v,l}^1)$, where $v=1, \dots, S^1$. Similarly the outputs from the first hidden layer represented by a_v^1 , where $v=1, \dots, S^1$ become inputs to the second hidden layer and so on.

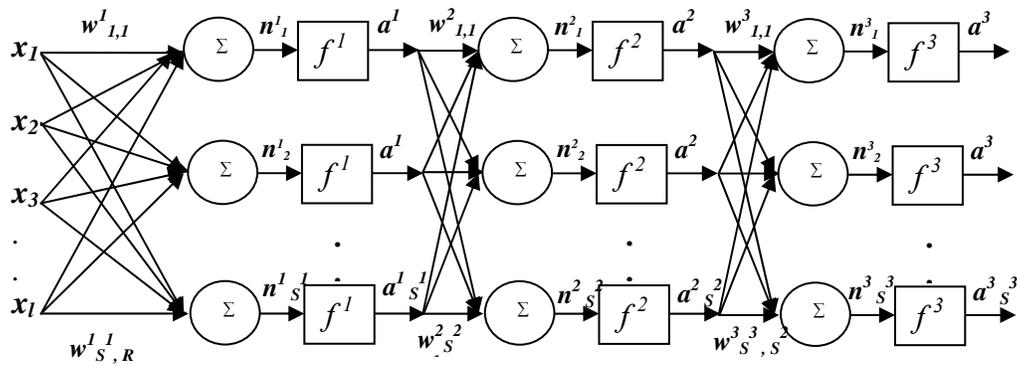


Figure 2-3 A three layer multi-neuron multi-layer ANN (Hagan et al., 1996)

The above network is also known as the feedforward network in literature since the connection is only in the forward direction. However, there is another network which is known as the recurrent network (Levin, 1990) wherein, there is a backward connection. (This recurrent network is out of scope of this thesis and

shall not be discussed here leaving it out for future work. Only the feedforward network has been used in this research). The real challenge in a multi-neuron multi-layer network is adjusting the various weight parameters. The process in which the weights are adjusted to the desired level so that the network can give the desired output is called learning or training. At the training stage the desired output can be the output available from the historical data. The weights are adjusted until the network starts giving output that is very close to the desired output.

Since we now have a preliminary idea of the objective of the training process, let us next discuss the actual mathematics involved in the training process.

2.4 Training of the Network

As mentioned in the previous section, first the problem of training is formulated as an optimization problem. We can then use the available optimization algorithms (line search or random search) for selection of the weights.

Let us consider the single neuron shown in Figure 2-1. Suppose we have T pairs of historical input and output instances such as $\{(x_{d,1}, y_1), \dots, (x_{d,T}, y_T)\}$.

Any input instance $i = 1, \dots, T$ is represented as $\mathbf{x}_{d,i} = [x_{1,i}, \dots, x_{l,i}]^T \in \mathbb{R}^l$, where $d=1, \dots, l$. The corresponding output instance (We call this as actual output) is represented as $y_i \in \mathbb{R}$. Suppose the output of the neuron for input

$\mathbf{x}_{d,i}=[x_{1,i},\dots,x_{l,i}]^{tr} \in \mathbb{R}^l$ at weight vector $\mathbf{w}=[w_1,\dots,w_l]^{tr} \in \mathbb{R}^l$ is $a_i \in \mathbb{R}$ (We call this as network output). It is understandable that for each of these T inputs, there will be T network outputs of the neuron represented as $a_i \in \mathbb{R}, i=1,\dots,T$. We wish to find out the weight vector $\mathbf{w}=[w_1,\dots,w_l]^{tr} \in \mathbb{R}^l$ at which the neuron can map the network outputs $a_i \in \mathbb{R}, i=1,\dots,T$ to the actual outputs $y_i \in \mathbb{R}, i=1,\dots,T$ as closely as possible. The training problem can now be formulated as the sum of squared errors of the neurons as follows:

$$\begin{aligned} & \text{Minimize } \sum_{i=1}^T \left(y_i - f(\mathbf{x}_i^{tr} \mathbf{w}) \right)^2 \\ & \text{Subject to } w_1, \dots, w_l > 0 \end{aligned}$$

where $f(\cdot)$ is the activation function and $f(\mathbf{x}_i^{tr} \mathbf{w})$ is the network output for input instance i

The above optimization problem can be solved by line search algorithms like the gradient search, the Newton's method and the conjugate gradient method (Chong and Žak, 2008). They can also be solved by the random search algorithms like the genetic algorithm (Goldberg, 1989), the simulated annealing (Kirkpatrick et al., 1983) and the particle swarm optimization technique (Kennedy and Eberhart, 1995). In the next section, this optimization problem for a feedforward network having one hidden layer. The gradient search algorithm is used to find out the updating equations. It is also known as the backpropagation algorithm (Hagan et al, 1996; Chong and Žak, 2008) because of the fact that the output error is propagated backward in the network from the output layer to the hidden layer.

2.5 The Backpropagation Algorithm

For better understanding, the backpropagation algorithm is explained with a feed-forward network as an example. It is worth mentioning that similar network has been used in this research with an aim that it would serve as a foundation of our understanding and for future use of such a network. A feedforward network is considered as shown in Figure 2-4.

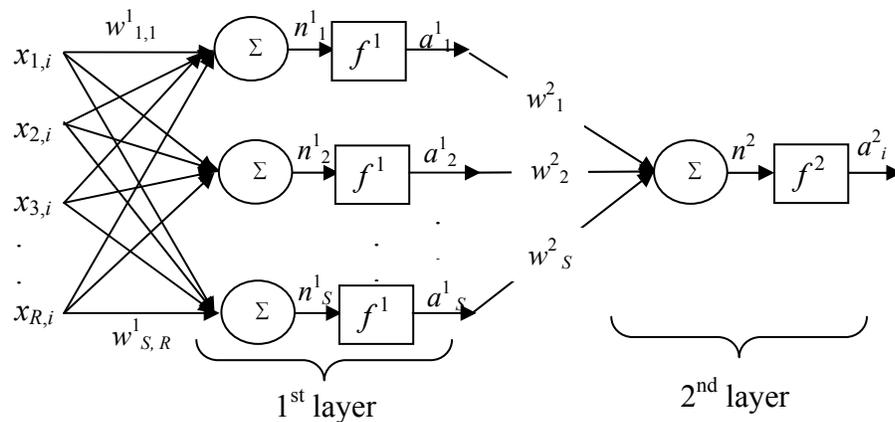


Figure 2-4 Two layer feedforward ANN

The two layers in the above network are referred to as the hidden (1st layer) and the output (2nd layer) layers. Suppose we have S hidden neurons in the hidden layer with activation function $f^1(\cdot)$ in each of the S neurons and a_j^1 is the outputs of hidden layer, where $j = 1, \dots, S$. Suppose we have one output neuron in the output layer with activation function as $f^2(\cdot)$ and a_i^2 is the network output for any input instance i . Suppose we have T pairs of historical input and output instances, $\{(x_{d,1}, y_1), \dots, (x_{d,T}, y_T)\}$. Suppose each of the input instance is repre-

sented as $\mathbf{x}_{d,i} \in \mathbb{R}^R$, where $d = 1, \dots, R$ and each of the actual output instance is represented as $y_i \in \mathbb{R}$, where $i = 1, \dots, T$. We denote the weighted sum of inputs to the hidden neurons as n_j^1 , where $j = 1, \dots, S$ and the weighted sum of inputs to the output neuron as n^2 . We denote the weights connecting inputs to hidden neurons by $w_{j,d}^1$, where $d = 1, \dots, R, j = 1, \dots, S$. We denote the weights connecting the outputs of the hidden layer to the output layer as w_j^2 , where $j = 1, \dots, S$.

The neural network implements a map from \mathbb{R}^R to \mathbb{R} . Then, we have weighted sum inputs to the hidden layer n_j^1 for any input instance i as

$$n_j^1 = \sum_{d=1}^R w_{j,d}^1 x_{d,i}$$

Outputs of hidden neurons a_j^1 as

$$a_j^1 = f^1(n_j^1) = f^1\left(\sum_{d=1}^R w_{j,d}^1 x_{d,i}\right)$$

Similarly, the weighted sum of inputs to output layer n^2 as

$$n^2 = \sum_{j=1}^S w_j^2 a_j^1,$$

Network output a_i^2 for input instance i as

$$a_i^2 = f^2(n^2) = f^2\left(\sum_{j=1}^S w_j^2 a_j^1\right)$$

Therefore, the relationship between the input instance i , $\mathbf{x}_{d,i} \in \mathbb{R}^R$, where $d=1, \dots,$

R and corresponding network output a_i^2 is given by

$$a_i^2 = f^2 \left(\sum_{j=1}^S w_j^2 f^1 \left(\sum_{d=1}^R w_{jd}^1 x_{d,i} \right) \right)$$

As discussed in the previous section, we need to estimate the final weights $w_{j,d}^1$ and w_j^2 by formulating an optimization problem as follows:

The objective of the training process is to adjust the weights of the network such that the network output a_i^2 is as close as possible to the actual output $y_i \in \mathbb{R}$ for all historical input and output instances, $\{(x_{d,1}, y_1), \dots, (x_{d,T}, y_T)\}$.

This can be formulated as the following optimization problem:

$$\begin{aligned} & \text{Minimize } \frac{1}{2} \sum_{i=1}^T (y_i - a_i^2)^2 \\ & \text{where } a_i^2 = f^2 \left(\sum_{j=1}^S w_j^2 f^1 \left(\sum_{d=1}^R w_{jd}^1 x_{d,i} \right) \right) \end{aligned}$$

The factor $\frac{1}{2}$ is added for notational convenience which has no effect on the minimizer. In order to solve this optimization problem using gradient search algorithm, we have to find out the partial derivative of the objective function $F(w_{j,d}^1, w_j^2)$ with respect to $w_{j,d}^1$ and w_j^2 , where $d = 1, \dots, R, j = 1, \dots, S$. We get

$$F(w_{j,d}^1, w_j^2) = \frac{1}{2} \sum_{i=1}^T \left(y_i - f^2 \left(\sum_{j=1}^S w_j^2 f^1 \left(\sum_{d=1}^R w_{jd}^1 x_{d,i} \right) \right) \right)^2$$

The partial derivative of $F(w_{j,d}^1, w_j^2)$ with respect to w_j^2

$$\frac{\partial F(w_{j,d}^1, w_j^2)}{\partial w_j^2} = - \sum_{i=1}^T \left(y_i - f^2 \left(\sum_{p=1}^S w_p^2 f^1 \left(\sum_{d=1}^R w_{pd}^1 x_{d,i} \right) \right) \right) f^{2'} \left(\sum_{p=1}^S w_p^2 f^1 \left(\sum_{d=1}^R w_{pd}^1 x_{d,i} \right) \right) \times f^1 \left(\sum_{d=1}^R w_{jd}^1 x_{d,i} \right)$$

Similarly, the partial derivative of $F(w_{j,d}^1, w_j^2)$ with respect to $w_{j,d}^1$

$$\frac{\partial F(w_{j,d}^1, w_j^2)}{\partial w_{j,d}^1} = - \sum_{i=1}^T \left(y_i - f^2 \left(\sum_{p=1}^S w_p^2 f^1 \left(\sum_{q=1}^R w_{pq}^1 x_{q,i} \right) \right) \right) f^{2'} \left(\sum_{p=1}^S w_p^2 f^1 \left(\sum_{q=1}^R w_{pq}^1 x_{q,i} \right) \right) \times f^1 \left(\sum_{q=1}^R w_{jq}^1 x_{q,i} \right) x_{d,i}$$

The updating equation for the gradient algorithm can now be formulated as follows:

At iteration $n+1$, the weight parameters are updated as

$$w_j^{2(n+1)} = w_j^{2(n)} - \alpha \frac{\partial F(w_{j,d}^{1(n)}, w_j^{2(n)})}{\partial w_j^{2(n)}}$$

$$w_{j,d}^{1(n+1)} = w_{j,d}^{1(n)} - \alpha \frac{\partial F(w_{j,d}^{1(n)}, w_j^{2(n)})}{\partial w_{j,d}^{1(n)}}$$

where α is a fixed step size.

The process of updating is continued until the algorithm converges to minima. Once the minima is reached, the weight parameters achieved at this iteration

are stored and used on test dataset. In the above example, we consider only one hidden layer but in reality there can be more than one hidden layer. For higher number of hidden layers the updating equations can be obtained likewise. The efficiency or speed with which the optimum weight parameters can be found depends on the searching capability of the algorithm. However, the generalization ability of the trained network directly depends on the historical data used in the training process.

During the network training process, the historical data is divided into three datasets such as training, validation and test dataset. The purpose of each of these data sets is discussed in the next section.

2.6 Data Partitioning in ANN and its Importance

The main objective of this section is to define the problem of data partitioning and why it is important to do data partitioning for a satisfactory performance of the ANN model. Figure 2-5 shows a schematic of the data partitioning process. As shown in Figure 2-5, we start with the entire dataset which will include the input and output data. By the end of data partitioning process we come up with three sub-groups of datasets, namely, training, validation and test datasets. Each of these sub-groups have important role to play during the network training and testing process.

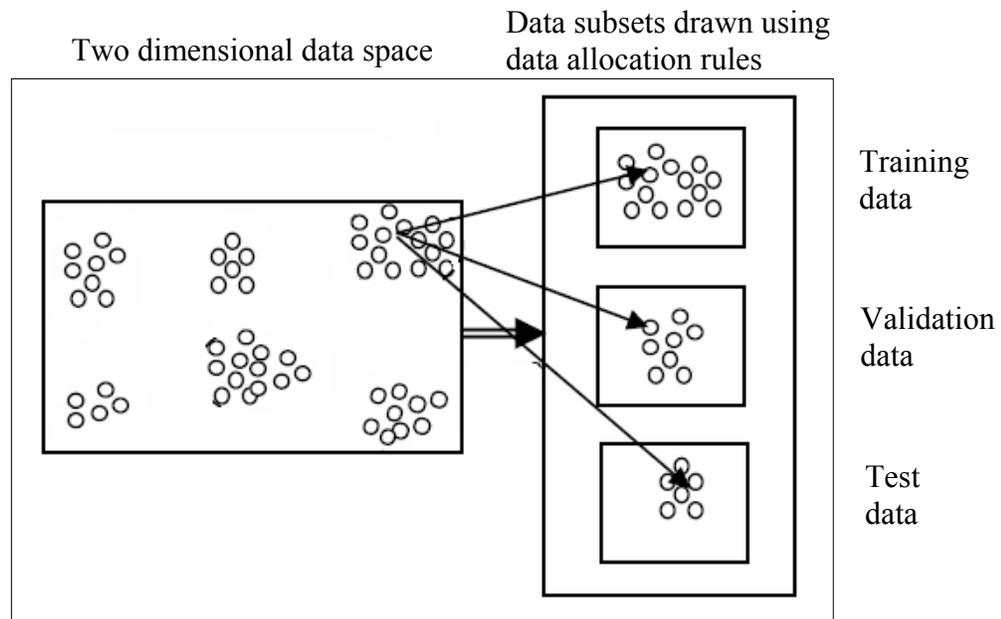


Figure 2-5 A schematic of the data partitioning process

The training data is used to compute and update the network weights parameters at the lowest network sum square error as discussed in the previous section. The validation data is used to cross-validate this error during the training process. Typically the network sum square error for both training and validation data decreases during the initial iterations. But after a certain number of iterations, the training error and the validation error begins move apart. This is the point when the network has started over-fitting the training data or the validation data depending on numerical value of the sum square error. For example if the training error is lower than the validation error, then the network has over-fitted the training data and vice versa, thus, the training process must be stopped. The network weights at this stage are finalized for future use on test data. It is always a better

practice to cross-validate the network with the validation data in order to avoid over-fitting.

The test data is used separately after the training is over to estimate the performance of trained network. The test data is needed to test the true generalization ability of the network. In some literatures, validation and test data have been used interchangeably although the objective of use remains the same. Some researchers have only used the training and test datasets.

2.7 Summary

In this chapter, the fundamentals of ANN were presented. The chapter was started with a bit of history. A comparison between the biological neuron and artificial neuron was made next. The mathematics involved in artificial neurons was illustrated. The general structure of a multi-neuron multi-layer artificial network was talked about. The purpose of training in ANN was discussed. Next, the back-propagation algorithm was discussed by deriving the updating equations for a single hidden layer network. It was discussed that the same approach can be implemented to higher number of hidden layers. Eventually the problem of data partitioning in ANN and its importance for a satisfactory performance of the network was defined. The role of training, validation and test data and their importance was mentioned.

The next chapter will cover the literature review of ANN with a special focus on data partitioning.

Chapter 3

LITERATURE REVIEW FOCUSING ON DATA PARTITIONING METHODS

3.1 Introduction

Over the last four decades there has been a plethora of publications on ANN. It is indeed a difficult task to include all of them in this literature review. However, here a general approach is followed by which most of the areas can be addressed with special focus on data partitioning. In general, literature on ANN can be described in two ways depending on the researcher's area of interest. The first category includes contributions of the researchers on the development of ANN models such as finding novel approach to optimize the number of hidden neurons in a network, finding the best weight parameters of the network and finding the best connection links in a network. While the second category includes contributions of the researchers on the application of ANN models for example to speech recognition, fault diagnosis, financial forecasting etc.

Talking about the first category, Teoh et al. (2006) estimated the necessary number of neurons in the hidden layer in a single hidden layer feedforward network using the pruning/growing technique based on the singular value decomposition. Trenn (2008) derived explicit formulas for the necessary number of hidden units and its distributions to the hidden layers of the multi-layer perceptron (MLP)

network. Gil et al. (2009) estimated the number of hidden neurons in recurrent neural networks for nonlinear system identification using singular value decomposition. Ludermir et al. (2006) proposed an optimization methodology to estimate the weights and architectures simultaneously of an MLP network. They combined simulated annealing (SA) (Kirkpatrick et al., 1983) and tabu search (TS) (Glover, 1986) for this work. Wan and Banta (2006) proposed a novel stochastic (or online) training algorithm for neural networks, named parameter incremental learning (PIL) algorithm. They claimed that the convergence speed and accuracy of PIL algorithm for MLP is measurably superior to the standard online backpropagation (BP) algorithm and the stochastic diagonal Levenberg–Marquardt (SDLM) algorithm. Peng and Li (2008) introduced a new Jacobian matrix to speed up the convergence of second-order learning algorithms such as Levenberg–Marquardt (LM), as well as to improve the network performance. This is achieved here by reducing the dimension of the solution space.

Schuster and Paliwal (1997) proposed a bidirectional recurrent neural network (BRNN) which is an extension of the regular recurrent neural network (RNN). The usual limitation of using input information just up to a preset future frame in RNN is overcome by training in positive and negative time direction simultaneously. Fallahnezhad et al. (2011) proposed a novel hybrid higher order neural classifier (HHONC) which contains different high-order units, in contrast with conventional fully-connected higher order neural networks (HONN). The proposed method uses fewer learning parameters and allocates the best fitted model in dealing with different datasets by modifying the orders of different high-

order units and updating the learning parameters. Abiyev et al. (2011) proposed a novel type-2 neuro-fuzzy system for identification of time-varying systems and equalization of time-varying channels using clustering and gradient algorithms. It combines the advantages of type-2 fuzzy systems and neural networks.

Let us next talk about the second category of research dealing with the application of ANN models. The second category includes contributions on the application of ANN models. In Chapter 1, Section 1.1 some of these applications were mentioned which included applications to the field of medicine, speech recognition, fault diagnosis, financial forecasting and so on. For these applications, the main contribution of the researchers has been the way input data preparation is carried out in different problem domains. This task can be often challenging while dealing with field data. Some of the most common anomalies include unwanted random noise, outliers and missing data.

Input data preparation in ANN mainly comprises three steps (Yu et al., 2007): (1) data cleaning which entails removing incomplete data, outliers and random noise; (2) input and output feature selection; and (3) data partitioning into three sub-groups, namely, training, validation and test datasets. One of the most prominent challenges faced by researchers is to divide the input data in a way such that the best generalization of the network is achieved. This is done by dividing the input data into three subsets: (a) training data, (b) validation data and (c) test data. Researchers have performed data partitioning in mainly two ways,

namely, conventional data partitioning method and non-conventional data partitioning method.

In Section 3.2 and 3.3 of this chapter the conventional data partitioning methods and their issues will be discussed. In Section 3.4 and 3.5, the non-conventional data partitioning methods and their issues are discussed. In Section 3.6, a brief summary is provided while the point key issues are addressed next.

3.2 Conventional Data Partitioning Methods

The conventional data partitioning method entails dividing the entire dataset randomly into either (a) training and test dataset or (b) training, validation and test dataset. Researchers have already proven serious issues with the conventional methods. We shall discuss them in the next section. In this section we will discuss some of the literatures which have applied conventional data partitioning.

Namia and Deyhimi (2011) applied ANN models to predict activity coefficients at infinite dilution for organic solutes in ionic liquids (ILs). They randomly divided the collected data into two groups, namely, 70% for training and 30% for test data out of 916 activity coefficients. The training set is used to evaluate the performance of the network and no cross-validation is performed.

Keeratipibul et al. (2011) used ANN model to predict the relationship among the initial bacterial load, type of vegetable/fruit, types and concentration of sanitizer and residual microorganism levels after the sanitizing. They also randomly divided the collected data into two groups, namely, 70% for training and

30% for test data. It was observed that the performance of neural models was clearly superior for the training data set and was reasonably good for the test data set. Although they invested time to optimize the network by varying the number of hidden neurons, they did not cross-validate the network using a validation dataset. This is why they observed a significant drop in network performance from training to test dataset.

Ray and Klindworth (2000) implemented ANN models to assess the pesticide and nitrate contamination of rural private wells. Although they recognized the importance of data partitioning and made their training dataset large enough to represent the full population, they did not cross-validate their ANN models. They used four separate ANN models and also checked the sensitivity of their models by varying the number of hidden neurons. The training efficiency of the network was estimated at the best prediction accuracy level of 95% and 100% for the four models. It is quite possible that their networks had been over-fitted to the training data and hence, lacked generalization. This is one possible reason why they achieved prediction accuracy ranging from as low as 50% to 90% for the test data.

Sahoo et al. (2006) applied feedforward ANN to assess pesticide contamination in shallow groundwater. Although they were aware of the importance of cross-validation in ANN, they did not divide their data into three subsets. Instead they randomly divided their datasets into two datasets: (i) 65% for training and (ii) the remaining 35% for testing. They mentioned that because of unavailability

of enough data samples, it was not possible to make three representative data sets of the same population.

He et al. (2011) implemented ANN model to estimate the monthly total nitrogen concentration in streams. They used 80% of the water quality data from 40 rivers as a training dataset and 20% of the records as the validation dataset (they named it test set). The data for the test dataset (they named it validation set) were collected from the other 19 rivers, which the network had never used. Although they mentioned the importance of dividing the dataset in such a way that both training and test datasets are statistically comparable (Palani et al., 2008), it is not clear how they divided it.

Sinha et al. (2007) applied backpropagation artificial neural network (ANN) in differentiating electroencephalogram (EEG) power spectra of syncopic and normal subjects. A classification accuracy of 85.75% for syncopic and 92% for normal subject was reported on the test dataset. Although the available dataset size is quite big (960 dataset), they have not utilized validation dataset for cross-validation. 160 for training and 800 for testing of power spectrum values were used to evaluate the performance of ANN. Final network parameters were decided on training data alone for which they registered 100% accuracy. It is quite possible that their ANN model had extrapolated the classification results over a long range and lacked generalization abilities. In the next section inferences about the reported results is made and the issues with conventional data partitioning is discussed.

3.3 Issues with Conventional Data Partitioning Methods

Cross-validation of the ANN model using the validation dataset is considered to be the most effective method to prevent over-fitting (Smith, 1993). The concept of over-fitting can be explained as follows: When test data is given as input to the trained network, the network error seems to be higher than the error obtained during training. This is because of the fact that the network has learned to memorize the training data instead of generalizing it to new data samples. Thus it is essential that the whole dataset is divided into three sub-groups, namely, training, validation and test data instead of two sub-groups, namely, training and test data. Thus validation dataset is mandatory for cross-validation purpose. In the previous section, Keeratipibul et al. (2011), Ray and Klindworth (2000), and Sinha et al. (2007) have not performed cross-validation while training their networks because of which their networks have over-fitted to the training dataset. This is the reason why they obtained lower accuracy for the test data as compared to the training data.

Satisfactory performance of the network cannot be guaranteed just by dividing the whole dataset into three sub-groups. It is understood that in order to achieve a satisfactory performance, all three datasets must be true representative of the entire dataset. Statistical properties of each of these sub-groups must be taken into account to ensure this representativeness. Conventional data partitioning often compels the network to extrapolate results which may not be reliable enough (Minns and Hall, 1996; Tokar and Johnson, 1999). In general it is ex-

pected that the test data must belong in the range of training data to avoid extrapolation. This can only be ensured when the test data is statistically equivalent to the training dataset. In recent years researchers have tried to overcome this problem by proposing non-conventional data partitioning methods using genetic algorithm (GA), self organizing map (SOM) and fuzzy clustering (FC) etc. In the next section some of the non-conventional data partitioning methods from the literature are discussed.

3.4 Non-Conventional Data Partitioning Methods

Bowden et al. (2002) had to deal with a case study to forecast salinity in the River Murray at Murray Bridge (South Australia) 14 days in advance. They challenged the conventional random data partitioning by introducing GA and a SOM for data partitioning. They checked the performance of their ANN model on a test dataset from July 1992 to March 1998. A reduction in error by 24.2% is reported for GA based data partitioning over the conventional data partitioning method. For SOM based data partitioning a reduction in network error by 9.9% is reported over the conventional data partitioning.

Shahin et al. (2002) applied ANN models to predict the settlement of shallow foundations on granular soils in geotechnical engineering. Previously they implemented ANN models to the same problem using random data partitioning (Shahin et al., 2001 and Shahin et al., 2002). In the third attempt, they implemented data partitioning methods like (1) data partitioning to ensure statistical consistency of the subsets; (3) data partitioning using self organizing map (SOM);

and (2) data partitioning using fuzzy clustering. When compared to random data partitioning method, the drops in mean absolute error of the network on test dataset for the three methods was found to be approximately 1%, 20% and 30% respectively.

Samanta et al. (2004) applied genetic algorithm based data partitioning for their problem in estimating ore reserve based on sparse drill hole data for a placer gold property in Nome, Alaska. Statistical properties like mean and standard deviation were used to measure the degree of statistical agreement among the three sub-groups. Independent comparisons were made among three sub-groups for each of the input features. The closer the statistical agreement among the three sub-groups for input features, the better the performance of data partitioning. When genetic algorithm based data partitioning was compared with the random data partitioning, a relatively closer agreement in the former was observed. In another paper by Samanta et al. (2004), a comparison was made between random data partitioning and SOM. A closer statistical agreement among training, validation and test datasets was found for most of the input feature vectors.

May et al. (2010) compared random data partitioning with non-conventional methods like self organizing map (SOM), genetic algorithm (GA) and DUPLEX method (Snee, 1977). The problem under investigation was the Friedman regression function with different distributions of the random variable and noise. They considered six different datasets based on such varying distributions. Their results in terms of network bias and variance showed better results for

self-organizing map as compared to the genetic algorithm and DUPLEX method. Overall, all the participating algorithms outperformed random data partitioning. Reeves and Taylor (1998) applied genetic algorithm to find an optimal subset of the training data, which was then used for training the network. Bowden et al. (2006) attempted to forecast chlorine residuals in a water distribution system using general regression neural network. They implemented GA based data partitioning method to divide the available data into training, testing and validation dataset. The three sub-groups were made statistically representative subsets of the same population.

Other unconventional methods to ensure the statistical representativeness are as follows. Kennard-Stone sampling or CADEX is one of the earliest algorithms designed for data partitioning (Kennard and Stone, 1969). In this approach data samples are drawn iteratively based on selecting points farthest away from those already included in the sample, and ensuring maximum coverage of the data. Another improvised version of CADEX is the DUPLEX method (Snee, 1977). It is widely used in several ANN applications (Sprevak et al., 2004; Despaigne and Massart, 1998). The computational complexity of this algorithm sometimes prohibits its use on larger datasets. Bowden et al. (2005) used the Kolmogorov-Smirnov statistic to match the distribution of each variable across the sampled datasets.

3.5 Issues with Non-Conventional Data Partitioning Methods

Optimization based data partitioning like genetic algorithm (GA) may not be suitable for large datasets since the total number of combination of data split can be a gigantic task to explore. For an input dataset only having only 60 data points to be divided into 40 training, 10 validation and 10 test datasets, there will be ways of arranging the data points.

$$\frac{60!}{40!10!10!} = 7.7 \times 10^{20}$$

Although self organizing map (SOM) and fuzzy clustering (FC) based data partitioning have better performance, they are computationally expensive too. This is because SOM and FC are based on the principle of iterative learning and optimization. In their review of the literature, Yu et al. (2007) observed that 50-70 percent of the time and effort was spent in data preparation in complex data analysis projects. In the reported non-conventional data partitioning methods, the computational complexity was never taken into consideration. More descriptions on these algorithms are given in Chapter 5. Standard classification or clustering algorithms like SOM (Kohonen, 2001), fuzzy clustering (Kaufman and Rousseeuw, 1990; Bezdec, 1981), K-mean (MacQueen, 1967), and vector quantization (Linde et al., 1980) are designed to form natural clusters (See Chapter 4) at their best performance. Some of these natural clusters can be big in the case of large multivariate datasets, such that the extreme data points are statistically far away from the mean. This may affect the most fundamental assumptions of data partitioning that data points are in close statistical agreement inside each sub-groups.

3.6 Summary

It is indeed a difficult job to include all the areas of research in ANN in one chapter considering the humongous amount of publications existing in the literature. However, in this chapter, an overall approach or compromise has been tried to be achieved by considering some of the work from these areas. The approach adopted will help lay the foundation in the next chapter to the problem of data partitioning and its importance in ANN research. Both conventional and non-conventional data partitioning approaches and their drawbacks have been discussed from the literature. Some of the areas related to weight and architecture optimization, novel learning algorithms to improve convergence and speed has also been touched upon in this chapter.

Chapter 4

THE PROPOSED AND REPORTED DATA PARTITIONING METHODS FOR COMPARISON

4.1 Introduction

In Chapter 3 several existing issues with reported data partitioning methods were mentioned. Through this research, the objective is to address some of these issues. Considering the fact that it is not possible to address everything, two specific goals are focused particularly in this work: (1) to minimize the computational complexity in terms of CPU time; (2) to achieve better model performance in terms of prediction accuracy. Both of these objectives may be achieved by either making changes to the existing algorithms or developing a new algorithm. In this work the second approach is adopted. In this chapter the proposed custom design clustering algorithm (CDCA) is discussed in detail.

There are mainly three approaches followed by researchers to counter the problems in random data partitioning: (1) stratified data partitioning (SDP) based on self organizing map (SOM) (May et al., 2010 and Samanta et al., 2004) and fuzzy clustering (FC) (Bowden et al., 2002); (2) optimized data partitioning (ODP) based on genetic algorithm (GA) (Shahin et al., 2004); (3) data partitioning to ensure statistical consistency of the subsets (Shahin et al., 2004 and Bowden et

al., 2005). From the literature review, it is observed that the SDP class algorithms have given better network performance when compared to the other two categories (Shahin et al., 2002 and May et al., 2010). On the other hand ODP method based on GA has performed better on one occasion over the SDP method based on SOM (Bowden et al., 2002). ODP methods are not suitable for large datasets since the search space can be humongous. As mentioned in Chapter 3, for a dataset of 60 data samples, the search space in ODP class can be in 10^{20} .

Being data driven method, ANN models are often required to handle large datasets. In this work, a large dataset is defined as a dataset having more than 800 data samples. In reality it is not very rare to encounter datasets having data points in the multiple of thousand. In other words, the first objective of minimizing computational complexity cannot be achieved using the ODP method. On the other hand, the SDP class uses a strategy in which the whole dataset is stratified into smaller clusters and then data points are extracted from each of these clusters. In the next section a detailed discussion is given on the SDP approach since the proposed method belongs to the SDP category.

In Section 4.2 the challenges and issues with the stratified data partitioning are discussed. Section 4.3 presents the motivation and basic structure of the proposed clustering algorithm. The data sampling scheme used for data partitioning is discussed in Section 4.4. Section 4.5 includes the general steps of the proposed data partitioning method. In Section 4.6, a performance comparison has been pre-

mented with respect to the reported algorithms. Eventually, a summary of this chapter is given in Section 4.7.

4.2 Problem Statement

The SDP methods share similar principles with the “divide and conquer” algorithm (Rugina and Rinard, 2001). A “divide and conquer” algorithm works by breaking down a problem into two or more sub-problems of the same (or related) type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

Similarly instead of extracting data into training, validation and test data from the entire dataset (all inputs and outputs data points), it is first divided into smaller clusters in Step 1, as shown in Figure 4-1. This strategy helps to maintain better statistical agreements among training, validation and test datasets. In Step 2 data points are sampled into three sub-groups from each of these clusters.

Standard clustering algorithms like SOMs and fuzzy clustering are employed for this job. Allocation rules like equal allocation (Bowden et al., 2002), proportional allocation (May et al., 2010) and Neyman allocation (Cochran, 1977; May et al., 2010) are implemented in Step 2. These allocation rules differ from each other by the way the ratio of data samples (to be collected from each of these clusters) is calculated. But once this ratio is finalized, data samples are extracted in a random fashion as in random data partitioning.

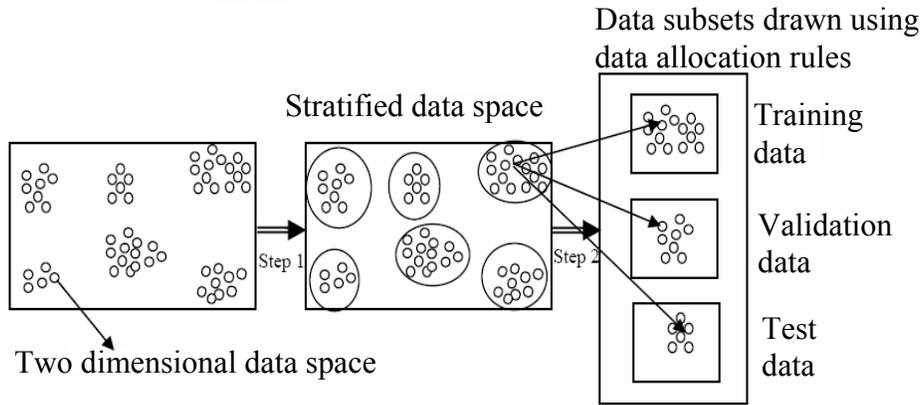


Figure 4-1 Schematic diagram of SDP data partitioning

Although researchers believed that data points in any individual cluster are in close statistical agreement, it is not always true since some clusters can have very large spatial territory. Standard classification or clustering algorithms like SOM (Kohonen, 2001), fuzzy clustering (Kaufman and Rousseeuw, 1990; Bezdec, 1981), *k*-mean (MacQueen, 1967), and vector quantization (Linde et al., 1980) are designed to form natural clusters. The stratified data space in Figure 4-1 illustrates an example of natural clustering or clusters. Some of these natural clusters can be very large, such that the extreme data points are far away from the geometric center or statistically far away from the mean from statistical point of view. This will invalidate the assumptions made in Step 1 that data points are in close statistical agreement inside the stratum. Now it is easily understood that the goal of the clustering algorithm in SDP approach is not finding natural clusters but smaller clusters irrespective of its natural orientation in the data space which will ensure the statistical agreement. It is not possible to obtain such customized clusters by using standard clustering algorithm. Thus a novel clustering algorithm

is proposed in this work. The proposed algorithm is specifically designed to overcome the limitations of reported algorithms like SOM and FC. In the subsequent section the proposed CDCA clustering method is discussed.

4.3 The Proposed CDCA Data Clustering Algorithm

The goal of this custom designed clustering algorithm (CDCA) are as follows: (1) to obtain customized clusters having close statistical agreement; (2) to work faster in terms of CPU time as compared to its competitors like SOM and FC; (3) to successfully integrate CDCA in the overall data partitioning process; and (4) to have better performance relative to the reported DP methods. Table 4-1 illustrates the nomenclature of symbols used in the proposed data clustering algorithm.

Table 4-1 Nomenclature of the proposed data clustering algorithm

Notation	Definition
d	Territory size or the Euclidian distance between the representative object n_k and its boundary where k is the k -th cluster.
\mathfrak{R}^l	l -dimensional Euclidian space
n_k	Representative object of cluster k , where $k=1,2,\dots,m$ and m is the number of clusters.
\mathbf{x}^i	Data point i , where $i = 1,2,\dots,r$, where r is total number of data points in the entire dataset
$\mathbf{w}_k^{(q)}$	q -th weight vector of representative object n_k , where $0 \leq q \leq p$ and p is the total number of data points in cluster k
t	Updating coefficient
\tilde{s}	Average silhouette width
SC	Silhouette coefficient or maximum average silhouette width
$s(\mathbf{x}^i)$	Silhouette value of data point \mathbf{x}^i
$a(\mathbf{x}^i)$	Average Euclidian distance of data point \mathbf{x}^i to all other data points of the same cluster
$b(\mathbf{x}^i)$	Lowest average inter-cluster Euclidian distance of data point \mathbf{x}^i
$d(\mathbf{x}^i)$	average inter-cluster Euclidian distance of data point \mathbf{x}^i from any cluster k

**Notations in bold are vectors.*

4.3.1 Motivation

The proposed CDCA algorithm is motivated to address the shortcomings of the reported SDP algorithms. The reported algorithms like self organizing map (SOM) and fuzzy clustering (FC) are designed to form natural clusters. In case of large multivariate datasets which is often the case in most data driven research, these natural clusters can be large enough such that two extreme data points may be statistically far from each other. This violates the most fundamental notion of stratified data partitioning which relies on the assumption that data points in each cluster are in close statistical agreement. The above discussion is further illustrated with the help of a two dimensional dataset in Figure 4-2.

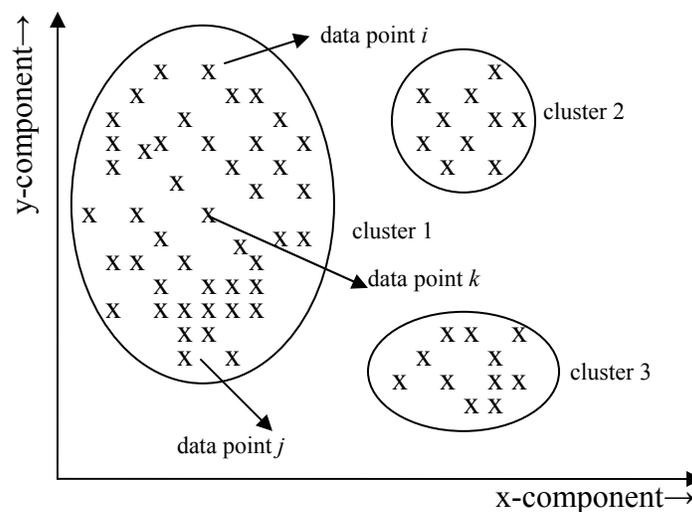


Figure 4-2 Example dataset to illustrate natural clustering

The example dataset in Figure 4-2 has three natural clusters which can be

obtained by using the reported clustering algorithms. In cluster 1, data point i , data point j and data point k are far away from each other, thus, are not in a close statistical agreement. When data points are sampled, it is possible that data point i is sampled into training dataset, data point j is sampled into validation dataset and data point k is sampled into test dataset. But based on the desired statistical agreement, data point j and data point k are not the true representative of data point i . Such extreme data points will have an adverse effect on the statistical properties (mean μ and standard deviation σ) of each subset (training, validation and test). Cluster 2 and cluster 3 are smaller clusters, hence, are statistically in a better agreement than cluster 1. Thus, it is easily understood that smaller clusters have a better statistical agreement between subsets.

Standard clustering algorithms like SOM and FC are designed to form clusters of different cluster or territory sizes. More about fuzzy clustering and self organizing map is discussed in Section 4.6. If the standard clustering algorithms are used and a large number of clusters are specified, though smaller clusters are achieved, we still end up with clusters of different territory sizes. This will have an adverse effect on the statistical properties, like, the standard deviation of clusters. Thus, in this research the aim is to develop a strategy to obtain clusters of equal territory size. In Figure 4-3 the ideal desired clusters for the above mentioned dataset are redrawn. These clusters being equal in territory size are expected more likely to maintain a close statistical agreement.

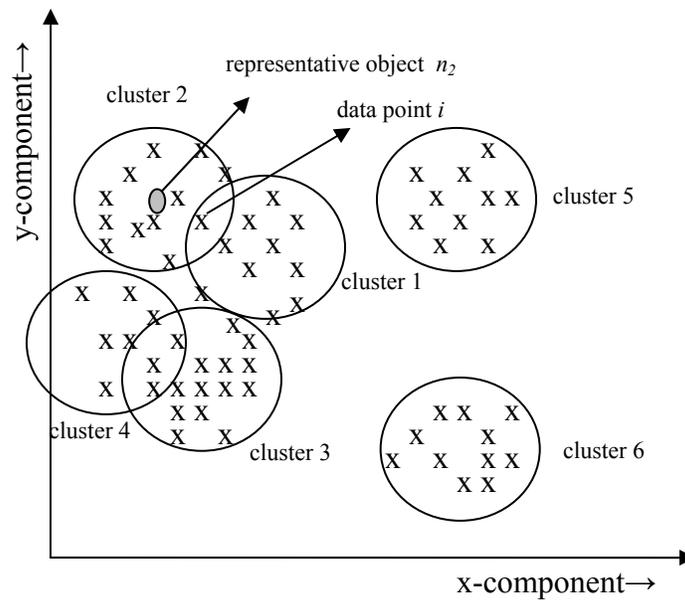


Figure 4-3 Example dataset to illustrate customized clustering

In the literature, the vast majority of clustering algorithms are mainly of two types, namely, partitioning and hierarchical (Duran and Odell, 1974). Partitioning method classifies the data into k clusters only. On the other hand hierarchical algorithms deal with all values of k in the same run. The proposed CDCA method is motivated from the partitioning class of clustering algorithm. Any clustering algorithm essentially satisfies the following two requirements:

- Each object must belong to exactly one cluster which means all clusters together add up to the entire dataset.
- Each cluster must contain at least one data point or object which means there are at most as many clusters as there are data points.

To satisfy the above requirement, data points in the overlapped zone in Figure 4-3 (for example, data point i in cluster 2) can be assigned to either cluster

1 or cluster 2. This decision is made based on the lowest Euclidian distance (See Section 4.3.2 for definition) between data point i and the representative objects n_1 and n_2 of cluster 1 and 2 of that cluster. Representative object n_k , where $k=1, \dots, m$ and m represents the number of clusters, is the geometric center of the cluster and can be represented by a hypothetical data point w_k . In reported clustering algorithms (FC, SOM) the number of clusters m , is given by the user. Thus, the user has no direct control over the territory size. However, in the proposed clustering algorithm, the goal is to obtain equal territory size for each cluster. The geometrical shape of the territory is a circle for a two dimensional data in contrast to a sphere for a three dimensional data. In this research, work has been carried out on datasets having more than three dimensions or inputs. Thus, two important factors in the proposed algorithm are:

- A suitable distance metric to work in higher dimensional space
- A territory size d for which data points are sampled while maintaining the closest statistical agreement among the three sub-groups.

4.3.2 Selecting a Suitable Distance Metric and Territory Size d

A proper distance metric is required to measure the spatial dissimilarities among data points in higher dimensional space. From literature, we select the most commonly used distance metric, “the Euclidian distance,” since we are interested in the actual spatial distance. Euclidian distance between data point

$\mathbf{x}^i = [x_1^i, x_2^i, \dots, x_n^i]^T$, and $\mathbf{x}^j = [x_1^j, x_2^j, \dots, x_n^j]^T$ is defined as

$$d(i, j) = \sqrt{(x_1^i - x_1^j)^2 + (x_2^i - x_2^j)^2 + \dots + (x_n^i - x_n^j)^2}$$

Territory size d is defined as the Euclidian distance between the representative object n_k , where $k = 1, \dots, m$ and m represents the total number of clusters and its boundary. In the proposed algorithm territory size d is given by the user, thus, the boundary is indirectly set by the user as well. Representative object n_k is represented in the n -dimensional Euclidian space by weight vector $\mathbf{w}_k^{(q)}$, where k is cluster k ; $0 \leq q \leq p$ and p which is the total number of data points in cluster k represents the updation number. During the implementation process of the algorithm, $\mathbf{w}_k^{(q)}$ gets updated every time a new data point is introduced to the cluster. In the next section, we will discuss the updating process.

The territory size d of each cluster is equal which means the n -tuple Euclidian space is shared equally by all clusters. The number of objects or data points in each cluster may vary depending on their corresponding spatial locations. This property helps in avoiding the formation of natural clusters and gives total control to the user to determine the size of the cluster. Since every cluster has identical territory size, the likelihood of agreement in standard deviation σ among clusters is more compared to the other reported clustering algorithms. This facilitates the use of sampling rules like equal allocation and proportional allocation which were not naturally suitable in the reported stratified data partitioning methods (May et al., 2010). The choice of territory size d directly influences the number of clusters. As the former grows bigger, the latter becomes smaller and vice versa. The

minimum territory size d_{min} is obtained when the number of clusters k , is equal to the total number of input data points r . Similarly, the maximum territory size d_{max} is obtained when the number of clusters k , is equal to one.

Previously, Shahin et al. (2004) and May et al. (2010) implemented silhouette coefficient SC (Rousseeuw, 1987) to find the optimal number of cluster k in FC and SOM. Silhouette coefficient SC measures the degree of membership of individual data points to the cluster they belong. In the proposed algorithm, silhouette coefficient SC is used to find the optimal territory size d . Silhouette coefficient SC shall be discussed in Section 4.3.4. Before discussing the silhouette coefficient, the basic structure of the proposed algorithm for a particular territory size d is formed and discussed.

4.3.3 Basic Structure of the Proposed Algorithm

For a specified or given territory size d , suppose we have an input data matrix of dimension l by r , where l represents the dimension of each data point and r represents the number of data points. Data points are represented as $\mathbf{x}^i \in (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^r)$

. Any data point i which includes both input and output vectors is represented

$\mathbf{x}^i = [x_1^i, x_2^i, \dots, x_l^i]^T \in \mathfrak{R}^l$. In the proposed algorithm, each cluster is to be represented

by a representative object n_k . The representative object n_k which represents

cluster k , has a magnitude or weight of $\mathbf{w}_k^{(q)}$. It is obvious that the weight vector is

l -dimensional and thus, is represented as $\mathbf{w}_k^{(q)} = [w_{k,1}^q, w_{k,1}^q, \dots, w_{k,l}^q]^T \in \mathfrak{R}^l$, where k

represents the cluster number and q represents the updating number. The following steps are adopted for the proposed clustering algorithm.

In the proposed algorithm data points are randomly introduced into the Euclidian space. Let us assume that the 1st data point \mathbf{x}^i is randomly selected and introduced into the Euclidian space. The next step is to introduce the 1st representative object n_1 with a weight vector $\mathbf{w}_1^{(0)}$. Whenever a new representative object n_k is introduced to the Euclidian space, its weight vector $\mathbf{w}_k^{(0)}$ is assumed to be $[0, 0, \dots, 0]^T \in \mathfrak{R}^l$. The weight vector $\mathbf{w}_1^{(0)}$ is updated using the equation $\mathbf{w}_1^{(1)} = \mathbf{w}_1^{(0)} + t(\mathbf{x}^i - \mathbf{w}_1^{(0)})$, where t is the updating coefficient which can vary from $0 \leq t \leq 1$.

A higher value of t close to 1 (we have used 0.9) is preferred when the representative object n_k is introduced for the first time. This will bring the representative object n_1 in close proximity of data point \mathbf{x}^i . At this stage, representative object n_1 has one data point in its territory. The input data matrix contains all data points except data point \mathbf{x}^i and is represented as l by $(r-1)$. Now, we randomly select the second data point \mathbf{x}^j from the data matrix l by $(r-1)$. The following decision rule is implemented to cluster the second data point.

- If the second data point \mathbf{x}^j falls inside the boundary of the representative object n_1 , i.e., Euclidian norm, $\|\mathbf{w}_1^{(1)} - \mathbf{x}^j\| \leq d$, it is assigned to representative object n_1 . At this stage, representative object n_1 has data points \mathbf{x}^i and \mathbf{x}^j in its territory. Since data point \mathbf{x}^j is already inside the territory of repre-

representative object n_1 , a low value of updating coefficient t is preferred. Thus, we update the weight vector $\mathbf{w}_1^{(1)}$ of representative object n_1 using the equation $\mathbf{w}_1^{(2)} = \mathbf{w}_1^{(1)} + t(\mathbf{x}^j - \mathbf{w}_1^{(1)})$ with a lower t close to zero (we have used 0.25).

- If the second data point \mathbf{x}^j falls outside the boundary of representative object n_1 , i.e., Euclidian norm, $\|\mathbf{w}_1^{(1)} - \mathbf{x}^j\| > d$, representative object n_2 is introduced with weight vector $\mathbf{w}_2^{(0)}$ is assumed to be $[0, 0, \dots, 0]^T \in \mathfrak{R}^l$. The weight vector $\mathbf{w}_2^{(0)}$ is updated using the equation $\mathbf{w}_2^{(1)} = \mathbf{w}_2^{(0)} + t(\mathbf{x}^j - \mathbf{w}_2^{(0)})$, where t is the updating coefficient which can vary from $0 \leq t \leq 1$. A higher value of t close to 1 (we have used 0.9) is preferred as in representative object n_1 .
- The above two decision rules are applied to all other subsequent data points until the last data point is clustered.

Overlap of boundaries of neighboring clusters is allowed in the proposed algorithm. A data point in the overlap zone belongs to the cluster having the lowest distance between the data point and the representative object of that cluster. If a data point falls at equal distance from more than one representative object, preference is given in a chronological order, i.e., representative object n_k must be preferred before representative object n_{k+1} and so on. There is no specific argument for this preference. The flowchart of the proposed clustering algorithm is shown in Figure 4-4.

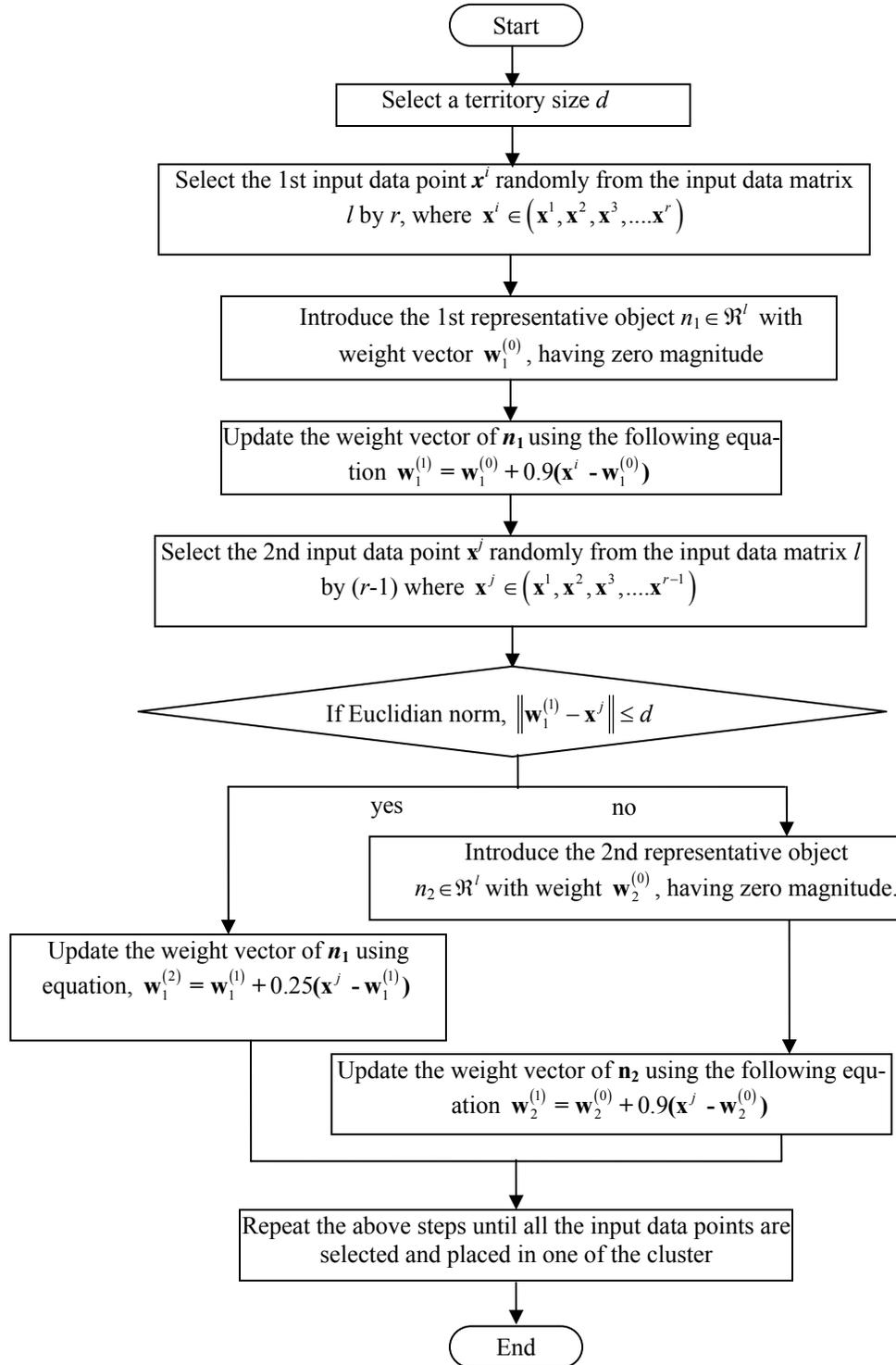


Figure 4-4 Flowchart of the proposed CDCA clustering algorithm

Special cases:

- Suppose for a certain data point, more than one representative object satisfies the belongingness criteria, i.e. $\|\mathbf{w}_k^{(g)} - \mathbf{x}^i\| \leq d$, in this case preference is given to the closet representative object.
- If a data point lies exactly at the same distance from more than one representative object, preference is given in a chronological order, i.e. representative object n_1 must be preferred before representative object n_2 and so on depending on the number of representative objects.

The above steps are repeated for any territory size d . As mentioned earlier, the territory size d was given by the user. Thus, it is important to know the optimal territory size $d_{optimal}$. Theoretically, the minimum territory size d_{min} is obtained when the number of clusters k , is equal to the total number of input data points r . Similarly, the maximum territory size d_{max} is obtained when the number of clusters k , is equal to one. But in this work our objective is to divide the entire data space into smaller clusters of equal size. These smaller clusters must carry a sufficient number of data points to participate in the data allocation process in Step 2, as shown in Figure 4-1. From trial and error it is observed that d_{min} can be limited to a value at which the number of clusters will be $r/2$. Similarly, d_{max} can be limited to a value at which the number of clusters will be 2. In this work exhaustive search technique is used between d_{min} and d_{max} to find the optimal value of $d_{optimal}$. Bowden et al. (2002) and May et al. (2010) utilized silhouette coefficient SC to

evaluate the number of cluster k in FC and SOM. The same strategy is utilized to optimize territory size d which is discussed in the next section.

4.3.4 Optimization of Territory Size d using Silhouette Coefficient SC

In the previous section the functioning of the proposed CDCA algorithm for a particular territory size d was explained. In this section silhouette coefficient SC is used to optimize the territory size d . Silhouette coefficient SC (Kaufman and Rousseeuw, 1990) is a very standard procedure to estimate the clustering quality irrespective of the clustering algorithm. Previously, May et al. (2010) and Shahin et al. (2004) implemented silhouette coefficient to estimate the clustering quality of SOM and fuzzy clustering algorithm. SC is calculated for every territory size d . The highest SC value gives us the optimal territory size $d_{optimal}$, hence, the best clustering.

SC can be defined as the maximal average silhouette width for the entire data set. There are several building blocks associated in calculating SC . First of all, silhouette value $s(\mathbf{x}^i)$ of every data point is calculated. For example, for any data point \mathbf{x}^i , in cluster n , $s(\mathbf{x}^i)$ is computed as follows:

$$s(\mathbf{x}^i) = \frac{b(\mathbf{x}^i) - a(\mathbf{x}^i)}{\max\{a(\mathbf{x}^i), b(\mathbf{x}^i)\}}$$

where $a(\mathbf{x}^i)$ and $b(\mathbf{x}^i)$ are given as

$$a(\mathbf{x}^i) = \frac{1}{|C_n|} \sum_{\substack{\mathbf{x}^j \in C_n \\ \mathbf{x}^j \neq \mathbf{x}^i}} \|\mathbf{x}^i - \mathbf{x}^j\|$$

$$b(\mathbf{x}^i) = \min \{c(\mathbf{x}^i)\}$$

and $c(\mathbf{x}^i)$ is expressed as

$$c(\mathbf{x}^i) = \frac{1}{|C_k|} \sum_{\substack{\mathbf{x}^j \in C_k \\ k \neq n}} \|\mathbf{x}^i - \mathbf{x}^j\|$$

where $a(\mathbf{x}^i)$ measures the average intra-cluster distance between data point \mathbf{x}^i and rest of the data points in the same cluster, which is, cluster n , $|C_n|$ is the total number of data points in cluster n , $b(\mathbf{x}^i)$ measures the lowest $c(\mathbf{x}^i)$ while $c(\mathbf{x}^i)$ measures the average inter-cluster distance between data point \mathbf{x}^i and all data points that are not in cluster n , and $|C_k|$ is the total number of data points in cluster k . Lower $a(\mathbf{x}^i)$ value indicates that data point \mathbf{x}^i is closely packed with the rest of the data points in cluster n and vice versa.

Silhouette value $s(\mathbf{x}^i)$ varies between $[-1, 1]$. When $s(\mathbf{x}^i)$ is at its largest (close to 1), this implies that the intra-cluster distance $a(\mathbf{x}^i)$ is much smaller than the smallest inter-cluster distance $b(\mathbf{x}^i)$. Therefore it may be said that data point \mathbf{x}^i is well classified. Thus, a higher value of $s(\mathbf{x}^i)$ indicates that the data point \mathbf{x}^i is more compactly clustered in the current cluster than its nearest neighboring cluster. When $s(\mathbf{x}^i)$ is at its smallest (close to -1), this implies that the intra-cluster distance $a(\mathbf{x}^i)$ is much higher than the smallest inter-cluster distance $b(\mathbf{x}^i)$. Therefore it may be said that data point \mathbf{x}^i is much closer to the neighbor-

ing cluster than the currently assigned cluster. So we can almost conclude that the data point \mathbf{x}^i is misclassified. When a cluster is a singleton or consists of a single data point, $s(\mathbf{x}^i)$ is defined as zero. Likewise, $s(\mathbf{x}^i)$ of every data point is computed. Then, the average silhouette width \tilde{s} is defined as follows:

$$\tilde{s} = \frac{1}{N} \sum_{i=1}^N s(\mathbf{x}^i)$$

where N is the total number of data points in the cluster.

Likewise, other \tilde{s} values can be calculated for all other territory sizes $d_{min} \leq d \leq d_{max}$. Eventually, silhouette coefficient SC is obtained by finding the maximum \tilde{s} . In the next section, the data sampling scheme adopted in this work shall be discussed. Kaufman and Rousseeuw (1990) recommended Table 4-2 to interpret cluster quality based silhouette coefficient SC .

Table 4-2 Subjective interpretation of the silhouette coefficient SC

SC	Proposed interpretation
0.71-1.00	A strong structure has been found
0.51-0.70	A reasonable structure has been found
0.26-0.50	The structure is weak and could be artificial; please try additional methods on this dataset
≤ 0.25	No substantial structure has been found

In this section, how to obtain the optimal territory size d and thereby to find optimal clusters was discussed. Once this task is over, the next step in stratified data partitioning is to extract data points from each of these clusters as men-

tioned in Section 4.2. Data points are often sampled in a predefined data sampling scheme which is discussed in detail in the next section.

4.4 Data Sampling Scheme

In stratified data partitioning each stratum is sampled as an independent sub-population, out of which data points can be randomly selected. Different allocation or sampling approaches can be applied to different strata, potentially enabling to use the best suited approach for each identified stratum. May et al., (2010) mentioned about different kind of allocation schemes such as equal allocation, proportional allocation and Neyman allocation in their work. In this work, a combination of proportional allocation and heuristic rules to sample data points is used. In the following paragraph, the reason for this preference is explained.

Let us assume an example of clustered dataset. Suppose the number of stratum = h , total number of data points in any stratum $s = N_s$, where $s=1, \dots, h$, thus, the total number of data points in the whole dataset = $\sum_{s=1}^h N_s$. Suppose the data sampling ratios of training, validation and test dataset are $\rho_{training}$, $\rho_{validation}$ and ρ_{test} in percentage.

In equal allocation, equal numbers of data points are extracted from every stratum s irrespective of the stratum size as per the following formula:

Number of data points to be extracted from any stratum s to the training

$$\text{dataset} = \frac{\rho_{\text{training}} \sum_{s=1}^h N_s}{h}; \text{ for validation dataset} = \frac{\rho_{\text{validation}} \sum_{s=1}^h N_s}{h}; \text{ for test dataset}$$

$$= \frac{\rho_{\text{test}} \sum_{s=1}^h N_s}{h}$$

It is to be noted that to achieve a true representation of every stratum, individual stratum size must be considered in the above formulas. This is why equal allocation has not been considered in this thesis.

In proportional allocation, proportional number of data points is extracted from every stratum s depending on the stratum size as per the following formula:

Number of data points to be extracted from any stratum s to the training dataset = $\rho_{\text{training}} N_s$; for validation dataset = $\rho_{\text{validation}} N_s$ and for test dataset = $\rho_{\text{test}} N_s$

Higher the number of data points in the stratum, higher is its representation in the extraction process. This is a more reasonable strategy considering the requirement of statistical representativeness, as mentioned in Chapter 3 Section 3.3. This strategy has been adopted in this thesis.

Neyman allocation is the same as proportional allocation except the fact that standard deviation of any stratum σ_s , where $s=1, \dots, h$, is considered in addition to the above formulas as follows:

Number of data points to be extracted from any stratum s to the training dataset = $\rho_{\text{training}} N_s \sigma_s$; for validation dataset = $\rho_{\text{validation}} N_s \sigma_s$; for test dataset

$$= \rho_{test} N_s \sigma_s$$

Neyman allocation gives consideration to the statistical spread in individual stratum but there is no clear evidence that such strategy is more effective in comparison to the proportional allocation. Further since the proposed CDCA algorithm tries to form stratum of approximately equal size (d value), considering standard deviation might not be necessary. This is the reason why proportional allocation scheme has been selected to be used in this thesis.

Two scenarios, namely, (70%:20%:10%) and (50%:40%:10%) has been selected. In the second scenario, we divert 20% of training data to the validation dataset where the test dataset remains same as 10%. The idea is to use more data points for cross validation by reducing the training data in comparison with scenario 1. This will have different impact on the network generalization ability. Thus, the effect of the proposed method at two different scenarios of network generalization is measured.

Proportional allocation rules fail when $\rho_{training} N_s$, $\rho_{validation} N_s$, $\rho_{test} N_s$ are not integers. It is not possible to obtain clusters having data points as a multiple of ten which makes the sampling straight forward. This is where heuristic rules help the sampling process as earlier done by Bowden et al. (2002). In this thesis, the following heuristics rules have been proposed.

For scenario 1: 70% training, 20% validation and 10% test

- If the number of data vectors in any cluster is less than 9, we use Table 4-3

to extract data points.

- If the number of data vectors in any cluster is more than 9, this number is expressed as a sum of the nearest multiple of 10 and the remainder. For example, 25 is expressed as 20 plus 5. Now, randomly select 5 data vectors out of 25 and extract them according to Table 4-3.

Table 4-3 70%:20%:10% heuristic rules

No. of data points in cluster, 'c'	Desired			Actual		
	70% Training	20% Validation	10% Test	Training	Validation	Test
1	0.7	0.2	0.1	1	0	0
2	1.4	0.4	0.2	1	1	0
3	2.1	0.6	0.3	2	1	0
4	2.8	0.8	0.4	2	1	1
5	3.5	1	0.5	3	1	1
6	4.2	1.2	0.6	4	1	1
7	4.9	1.4	0.7	5	1	1
8	5.6	1.6	0.8	5	2	1
9	6.3	1.8	0.9	6	2	1

For scenario 2: 50% training, 40% validation and 10% test

If the number of data vectors in any cluster is less than 9, we use Table 4-4 to extract data points.

If the number of data vectors in any cluster is more than 9, we proceed as in scenario 1 and extract them.

Table 4-4 50%:40%:10% heuristic rules

No. of data points in cluster, 'c'	Desired			Actual		
	50% Training	40% Validation	10% Test	Training	Validation	Test
1	0.5	0.4	0.1	1	0	0
2	1	0.8	0.2	1	1	0
3	1.5	1.2	0.3	2	1	0
4	2	1.6	0.4	2	2	0
5	2.5	2	0.5	3	1	1
6	3	2.4	0.6	3	2	1
7	3.5	2.8	0.7	4	2	1
8	4	3.2	0.8	4	3	1
9	4.5	3.6	0.9	5	3	1

These heuristic rules are used so that the actual sampling would be close enough to the desired sampling ratio. Those data points that are multiples of 10 are sampled into three sub-groups according to proportional allocation rule.

4.5 The General Steps of the Proposed Data Partitioning Method

In this section we combine the proposed CDCA clustering algorithm, the silhouette coefficient and the data sampling scheme to form the proposed data partitioning method. The following steps are to be followed.

Step 1: Start the clustering process using the CDCA algorithm mentioned in Section 4.3.3 for the minimum territory size d_{min} which is when the number of clusters would be equal to half of the total number of data points.

Step 2: Evaluate the average silhouette value \tilde{s} as mentioned in Section 4.3.4.

Step 3: Perform Step 1 and 2 for ten more iterations to ensure that the best average silhouette value \tilde{s} is obtained. (The number of iterations was determined as ten only after ensuring the fact that these random iterations have negligible effect on the average silhouette value of the entire dataset.)

Step 4: Perform the clustering using the CDCA algorithm for all other d values and repeat Step 2 and 3. For convenience only integer values of d are used.

Step 5: This process is continued until d_{max} is reached where d_{max} is the territory size where the number of clusters is two.

Step 6: The optimal territory size $d_{optimal}$ is obtained for the maximum average silhouette value \tilde{s} which is also known as silhouette coefficient SC .

Step 7: Determine the data sampling ratio (for example, training 70%: validation 30%: test 10%). Here two such ratios as mentioned in Section 4.4 are used for better illustration.

Step 8: Use the proposed sampling scheme to extract data points into training, validation and test datasets, respectively.

Step 9: Eventually the ANN model is trained, cross-validated and tested.

4.6 Reported Data Partitioning (DP) Algorithms for Comparison

In this study, three most widely reported data partitioning methods are considered for comparison with the proposed DP method. From the SDP class, two DP algo-

rithms, namely, SOM and fuzzy clustering are selected while from the ODP class, one algorithm, namely, genetic algorithm (GA) is selected. The proposed algorithm, SOM, fuzzy clustering belongs to the SDP class, thus, it is a natural choice for comparison in this work. Genetic algorithm (GA) being one of the most reported data partitioning algorithm is another choice. In the subsequent sections we describe each of these algorithms in details.

4.6.1 Self Organizing Map Based Data Partitioning (SOMDP)

Self organizing map belongs to the category of unsupervised neural networks, mainly used for clustering application. Over the years researchers have exploited this ability in stratified data partitioning process. SOM utilizes the competitive transfer function mentioned in Table 2-2 of Chapter 2. The network consists of two layers, namely, an input layer and a competitive layer. The output of competitive layer is the output of the network. During the learning process input vectors are introduced to the network and all neurons in the competitive layer compete with each other in this process. The neuron closest to the input vector gives an output of one and is considered as the winner in this competition. All other neurons return zero and hence, are considered as losers in this competition. Suppose the i th neuron wins, the i th row of the input weight matrix are adjusted with the Kohonen learning rule (Kohonen, 2001) as shown below.

$$w(q) = w(q-1) + \alpha(p(q) - w(q-1))$$

where α is the learning rate; p is the input; q is the time sequence; and w is the

weight parameter. As more and more inputs are presented, each neuron in the layer closest to a group of input vectors soon adjusts its weight vector toward those input vectors. The neurons in the layer of a SOM are arranged originally in physical positions according to a topology function. The MATLAB (<http://www.mathworks.com>) functions `gridtop`, `hextop` or `randtop` can arrange the neurons in a grid, hexagonal, or random topology respectively. Distances between neurons are calculated from their positions with a distance function. There are four distance functions in MATLAB, namely, `dist`, `boxdist`, `linkdist` and `mandist`. For the research presented in this thesis, `hextop` and `linkdist` have been used after a few trial and errors. A detailed discussion about SOM is given in Hagan et al. (1996) and Kohonen (2001). The following steps are adopted for implementing SOM into the data partitioning work.

Step 1: Clustering process is started with a 2 by 2 hexagonal topology. MATLAB function `newsom` is used for creating SOM network.

Step 2: For an efficient learning, the network is trained for 1000 iterations. This may vary depending on the size of datasets and experience.

Step 3: Once the available data are clustered, the average silhouette value \tilde{s} is calculated.

Step 4: Then, the topology is changed to 4 by 4 and step 2 to 3 are repeated.

Step 5: Likewise, the average silhouette value \tilde{s} is evaluated for 8 by 8, 16 by 16 and 20 by 20 hexagonal topologies.

Step 6: The optimal number of clusters is obtained at the maximum average silhouette value \tilde{s} which is also known as silhouette coefficient SC .

Step 7: Data sampling is done according to the proposed sampling scheme in Section 4.4.

Step 8: Eventually the ANN model is trained, cross-validated and tested.

4.6.2 Fuzzy Clustering Based Data Partitioning (FCDP)

Fuzzy clustering aims to minimize the following objective function by finding the membership values u_{iv} :

$$C = \sum_{v=1}^k \frac{\sum_{i,j=1}^n u_{iv}^2 u_{jv}^2 d(i,j)}{2 \sum_{j=1}^n u_{jv}^2}$$

where $d(i,j)$ represents the distance between data points i and j , where $i, j = 1, \dots, n$; u_{iv} is the membership of data point i to cluster v , where $v = 1, \dots, k$ and k is the given number of clusters. The sum in the numerator ranges over all pairs of data points. The membership values are subject to the following constraints.

$$u_{iv} \geq 0 \text{ for } i=1, \dots, n; v=1, \dots, k$$

$$\sum_v u_{iv} = 1 \text{ for } i=1, \dots, n$$

These constraints mean that a membership cannot be negative and each data point belongs to all clusters with a certain membership value. The total membership value of a data point for all clusters is always one. Membership values of

each data points are evaluated by solving the above objective function. Data point i is assigned to cluster v if cluster v has the highest membership value for data point i . More details on fuzzy clustering algorithm can be found in (Kaufman and Rousseeuw, 1990). The following steps are adopted from Shahin et al. (2004) for implementing fuzzy clustering into the data partitioning work.

Step 1: An initial number of clusters, not less than two, are chosen for the initial number of clusters.

Step 2: The available data are clustered using the fuzzy clustering technique. MATLAB function `fcm` is used.

Step 3: The average silhouette width \tilde{s} of the entire data set is calculated.

Step 4: The number of clusters is increased by one and step 2 is repeated until \tilde{s} remains constant or the number of clusters reaches 50% of the available data.

Step 5: The optimal number of clusters is obtained at the maximum average silhouette value \tilde{s} which is also known as silhouette coefficient SC .

Step 6: Data sampling is done according to the proposed sampling scheme in Section 4.4.

Step 7: Eventually the ANN model is trained, cross-validated and tested.

4.6.3 Genetic Algorithm Based Data Partitioning (GADP)

Previously, Bowden et al. (2002) implemented GADP to obtain the optimal data

partitioning. They conducted a random search of 100,000 combinations of data splits for their water resource data which consists of 2005 data samples. As an example, suppose there are 10 data samples, each having a distinct index from 1 to 10. If these index are divided randomly and data is allocated as follows: 60% for training, 20% for validation and 10% for testing the arrangements that can be possible solutions are as shown in Figure 4-5.

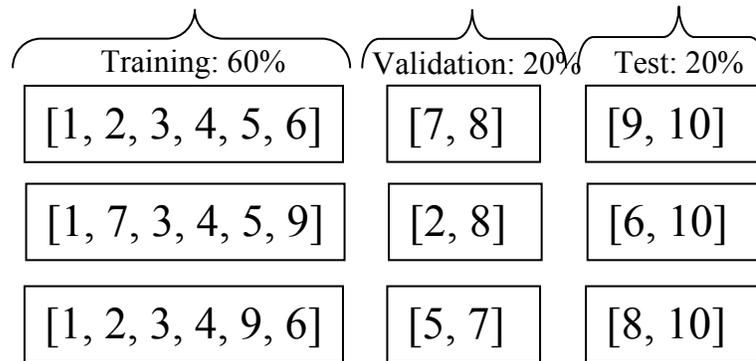


Figure 4-5 Random sampling of 10 data samples into training: 60%, validation: 20% and test: 20%

For the water resource data of 2005 data samples, the total number of possible combinations are much higher than 100,000. However, Bowden et al. (2002) considered this to be a reasonable size search space. In our research, the largest dataset consists of 1000 data samples so we believe 10,000 would be a reasonable search space. It is understandable that the higher the search space, the higher is the computational time. Thus, a trade off is always made between CPU time and the quality of data partitioning. The following objective function is used.

$$f = \sum_{i=1}^n |\mu_{i,T} - \mu_{i,S}| + |\mu_{i,S} - \mu_{i,V}| + |\mu_{i,V} - \mu_{i,T}| + |\sigma_{i,T} - \sigma_{i,S}| + |\sigma_{i,S} - \sigma_{i,V}| + |\sigma_{i,V} - \sigma_{i,T}|$$

where $\mu_{i,T}$, $\mu_{i,S}$ and $\mu_{i,V}$ denote the mean while $\sigma_{i,T}$, $\sigma_{i,S}$ and $\sigma_{i,V}$ are standard deviations of input feature i , where $i = 1, \dots, n$ (n is the number of input features) in the training, validation and test dataset. The following steps are adopted for implementing genetic algorithm into the data partitioning work.

Step 1: Select the percentage of data points in training, validation and test dataset. In this research, two different ratios have been used, namely, (1) 70% for training, 20% for validation and 10% for testing (2) 50% for training, 40% for validation and 10% for testing.-

Step 2: An initial population size of 20 is chosen out of the search space (10,000 solutions). In this population each member represents one candidate solution and a random arrangement of data samples in three sub-groups such as training, validation and test data.

Step 3: Each of these solutions is evaluated using the above mentioned objective function and the best solution is recorded.

Step 4: Using the roulette wheel scheme (Chong and Žak, 2008), best solutions are selected into the mating pool until the mating pool has 20 members.

Step 5: Linear crossover ($0.5S_i + 0.5S_j$) with a crossover probability of 0.7 is used to perform crossover operation. S_i and S_j represents random solution index and $i, j = 1, \dots, 10,000$.

Step 6: Non-uniform mutation equation from Bowden et al. (2002) is used with a mutation probability of 0.01 to perform mutation.

Step 7: Repeat Step 3 to Step 6 for 200 iterations or until the algorithm converges, whichever ever happen earlier.

Step 8: Eventually the ANN model is trained, cross-validated and tested.

Up to this point we have discussed the proposed DP method against the reported DP method. A brief summary is presented in the next section to conclude this chapter.

4.7 Summary

The SDP class data partitioning algorithms have certain advantages over the ODP class in terms of computational complexities. This is the reason why for this thesis, we were more inspired by the SDP class. The basic principle of SDP class is to divide the entire dataset into smaller clusters, thereby sampling data points into training, validation and test sub-groups. In this chapter the abilities of standard cluster algorithm like SOM and FC have been challenged to achieve smaller clusters in which individual data points are in close statistical agreement inside a stratum. A novel CDCA clustering algorithm is proposed which can divide the entire data space into smaller clusters of equal territorial size. This strategy will further enhance the potential of achieving better intra-cluster statistical agreement. The optimal territory size was decided by using silhouette coefficient SC . Proportional allocation rule is used for sampling data points from individual clusters. In the

next chapter the performance of the proposed data partitioning algorithm is compared with the other existing data partitioning algorithms.

CHAPTER 5

EVALUATION OF THE PROPOSED DATA PARTITIONING ALGORITHM

5.1 Introduction

In Chapter 4, we have explained the proposed data partitioning algorithm. In this Chapter, we need to evaluate and compare the performance of the proposed algorithm with the reported data partitioning methods as mentioned in Chapter 4. Based on literature review, it is observed that SOM based data partitioning (SOMDP), fuzzy clustering based data partitioning (FCDP) and genetic algorithm based data partitioning (GADP) are the most widely used data partitioning algorithms. This is the reason we have selected the abovementioned algorithms in this work. Other reasons of including SOMDP and FCDP were because of the fact that the proposed algorithm belongs to the same category (SDP). One of the reasons GADP is included in our comparison is because, it is one of the most studied DP algorithms.

Previously researchers used statistical measures to check the validity of the DP algorithms (Bowden et al., 2002; Samanta et al., 2004 and Shahin et al., 2004). After dividing the whole dataset into training, validation and test sub-groups, mean and standard deviations of the three sub-groups were calculated and compared for every individual input feature vectors. Decisions on the best data

partitioning method were made on the number of input features vectors in close statistical agreement. The more the number of feature vectors that are in agreements, the better the method is. Although such strategies holds good from a data partitioning or statistical point of view, but the final decision on the efficacy of the method must be based on the final performance of the ANN model. Some of the researchers have included ANN model performance in addition to the above decision strategy but greater importance was given to the decision strategy not the network performance. Very recently, May et al. (2010) used network bias and variance of ANN model to assess the performance of data partitioning algorithm. In this work, network bias and variance are chosen to measure the validity of the proposed method. More discussions on the selection of performance measures are given in Section 5.5. As we mentioned before, we also compare the CPU time for computational complexity check.

The computational complexity in terms of CPU time was never checked before for any of these reported data partitioning methods. In the literature computational complexity for algorithms were checked by convergence speed or number of iterations and sometimes the big-O notation (Chong and Žak, 2008). In this research CPU time is a worth investing parameter for computational complexity since data preparation is a time consuming process. Yu et al. (2007) showed that nearly (60-70) % time is spent on data processing in any data analysis job. As the input data size increases the processing time also increases significantly. In this work datasets ranging from small, medium and large data size are tested.

Application of ANN has been mainly reported in three problem domains, namely, classification, function approximation and prediction. Therefore, three benchmark datasets one from each category is selected to validate the proposed methodology. For classification, ultrasonic experimental data of size (10 x 343) from Sahoo et al. (2007) is used; for function approximation, Friedman regression function data of (6 x 1000) from May et al. (2010) is used; for prediction, housing dataset of (14 x 506) from Blake and Merz (1998) is used. For the size of data matrix, the first number represents the number of rows or the number of features including one output or the dimensions of a single data point. The second number represents the number of columns or the number of data points. Each of these datasets are explained in the following sections. Finally, the impact of data sampling ratio is also taken into consideration by undertaking experiments at different proportion of training, validation and test dataset using 70% : 20% : 10% and 50%:40%:10% sampling ratios. In the following sections we explain the experimental datasets.

In the sections to follow, we present this chapter as follows: In Section 5.2, 5.3 and 5.4 we describes the three datasets used in this work. We discuss the performance measures used in the work in Section 5.5. Section 5.6 presents the ANN model selection strategies implemented in this work. In Section 5.7 we cover the results and discussions part. Finally, Section 5.8 we give a summary of this chapter.

5.2 Friedman Regression Function Dataset

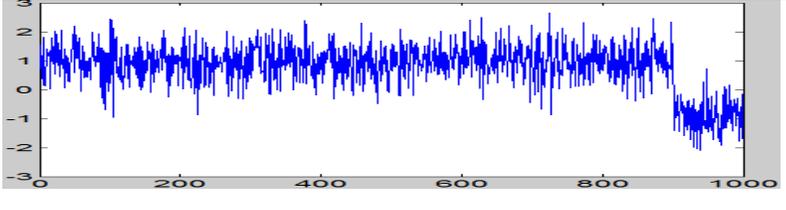
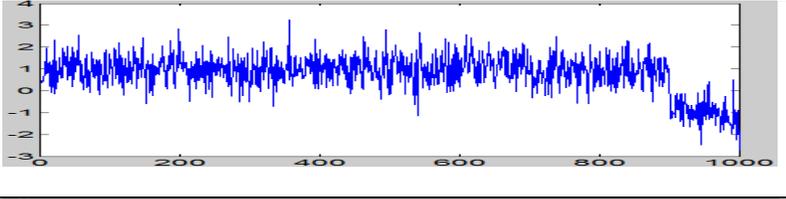
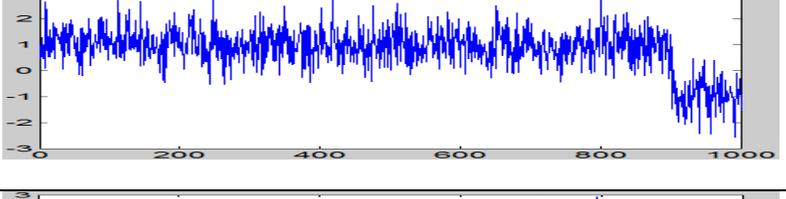
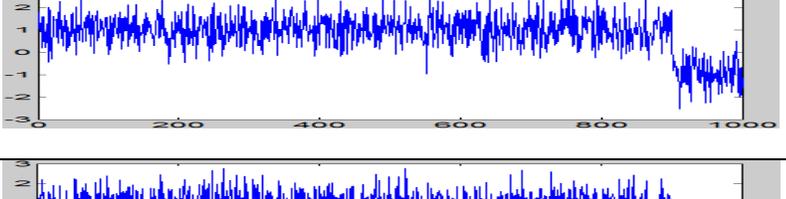
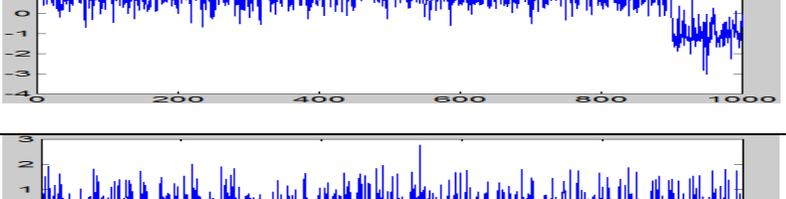
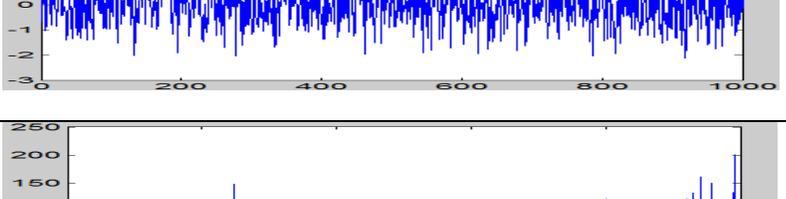
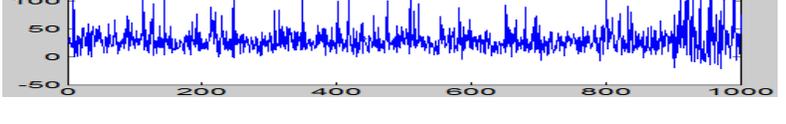
May et al. (2010) used Friedman regression function to assess the performance of data partitioning algorithms in their work. In this work the same function is used as one of the dataset to analyze function approximation problem. The Friedman regression function as shown below for regression applications is difficult to approximate and has a five dimensional input space.

$$y = 5(2 \sin(\pi x_1 x_2) + 4(x_3 - 0.5)^2 + 2x_4 + x_5) + \varepsilon$$

where ε (Gaussian noise) follows the normal distribution with mean 0 and variance 0.8, denoted by $N(0, 0.8)$. Input variables (we call features in this work) x_1, x_2, x_3, x_4 and x_5 were extracted from a mixture of two Gaussian distribution, generated by sampling 90% of data from the distribution $x_i \sim N(1, 0.6)$, and 10% of data from the distribution $x_i \sim N(-1, 0.6)$. We generate 1000 data points for this dataset. Thus, the Friedman regression dataset becomes a matrix of the size of 6x1000 which includes five input vectors and one output vector.

They have set the benchmark by generating six different datasets generated by controlling the characteristics of the input domain, such as skewness, noise and correlation between input variables, and number of data. This is achieved by sampling independent input variables from independent normal distributions of different mean and variance. For our investigation we adapt similar distributions used for dataset I in May et al. (2010). Table 5-1 shows the minimum and maximum values of the input, output variables, Gaussian noise and the corresponding feature trends. Each of these plot represents 1000 data samples on horizontal axis and their corresponding values on the vertical axis.

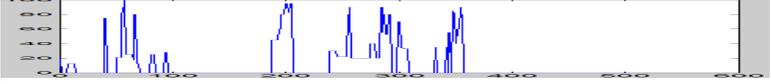
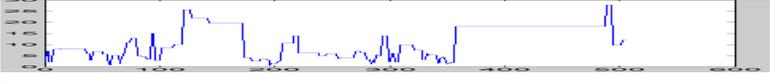
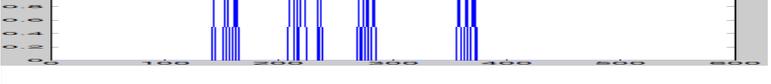
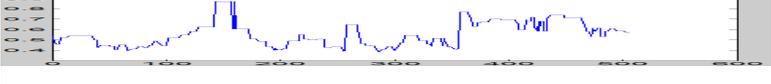
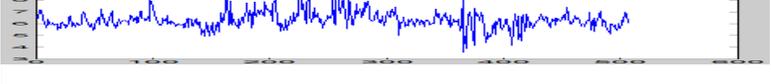
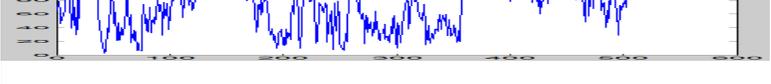
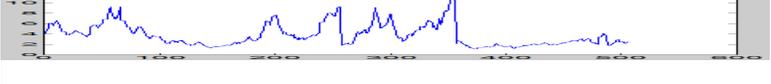
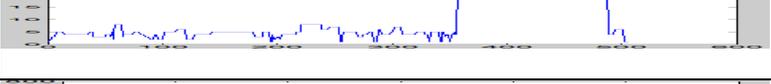
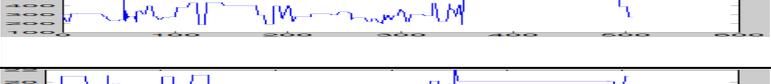
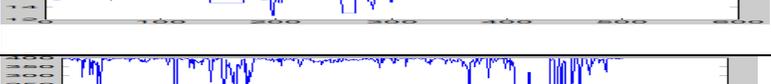
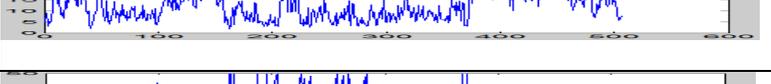
Table 5-1 Friedman regression dataset feature trends

Features	Value range	Individual feature trends of 1000 data sample
Input (x_1)	-2.1 - 2.66	
Input (x_2)	-2.74 - 3.25	
Input (x_3)	-2.57 - 2.94	
Input (x_4)	-2.54 - 2.93	
Input (x_5)	-3.04 - 2.77	
Noise (ϵ)	-2.12 - 2.77	
Output (y)	-21.29 - 200.54	

5.3 Housing Dataset

We obtain housing dataset from Blake and Merz (1998). The housing dataset consists of 506 data points which includes 13 input real estate parameters such as; per capita crime rate by town (input 1), proportion of residential land zoned for lots over 25,000 sq ft.(input 2), proportion of non-retail business acres per town (input 3), Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) (input 4), nitric oxide concentration (parts per 10 million) (input 5), average number of rooms per dwelling (input 6), proportion of owner-occupied units built prior to 1940 (input 7), weighted distances to five Boston employment centers (input 8), index of accessibility to radial highways (input 9), full-value property-tax rate per \$10,000 (input 10), pupil-teacher ratio by town (input 11), $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town (input 12), % lower status of the population (input 13), and one output to predict as median value of owner-occupied homes in \$1000's. Thus, the housing dataset becomes a matrix of the size of 14x506. The objective of this dataset is to predict the housing cost using ANN models. Table 5-2 shows the minimum and maximum values of the input and output variables and the corresponding feature trends.

Table 5-2 Housing dataset feature trends

Features	Value range	Individual feature trends of 506 data sample
Input 1	0.01 -88.98	
Input 2	0-100	
Input 3	0.46- 27.74	
Input 4	0-1	
Input 5	0.39-0.87	
Input 6	3.56-8.78	
Input 7	2.9-100	
Input 8	1.13- 12.13	
Input 9	1-24	
Input 10	187-711	
Input 11	12.6-22	
Input 12	0.32- 396.9	
Input 13	1.73- 37.97	
Output	5-50	

5.4 Ultrasonic Dataset

An experiment was conducted in May 2007 by the Reliability Research Group. The experimental procedures and data collection details were documented in (Zhang et al., 2007). In the following the experimental setup and data collection procedure is briefly described.

5.4.1 Experimental Setup

Ultrasonic experimental setup available in Reliability Research Lab, University of Alberta was used to collect ultrasonic data. Figure 5-1 shows a snapshot of the experimental setup. The system consists of the OmniScan data acquisition system. Bi-slide system was used to achieve an automated one dimensional movement of the sensor. The Bi-slide system allows automated and accurate positioning of the ultrasonic transducer on the specimen's surface. A Windows-based GUI called COSMOS was used to program the bi-slide movement. It was programmed to linearly move the transducer a distance of 30mm with a step size of 0.25mm. At every step, a pulse was sent to the OmniScan unit to mark the current position of the transducer.

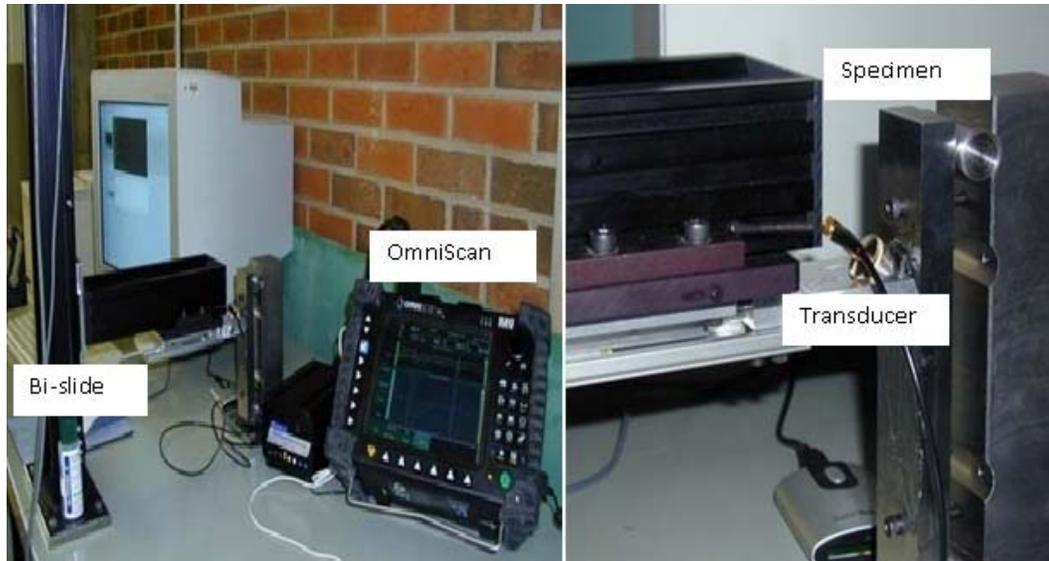


Figure 5-1 Ultrasonic experimental setup

In the experiment seven 4140-steel specimens (185mm x 40 mm x 16 mm) were prepared using the electric discharge machine (EDM). Cracks of different depths were simulated by creating slots of different depths. The seven specimens have the slot depths as follows: 0, 0.5, 1.0, 1.5, 2.0, 2.5 and 3.0 in mm. The length of each EDM cut was 40 mm (each cut was a full length cut along one dimension of the specimen).

5.4.2 Data Collection Process

An ultrasonic pulse was generated and transmitted to the specimen with a step size of 0.25mm of the Bi-slide movement. The step size was carefully chosen so that the ultrasound covers the whole path of its movement. The reflected ultrasonic echo was recorded 2048 times by the OmniScan. For a travel distance of 30mm with a step size of 0.25mm, there were 121 positions for ultrasonic data

collection. Thus, the data file has a size of 121x2048. For seven specimen, seven 121x2048 *.oud data files were collected. Later the *.oud files were converted to *.mat files in the MATLAB environment. In this problem, the goal is to classify seven different levels of crack size (0mm, 0.5mm, 1mm, 1.5mm, 2mm, 2.5mm and 3mm) in a steel specimen using ANN models. Thus, we need to redesign the input data which will include input feature vectors for all seven classes. We decide to extract statistical features such as root mean square (RMS) (input 1), peak (input 2), peak to peak (input 3), time (input 4), kurtosis (input 5), skewness (input 6), shape factor (input 7), impulse factor (input 8), crest factor (input 9) and one output as crack size from the original data file. The definition and formula for each of these statistical features are given in Table 5-3.

Table 5-3 Definition and formula for statistical features

S/N	Feature	Definition	Formula
1	RMS	It is defined as the square root of the mean of squares of samples, x_i , where $i = 1, \dots, n$	$= \sqrt{\frac{\sum_{i=1}^n x_i^2}{n}}$
2	Peak	It is defined as the peak value of absolute	$= \max(x_i)$
3	Peak to peak	It is the range between the lowest peak and the highest peak.	$= \max(x_i) - \min(x_i)$
4	Time	It is defined as the time of travel of the peak signal in time domain. This is calculated by dividing index of the peak signal by 100 in microsecond. Peak index varies from 1 to 2048.	$= \frac{\text{peak index}}{100} \mu \text{sec}$
5	Kurtosis	It is a measure of how outlier-prone a distribution is. The kurtosis of the normal distribution is 3. Distributions that are more outlier-prone than the normal distribution have kurtosis greater than 3 and vice versa.	$= \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^4}{\left(\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2 \right)^2}$
6	Skewness	It is a measure of the asymmetry of the data around its sample mean. If skewness is negative, the data are spread out more to the left of the mean than to the right and vice versa. The skewness of the normal distribution is zero.	$= \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^3}{\left(\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2 \right)^{3/2}}$
7	Shape factor	It is defined as the ratio between the root mean square and the mean absolute	$= \frac{\text{RMS}}{\frac{1}{n} \sum_{i=1}^n x_i }$
8	Impulse factor	It is defined as the ratio between the peak and the mean absolute	$= \frac{\text{peak}}{\frac{1}{n} \sum_{i=1}^n x_i }$
9	Crest factor	It is defined as the ratio between the peak and the root mean square	$= \frac{\text{peak}}{\text{RMS}}$

After a careful observation it was found that signal from the crack was significant only between data collection position 33 to 81 for all seven specimen. After removing the irrelevant data the original data files were reduced to 49x2048. From this dataset, 49 data samples from each of these 49 rows keeping

n as 2048 were extracted. Thus, for seven crack sizes there will be 343 (49x7) data points and hence, the input data matrix obtained become of the size of 9x343. The output vector is of the size of 1x343. Therefore, the ultrasonic dataset became a matrix of 10x343. Table 5-4 shows the input and output feature trends and their maximum and minimum values.

Table 5-4 Ultrasonic dataset feature trend

Features	Value range	Individual feature trends of 343 data sample
RMS	0.5456-10.5389	
Peak	2.0078-99.3882	
Peak to peak	4.0157-187.7333	
Time	0.07-17.28	
Kurtosis	3.6875-52.3703	
Skewness	-0.6842-1.2078	
Shape factor	1.8262-4.4211	
Impulse factor	6.1687-42.3122	
Crest factor	3.3778-9.8089	
Crack size	0-3 mm	

5.5 Performance Measures

5.5.1 Bias and Variance

In general, the validation process in any ANN research involves estimating the overall accuracy in percentage on test data (or in other words, the number of instances the network has given a wrong estimation). Also, the mean square error (*mse*) of the network has been implemented in many reports which is calculated as follows:

$$mse = \frac{1}{M} \sum_{m=1}^M (x_o - x_t)^2$$

where x_o represents the network output; x_t represents the target output and M is total number of output instances. As shown in Chapter 4, the step 2 of the SDP class involves random extraction of data points from individual clusters into training, validation and test dataset. It is possible that every time a new sampling is done, the *mse* may vary. Thus it is essential to capture this variation to give a robust conclusion. May et al. (2010) used bias and variance to capture the bias and variance or sensitivity of ANN model. In this paper, we also use the same performance measures.

Network bias for test data is estimated as the hold out test error from N bootstrap sampling (Twomey and Smith, 1998) of training, validation and test data independently for N models. We have selected N to be 100 for this study. Thus, the bias is determined as (Tong and Liu, 2005):

$$\text{Bias} = \frac{1}{N} \sum_{n=1}^N mse_n$$

which is the average mean square error of the model. Variance of the mean square error is given by

$$\text{Variance} = \frac{1}{N-1} \sum_{n=1}^N [mse_n - E(mse)]^2$$

The lower the bias and the variance, the better is the performance of the ANN model and thus, the data partitioning algorithm.

5.5.2 CPU Time

Computational complexities of the three data partitioning algorithms have never been analyzed so far. Thus we consider the CPU time to analysis the computational complexity of the DP algorithms. In the case of stratified data partitioning (SDP) algorithms like SOMDP, FCDP and the proposed CDCADP algorithm, computational time is distributed among three activities, namely, time spent in clustering the data $T_{clustering}$, time spent in sampling into three sub-groups $T_{sampling}$, and time spent in bias/variance estimation for N bootstrap sampling $T_{bias/variance}$. Thus, the total CPU time is determined as follows:

$$T_{total} = T_{clustering} + T_{sampling} + T_{bias/variance}$$

In the case of optimized data partitioning (ODP) class algorithm like GA, it is distributed among two steps, namely, finding the optimal partition T_{rand_search} , and

time spent in bias/variance evaluation $T_{bias/variance}$. Thus, the total CPU time is determined as:

$$T_{total} = T_{rand_search} + T_{bias/variance}$$

5.6 Network Selection and Training

The main objective of this study is to determine the efficacy of data partitioning algorithms not the ANN model. Therefore it is essential to consider a network which can handle all three kinds of datasets. For this study the feedforward back-propagation (FFBP) network which has versatile application ranging from regression to classification problems was considered. Network design parameters like number of hidden layers, training algorithm were kept same for all three datasets. A single hidden layer with sufficient number of hidden neurons is considered to be efficient enough to deal with any kinds of datasets. Therefore in this work a single hidden layer and the Levenberg–Marquardt back propagation (LMBP) training algorithm (MATLAB function, `trainlm`) was used.

A two layer standard FFBP network was selected. MATLAB function `newff` was used for FFBP network. As discussed in Chapter 2 the input layer is not considered as one layer since there is no real processing in this layer. The hidden layer uses the log-sigmoid activation function in all processing neurons. Researchers have used trial and error method to find the optimal number of hidden neurons in this layer. The output layer uses the `purelin` activation function and has one neuron. The ANN toolbox of MATLAB (<http://www.mathworks.com>) for

design and analysis of the network has been used. Unless stated otherwise, the default parameters like learning rate, stopping criteria, etc suggested by the toolbox were used.

A rule of thumb for selecting the number of hidden nodes relies on the fact that the number of samples in the training set should at least be greater than the number of synaptic weights (Tarassenko, 1998). Fletcher and Goss (1993) also suggested that the appropriate number of nodes in a hidden layer ranges from $(2n^{1/2} + m)$ to $(2n + 1)$, where n is the number of input nodes and m is the number of output nodes.

Our experience confirms the fact that the network performance drops significantly if the number of hidden neurons used are less than the number of input nodes on complex data analysis. In this study, we have designed an experiment and set our range from n to $2n$, where n is the number of input nodes. In this work three datasets of different sizes and domain are used, thus, the number of hidden neurons must be different for each of them. This number is evaluated by testing different number of hidden neurons in abovementioned range as follows:

Each of the datasets is randomly divided into three datasets such as 70% training, 20% validation and 10% test data. Validation data is used to cross-validate the training process. Table 5-5 shows network with varying number of hidden neurons and their test mean square error on each of the three datasets. For housing data and ultrasonic data, only alternate numbers of hidden neurons were tested.

Table 5-5 Varying hidden neurons and their test error

Freidman regression data		Housing data		Ultrasonic data	
Network configuration	Test error	Network configuration	Test error	Network configuration	Test error
5-5-1	44.74	13-13-1	43.41	9-9-1	0.08
5-6-1	39.96	13-15-1	10.18	9-11-1	0.07
5-7-1	45.51	13-17-1	17.63	9-13-1	0.1
5-8-1	45.12	13-19-1	6.85	9-15-1	0.1
5-9-1	44.24	13-21-1	37.11	9-17-1	0.08
5-10-1	38.76	13-23-1	43.96	9-18-1	0.09

Based on the lowest test mean square error results obtained in Table 5-5, it was decided to use network 5-10-1, 13-19-1 and 9-11-1 for Freidman regression data, housing data and ultrasonic data, correspondingly.

5.7 Results and Discussions

In this section, the results obtained for each of the three datasets is discussed.. All computational works were carried out on a single system to avoid any variance in processor speed. We have used a Window PC having Intel(R) Pentium(R) dual CPU T2370 @ 1.73 GHz processor and 32 bit operating system. The MATLAB R2008a software was used for all computation. Each of the three datasets is discussed in three separate sections for clarity. In the next section the results obtained for Friedman regression dataset are explained.

5.7.1 Friedman Regression Function Dataset Results

This section is divided into two sections, namely, 70% training, 20% validation, 10% test and the 50% training: 40% validation: 10% test. For SOMDP, FCDP

and the proposed method clustering was done only once and then subsequently data points were sampled twice for both the ratios. Thus, time spent in clustering $T_{clustering}$ is same for both cases. Table 5-6 shows the results for Friedman Regression Function Dataset. Even though the total CPU time comprises of $T_{clustering}$ or T_{rand_search} , $T_{sampling}$ and $T_{bias/variance}$, $T_{sampling}$ is same for SOMDP, FCDP and the proposed method. It is zero for GADP since there is no separate sampling activity in GADP. Thus, we combine $T_{sampling}$ with $T_{bias/variance}$.

For SOMDP, the total CPU time spent for clustering $T_{clustering}$ was obtained as 2,509 sec. Silhouette coefficient SC was obtained as 0.33 at topology size of 2 x 2. The number of clusters obtained at this topology was 4. For FCDP, the total CPU time spent for clustering $T_{clustering}$ was obtained as 8,339 sec. Silhouette coefficient SC was obtained as 0.39 when the number of clusters is 4.

As mentioned in Section 4.5 of the previous chapter, the lowest territory size d_{min} was estimated as 1 and the highest territory size d_{max} was obtained as 130. The proposed algorithm was evaluated for ten random iterations for which time consumed in clustering $T_{clustering}$ was obtained as 689 sec. Silhouette coefficient SC was obtained as 0.51 at an optimal territory size $d_{optimal}$ of 62. The number of clusters at this territory size was obtained as 7.

5.7.1.1 70% for Training: 20% for Validation: 10% for Test

When compared to the ultrasonic dataset, this dataset is more than double in size when. Thus, the required CPU time was expected to be much higher. For GADP T_{rand_search} was registered as 18,535 sec which was much higher than the other me-

thods. The CPU time for random search based data partitioning method increased linearly as the dataset size became bigger. The difference in time spent on $T_{sampling}$ and $T_{bias/variance}$ for all four methods was almost negligible because of the fact that all four methods have used the same network. The total computational time was computed by adding $T_{clustering}$ or T_{rand_search} with $(T_{sampling} + T_{bias/variance})$. The average drop in total CPU time for the proposed method was calculated as follows:

$$T_{avg_drop} = \left(\frac{\frac{T_{total(SOMDP)} - T_{total(proposed)}}{T_{total(SOMDP)}} + \frac{T_{total(FCDP)} - T_{total(proposed)}}{T_{total(FCDP)}} + \frac{T_{total(GADP)} - T_{total(proposed)}}{T_{total(GADP)}}}{3} \right) \times 100$$

The average drop in the total CPU time was recorded as 83.5%. In terms of bias, the proposed method has registered the lowest value as 34.11. Bias for SOMDP was the highest as 44.80 among all the participating methods. Variance for the proposed method was recorded as 130.1, which was again the lowest. The highest variance was recorded for GADP.

Table 5-6 Friedman dataset results for sampling ratios 70:20:10 and 50:40:10

70% Training: 20% Validation: 10% Test					
Bias and variance			CPU time in sec		
	bias	variance	$T_{clustering}$ or T_{rand_search}	$T_{sampling} + T_{bias/variance}$	T_{total}
SOMDP	44.80	467.6	2,509	241	2,750
FCDP	38.41	143.2	8,339	250	8,589
GADP	44.57	537.6	18,535	245	18,780
The proposed	34.11	130.1	689	239	928
50% Training: 40% Validation: 10% Test					
Bias and variance			CPU time in sec		
	bias	variance	$T_{clustering}$ or T_{rand_search}	$T_{sampling} + T_{bias/variance}$	T_{total}
SOMDP	55.16	471.1	2,509	196	2,705
FCDP	46.41	237.1	8,339	202	8,541
GADP	55.44	556.4	20,682	198	20,880
The proposed	39.21	191.2	689	202	891

5.7.1.2 50% for Training: 40% for Validation: 10% for Test

For this scenario, GADP, T_{rand_search} was recorded as 20,682 sec which was higher than the previous scenario. The average drop in the total CPU time was recorded as 84.12%.

As expected a higher bias was obtained for the 50:40:10 scenario when compared with the 70:20:10 scenario. Bias, for the proposed method was recorded as 39.21 which was the lowest among all the participants. FCDP has also registered the second best bias as 46.41. Variance for the proposed method was obtained as 191.2 which was the lowest. The next best variance to the proposed method was again FCDP. GADP has registered the highest bias and variance. The proposed method became the overall winner in all three validation metrics for this dataset.

5.7.2 Housing Dataset Results

Table 5-7 shows the results for housing dataset. For SOMDP, the total CPU time spent for clustering $T_{clustering}$ was obtained as 1,380 sec. Silhouette coefficient SC was obtained as 0.38 at topology size of 2 x 2. The number of clusters obtained at this topology was 4. For FCDP, the total CPU time spent for clustering $T_{clustering}$ was obtained as 1,318 sec. Silhouette coefficient SC was obtained as 0.41 when the number of clusters was 10.

As mentioned in Section 4.5 of the previous chapter, the lowest territory size d_{min} was estimated as 13 and the highest territory size d_{max} was obtained as 335. The proposed algorithm was evaluated for ten random iterations for which time consumed in clustering $T_{clustering}$ was obtained as 944 sec. Silhouette coefficient SC was obtained as 0.6 at an optimal territory size $d_{optimal}$ of 78. The number of clusters at this territory size was obtained as 13.

5.7.2.1 70% for Training: 20% for Validation: 10% for Test

For GADP, time spent on random search, T_{rand_search} was recorded as 8,259 sec which was much higher than any of the clustering based data partitioning methods. The average drop in CPU time was obtained as 45.35%.

The proposed method has given the lowest bias and variance as 30.34 and 198.4. Thus, the proposed method has not only performed well in terms of CPU time but also on bias and variance.

Table 5-7 Housing dataset results for sampling ratios 70:20:10 and 50:40:10

70% Training: 20% Validation: 10% Test					
Bias and variance			CPU time in sec		
	bias	variance	$T_{clustering}$ or T_{rand_search}	$T_{sampling} + T_{bias/variance}$	T_{total}
SOMDP	44.13	360.3	1,380	291	1,671
FCDP	35.34	302.4	1,318	254	1,572
GADP	40.12	384.1	8,259	287	8,546
The proposed	30.34	198.4	944	269	1,213
50% Training: 40% Validation: 10% Test					
Bias and variance			CPU time in sec		
	bias	variance	$T_{clustering}$ or T_{rand_search}	$T_{sampling} + T_{bias/variance}$	T_{total}
SOMDP	44.24	282.5	1,380	246	1,626
FCDP	41.03	287.2	1,318	251	1,569
GADP	43.67	316.7	8,675	265	8,940
The proposed	33.25	203.1	944	271	1,215

5.7.2.2 50% for Training: 40% for Validation:10% for Test

For GADP it was recorded as 8,675 sec. Time spent in estimating bias and variance, $T_{bias/variance}$ was again comparable. The average drop in CPU time was computed as 44.74%.

Bias and variance for the proposed method were registered as 33.25 and 203.1 respectively which were the lowest. As expected, bias for all the participant methods has increased for this scenario. Thus, the proposed method has given both the lowest bias and variance among all the participants.

5.7.3 Ultrasonic Dataset Results

Table 5-8 illustrates bias, variance and CPU time obtained for this scenario. For SOMDP, the total CPU time spent for clustering $T_{clustering}$ was obtained as 918 sec.

Silhouette coefficient SC was obtained as 0.44 at topology size of 2×2 . The number of clusters obtained at this topology is 4.

For FCDP, the total CPU time spent for clustering $T_{clustering}$ was obtained as 274 sec. Silhouette coefficient SC was obtained as 0.48 when the number of clusters was 5.

The lowest territory size d_{min} was estimated as 3 and the highest territory size d_{max} was obtained as 60. The proposed algorithm was evaluated for ten random iterations for which time consumed in clustering $T_{clustering}$ was obtained as 88 sec. Silhouette coefficient SC was obtained as 0.54 at an optimal territory size $d_{optimal}$ of 50. The number of clusters at this territory size was obtained as 5.

5.7.3.1 70% for Training: 20% for Validation: 10% for Test

For GADP T_{rand_search} was registered as 5,160 sec which was again higher than the other methods. There was almost an average drop of 69.54% in total CPU time for the proposed method when compared with its competitors.

Although GADP has given the lowest bias as 0.18 for this scenario, the variance was higher than the rest of the competitors. A higher value of bias indicates a higher mean of 'mse' for the 100 instances of bootstrap sampling. A higher value of variance indicates that the classifier is more unstable and vice versa. Although the proposed method is the second best in terms of bias, next to GADP, in terms of variance it has performed better than rest of the competitors.

Table 5-8 Ultrasonic dataset results for sampling ratios 70:20:10 and 50:40:10

70% Training: 20% Validation: 10% Test					
Bias and variance			CPU time in sec		
	bias	variance	$T_{clustering}$ or T_{rand_search}	$T_{sampling} + T_{bias/variance}$	T_{total}
SOMDP	0.20	0.003	918	171	1,089
FCDP	0.20	0.003	274	182	456
GADP	0.18	0.004	5,160	180	5,340
The proposed	0.19	0.002	88	189	277
50% Training: 40% Validation: 10% Test					
Bias and variance			CPU time in sec		
	bias	variance	$T_{clustering}$ or T_{rand_search}	$T_{sampling} + T_{bias/variance}$	T_{total}
SOMDP	0.21	0.004	918	182	1,100
FCDP	0.25	0.003	274	168	442
GADP	0.31	0.005	4,141	179	4,320
The proposed	0.2	0.003	88	185	273

5.7.3.2 50% for Training: 40% for Validation: 10% for Test

For this scenario, the CPU time for GADP in random search, T_{rand_search} was obtained as 4,141 sec which was still higher than the other three participants. Time spent for $T_{sampling}$ and $T_{bias/variance}$ were again comparable among the competitors. There was an average drop of 69% in total CPU time when compared with its competitors.

In general, it is expected that when number of training data points are reduced, the network performance drops. The bias for the proposed method was registered as 0.2 but the next best bias was obtained for SOMDP which was 0.21. As shown in Table 5-8, variance has gone up by 0.01 for SOMDP, GADP and the proposed method. However, for FCDP variance has remained unchanged at 0.003.

Over all, it can be concluded that the proposed method requires much lower computational time to give similar or even better performance for both the scenarios. The main observations in this study are listed as follows:

- The average reduction in CPU time recorded for the Friedman regression is 83.5% for 70:20:10 and 84.12% for 50:40:10 scenarios.
- The average reduction in CPU time recorded for the housing datasets is 45.35% for 70:20:10 and 44.74% for 50:40:10 scenarios.
- The average reduction in CPU time recorded for the ultrasonic datasets is 69.54% for 70:20:10 and 69% for 50:40:10 scenarios.
- In terms of network performance, the proposed method has always registered lowest variance for all three datasets which means it is the most stable.
- On the other hand, GADP has always registered the highest variance.
- In terms of bias, the proposed method has given the lowest bias for the Friedman regression and the housing datasets. For the ultrasonic dataset, it is the second best for 70:20:10 scenario and the best for 50:40:10 scenario among all the participants.
- From the above discussions, it can be concluded that the proposed method has given an overall better performance than the existing data partitioning methods.

5.8 Summary

This chapter included a detailed discussion of the three dataset used in this work. The input and the output features and their corresponding trend were also plotted. The ultrasonic experimental data was briefly discussed with a main focus on the presentation of the procedure of data preparation from raw ultrasonic signals to the final input and output data. Then performance measures used in this research to compare the participating DP methods were discussed. Bias, variance and CPU time were described. For bias and variance we have used the same formula and bootstrap sampling rate of 100 as in May et al. (2010). The most versatile ANN model that is the FFBP network was considered in this research. This network has been used for both classification and function approximation problems in the literature. The optimal number of hidden neurons in the hidden layer is found by a performance check method. Eventually, we present the results and discussion part. In all three datasets it is observed that the proposed method has significant advantages over the other methods in terms of CPU time. In terms of bias and variance we also observe better results in most cases.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Ensuring statistical agreement for the training, the validation and the test datasets is important for optimal performance of ANN models. In literature, various data partitioning schemes based on SDP as well as ODP methods are proposed to accommodate this requirement. From this study, it is observed that both reported SDP and ODP methods are computationally expensive. Further, SDP methods based on standard clustering algorithms are based on natural clustering algorithms which could adversely affect the data partitioning process in case of large datasets. To overcome these shortcomings, a novel data clustering algorithm has been proposed in this thesis to facilitate data partitioning for the ANN models. A data sampling scheme combining proportional allocation and heuristic rule is introduced too.

The proposed CDCA algorithm and the data sampling scheme are then integrated together in the data partitioning process. Finally, a comparison is made with the reported data partitioning methods like SOMDP, FCDP and GADP. Three datasets from different problem domains such as classification, function approximation and prediction are used in this study. It is observed that the proposed method is more stable in terms of network variance. In terms of bias, it has given either the same or better performance in comparison to the existing meth-

ods. It is also evident that there is a significant drop in the total CPU time. The lowest average drop in CPU time was recorded as 45.35% (70%:20%:10% scenario) and 44.74% (50%:40%:10% scenario) for housing data. Similarly the highest average drop in CPU time was registered as 83.5% (70%:20%:10% scenario) and 84.12% (50%:40%:10% scenario) for the Friedman regression function dataset. It may be noted that as the data size increases, the rate of increment in CPU time of data partitioning algorithms like SOMDP, FCDP and GADP increases significantly when compared to the proposed data partitioning method.

6.2 Future Work

The proposed idea is based on the principle of stratified data partitioning process. The entire dataset is divided into smaller pockets of clusters. It is understandable that data points in a smaller cluster will be statistically closer to one another than those in a bigger cluster. Although this idea has been exploited in the proposed algorithm presented in this thesis, it would be interesting to check the performance of other existing clustering algorithms like *k*-mean clustering, agglomerative nesting (Kaufman and Rousseeuw, 1990), etc. It is worth mentioning here, that any clustering algorithm can be employed to perform the stratified data partitioning. The performance will vary depending on the clustering ability.

Also, ODP methods have not been exploited to a great extent in the literature. Researchers have only implemented genetic algorithm to do this job. Other existing random search algorithms like particle swarm optimization (PSO) (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998), ant colony optimization (ACO)

(Colorni et al., 1991 and Dorigo, 1992) can be used to perform this task. It would be interesting to compare the results among the ODP class algorithms. Considering the fact that the search space can be huge (for 60 data samples, it could be in 10^{20}), it is important to overcome this problem in future.

The current study is based on a fixed ratio of training, validation and test data subsets. In this thesis, analysis has been carried out for the proposed method considering two scenarios namely, (70%:20%:10%) and (50%:40%:10%). In future, we would like to develop an optimal strategy to estimate this ratio.

From our experience and literature review, it has been observed that data preparation in ANN is an important task. Yu et al. (2007) showed that 50-70 percent of the time and the effort are spent in data preparation in complex data analysis projects. Thus, further research on overall data preparation is necessary in future. However, since data preparation also depends on the kind of data available, it is difficult to generalize this process. To get rid of redundant features, feature selection can be one area to focus on. There has been a significant amount of activity taking place for the last couple of decades in this area. But as mentioned earlier, this process is also data specific. Feature extraction techniques like principal component analysis (PCA) can be another way to get rid of redundant dimensions of input features. However, feature selection or extraction techniques have not been used in this study. The focus here was to simply investigate the effect and efficacy of data partitioning methods. In future it would be interesting to implement these techniques to the data preparation process.

While processing ultrasonic signals, raw ultrasonic data was used to prepare the input and the output data. The fact that the signals for the lower levels of cracks such as 0.1mm are sometimes overlapped by the wedge noise was ignored. This problem can be resolved by using advanced signal processing techniques like wavelet transform (WT). Further improvements can be made in the classification results by filtering the raw ultrasonic signals.

BIBLIOGRAPHY

1. Abiyev, R. H., Kaynak, O., Alshanableh, T. and Mamedov, F., *A Type-2 Neuro-Fuzzy System Based on Clustering and Gradient Techniques Applied to System Identification and Channel Equalization*, Applied Soft Computing Journal, vol. 11, issue 1, pp. 1396-1406, 2011.
2. Bezdec, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
3. Blake, C. L. and Merz, C. J., *UCI Repository of Machine Learning Databases*. Irvine, CA: Dept. Inf. Comput. Sci., Univ. California, 1998.
4. Bodyanskiy, Y. and Popov, S., *Neural Network Approach to Forecasting of Quasiperiodic Financial Time Series*, European Journal of Operational Research, vol. 175, issue 3, pp. 1357-1366, 16 Dec. 2006.
5. Bowden, G. J., Maier, H. R. and Dandy, G. C., *Optimal Division of Data for Neural Network Models in Water Resources Applications*, Water Resources Research, vol. 38, no. 2, pp.2-1-2-11, 2002.
6. Bowden, G. J., Nixon, J. B., Dandy, G. C., Maier, H. R. and Holmes, M., *Forecasting Chlorine Residuals in a Water Distribution System Using a General Regression Neural Network*, Mathematical and Computer Modelling, vol. 44, issue 5-6, pp. 469-484, 2006.
7. Bowden, G. J., Dandy, G. C. and Maier, H. R., *Input Determination for Neural Network Models in Water Resources Applications. Part I- Background and Methodology*, Journal of Hydrology, vol. 301, issue 1-4, pp.75-92, 2005.
8. Chen, L., Sugi, T., Shirakawa, S., Zou, J. and Nakamura, M., *Feature Extraction for Mental Fatigue and Relaxation States Based on Systematic Evaluation Considering Individual Difference*, IEEJ Transactions on Electronics, Information and Systems, vol. 129, issue 2, pp. 302-307+16, 2009.
9. Chong, E. K. P. and Zak, S. H., *An Introduction to Optimization*, 3rd Ed, Wiley Inter-science Series in Discrete Mathematics and Optimization, John Wiley & Sons, Inc., Hoboken, New Jersey, 2008.
10. Cochran, W. G., *Sampling Techniques*, New York: Wiley, 1977.

11. Colormi, A., Dorigo, M. and Maniezzo, V., *Distributed Optimization by Ant Colonies*, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.
12. Dede, G. and Sazli, M. H., *Speech Recognition with Artificial Neural Networks*, Digital Signal Processing: A Review Journal, vol. 20 issue 3, pp.763-768, 2010.
13. Despagne, F. and Massart, D. L., *Neural Networks in Multivariate Calibration*, Analyst, vol. 123, pp. 157-178, 1998.
14. Dhaka, V.S. and Singh, M.P., Simulating Biological Neural Network Structure in Computers with help of MATLAB for Handwriting Recognition Tasks, Asian J. Exp. Sci., vol. 21, no. 2, pp. 365-375, 2007.
15. Dorigo, M., *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italie, 1992.
16. Duran, B. S., and Odell, P. L., *Cluster Analysis*, Springer, Berlin, 1974.
17. Fallahnezhad, M., Moradi, M. H. and Zaferanlouei, S., *A Hybrid Higher Order Neural Classifier for Handling Classification Problems*, Expert Systems with Applications vol. 38, issue 1, pp. 386-393, 2011.
18. Fletcher, D. and Goss, E., *Forecasting with Neural Networks: An Application Using Bankruptcy Data*, Information & Management, vol. 24, issue 3, pp. 159-167, 1993.
19. Gil, P., Cardoso, A. and Palmal, L., *Estimating the Number of Hidden Neurons in Recurrent Neural Networks for Nonlinear System Identification*, IEEE International Symposium on Industrial Electronics, Seoul, Korea, July 5-8, 2009.
20. Glover, F., *Future Paths for Integer Programming and Links to Artificial Intelligence*, Comput. Oper. Res., vol. 13, pp. 533-549, 1986.
21. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
22. Hagan, M. T., Demuth, H. B. and Beale, M., *Neural Network Design*, PWS Publishing Company, Boston, MA, 1996.
23. Han, Z., Wang, X. and Wang, J., *Speech Recognition System Based on Visual Features and Neural Network for Persons with Speech-Impairments*, International Journal of Modelling, Identification and Control, vol. 8, issue 3, pp. 240-247, 2009.

24. He, L. Huang, G. H., Zeng, G. M. and Lu, H. W., *Wavelet-Based Multi Resolution Analysis for Data Cleaning and Its Application to Water Quality Management Systems*, Expert Systems with Applications, vol. 35, issue 3, pp. 1301-1310, 2008.
25. He, B., Oki, T., Sun, F., Komori, D., Kanae, S. and Wang, Y., *Estimating Monthly Total Nitrogen Concentration in Streams by Using Artificial Neural Network*, Journal of Environmental Management, vol. 92, issue 1, pp.172-177, 2011.
26. Kaastra and Boyd, M., *Designing a Neural Network for Forecasting Financial and Economic Time Series*, Neurocomputing, vol. 10, issue 3, pp. 215-236, 1996.
27. Kaufman, L. and Rousseeuw, P. J., *Finding Groups in Data. An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.
28. Keeratipibul, S., Phewpan, A. and Lursinsap, C., *Prediction of Coliforms and Escherichia Coli on Tomato Fruits and Lettuce Leaves after Sanitizing by Using Artificial Neural Networks*, LWT - Food Science and Technology, vol. 44, pp.130-138, 2011.
29. Kennedy, J. and Eberhart, R., *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948, 1995.
30. Kennard, R. W. and Stone, L., *Computer Aided Design of Experiments*, Technometrics, vol. 11, pp. 137-148, 1969.
31. Kim, S. H. and Chun, S. H., *Graded Forecasting Using an Array of Bipolar Predictions: Application of Probabilistic Neural Networks to A Stock Market Index*, International Journal of Forecasting, vol. 14, issue 3, pp. 323-337, 1 Sept. 1998.
32. Kirkpatrick, S., Gellat, C. D. and Vecchi, M. P., *Optimization by Simulated Annealing*, Science, vol. 220, pp. 671–680, 1983.
33. Kohonen, T., *Self Organizing Maps*, Springer Series in Information Sciences, vol. 30, Springer, Berlin, Heidelberg, New York, 2001.
34. Levin, E., *A Recurrent Neural Network - Limitations and Training*, Neural Networks vol. 3, issue 6, pp. 641-650, 1990.
35. Linde, Y., Buzo, A. and Gray, R.M., *An Algorithm for Vector Quantizer Design*, IEEE Transactions on Communications, vol.28, no.1, pp. 84-95, 1980.

36. Ludermir, T. B., A. Yamazaki and C. Zanchettin, *An Optimization Methodology for Neural Network Weights and Architectures*, IEEE Transactions on Neural Networks, vol. 17, no. 6, pp. 1452-1459, 2006.
37. MacQueen, J. B., *Some Methods for Classification and Analysis of Multivariate Observations*, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press. pp. 281–297, 1967.
38. Maier, H. R. and Dandy, G. C., *The Use of Artificial Neural Networks for the Prediction of Water Quality Parameters*, Water Resour. Res., vol. 32, issue 4, pp. 1013–1022, 1996.
39. May, R. J., Maier, H. R. and Dandy, G. C., *Data Splitting for Artificial Neural Networks using SOM-Based Stratified Sampling*, Neural Networks, vol. 23, issue 2, pp. 283-294, 2010.
40. Namia, F. and Deyhimi, F., *Prediction of Activity Coefficients at Infinite Dilution for Organic Solutes in Ionic Liquids by Artificial Neural Network*, Journal of Chemical Thermodynamics, vol. 43, issue 1, pp. 22-27, 2011.
41. Palani, S., Liong, S.Y. and Tkalich, P., *An ANN Application for Water Quality Forecasting*, Marine Pollution Bulletin, vol. 56, pp.1586-1597, 2008.
42. Park, S. S., Shin, Y. G. and Jang, D. S., *A Novel Efficient Technique for Extracting Valid Feature Information*, Expert Systems with Applications, vol. 37, issue 3, pp. 2654-2660, 2010.
43. Peng, J. X. and Li, K., *A New Jacobian Matrix for Optimal Learning of Single-Layer Neural Networks*, IEEE Transactions on Neural Networks, vol. 19, no. 1, pp.119-129, 2008.
44. Picard, J., *Enbridge disaster and pipeline safety probed*, International Business Times, Sept. 17, 2010.
45. Poddig, T. and Rehkugler, H., *A 'World' Model of Integrated Financial Markets using Artificial Neural Networks*, Neurocomputing, vol. 10, issue 3, Financial Applications, Part II, pp. 251-273, April 1996.
46. Ray, C. and Klindworth, K. K., *Neural Networks for Agrichemical Vulnerability Assessment of Rural Private Wells*, J Hydrol Eng., vol. 5, issue 2, pp. 162–171, 2000.

47. Reeves, R. R. and Taylor, S. J., *Selection of Training Data for Neural Networks by a Genetic Algorithm*, In Fifth International Conference on Parallel Problem Solving from Nature, Amsterdam, 1998.
48. Rosenblatt, F., *The perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, Psychological Review, vol. 65, pp. 386-408, 1958.
49. Rousseeuw, P. J., *Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis*, Computational and Applied Mathematics vol. 20, pp. 53–65, 1987.
50. Rugina, R. and Rinard, M., *Recursion unrolling for divide and conquer programs, in Languages and Compilers for Parallel Computing*, chapter 3, pp. 34–48. Lecture Notes in Computer Science vol. 2017, Berlin: Springer, 2001.
51. Sahoo, A. K., Zhang, Y. and Zuo, M. J., *Estimating Crack Size and Location in a Steel Plate using Ultrasonic Signals and CFBP Neural Networks*, CCECE, Niagara Falls, Canada, 2008.
52. Sahoo, G. B., Ray, C., Mehnert, E. and Keefer, D. A., *Application of Artificial Neural Networks to Assess Pesticide Contamination in Shallow Groundwater*, Science of the Total Environment, vol. 367, pp. 234–251, 2006.
53. Samanta, B., Bandopadhyay, Ganguli, S., R. and Dutta, S., *Sparse Data Division Using Data Segmentation and Kohonen Network for Neural Network and Geostatistical Ore Grade Modeling in Nome Offshore Placer Deposit*, Natural Resources Research, vol. 13, no. 3, pp.189-200, 2004.
54. Samanta, B., Bandopadhyay, S. and Ganguli, R., *Data Segmentation and Genetic Algorithms for Sparse Data Division in Nome Placer Gold Grade Estimation Using Neural Network and Geostatistics*, Exploration and Mining Geology, vol. 11, no. 1-4, pp. 69-76, 2004.
55. Scanzio, S., Cumani, S., Gemello, R., Mana, F. and Laface, P., *Parallel Implementation of Artificial Neural Network Training for Speech Recognition*, Pattern Recognition Letters, vol. 31, issue 11, pp. 1302-1309, 2010.
56. Schuster, M. and Paliwal, K. K., *Bidirectional Recurrent Neural Networks*, IEEE Transactions on Signal Processing, vol. 45, issue 11, pp. 2673–2681, 1997.
57. Shahin, M. A., Maier, H. R. and Jaksa, M. B., *Data Division for Developing Neural Networks Applied to Geotechnical Engineering*, Journal of Computing in Civil Engineering, vol. 18, no. 2, pp.105-114, 2004.

58. Shahin, M. A., Jaksa, M. B. and Maier, H. R., *Artificial Neural Network Applications in Geotechnical Engineering*, Australian Geomech., vol. 36, issue 1, pp.49–62, 2001.
59. Shahin, M. A., H. R. Maier and M. B. Jaksa, *Predicting Settlement of Shallow Foundations Using Neural Networks*, J. Geotech. Geoenviron. Eng., vol. 128, issue 9, pp.785–793, 2002.
60. Shi, Y. and Eberhart, R. C., *A Modified Particle Swarm Optimizer*, Proceedings of IEEE International Conference on Evolutionary Computation. pp. 69–73, 1998.
61. Simani, S. and Fantuzzi, C., *Fault Diagnosis in Power Plant using Neural Networks*, Information Sciences, vol. 127, issues 3-4, pp. 125-136, Aug. 2000.
62. Sinha, R. K., Aggarwal, Y. and Das, B. N., *Backpropagation Artificial Neural Network Detects Changes in Electro-Encephalogram Power Spectra of Syncopic Patients*, Journal of Medical Systems, vol. 31, issue 1, pp. 63-68, 2007.
63. Smith, M., *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold, New York, 1993.
64. Snee, R. D., Validation of regression models: Methods and examples, Technometrics, vol. 19, issue 4, pp. 415-428, 1977.
65. Sprevak, D., Azuaje, F. and Wang, H., *A Non-Random Data Sampling Method for Classification Model Assessment*, In 17th international conference on pattern recognition, vol. 3, pp. 406-409, 2004.
66. Tarassenko, L., *A Guide to Neural Computing Applications*, Neural Computing Applications Forum, 1998.
67. Teoh, E. J., Tan, K. C. and Xiang, C., *Estimating the Number of Hidden Neurons in a Feedforward Network Using the Singular Value Decomposition*, IEEE Transactions on Neural Networks, vol. 17, no. 6, pp.1623-1629, 2006.
68. Tokar, S. and Johnson, P. A., *Rainfall-Runoff Modeling Using Artificial Neural Networks*, J. Hydrol. Eng., vol. 4, issue 3, pp. 232– 239, 1999.
69. Tong, F. and Liu, X., *Samples Selection for Artificial Neural Network Training in Preliminary Structural Design*, Tsinghua Science and Technology, vol.10, no. 2, pp. 233-239, 2005.

70. Tokar, A. S. and Johnson, P. A., *Rainfall-Runoff Modeling Using Artificial Neural Networks*, J. Hydrologic Eng., vol. 4, issue 3, pp. 232–239, 1999.
71. Trenn, S., *Multilayer Perceptrons: Approximation Order and Necessary Number of Hidden Units*, IEEE Transactions on Neural Networks, vol. 19, no. 5, pp. 836-844, 2008.
72. Twomey, J. M. and Smith, A. E., *Bias and Variance of Validation Methods for Function Approximation Neural Networks under Conditions of Sparse Data*, Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 28, no. 3, pp. 417-430, 1998.
73. Wan, S. and Banta, L. E., *Parameter Incremental Learning Algorithm for Neural Networks*, IEEE Transactions on Neural Networks, vol. 17, no. 6, pp.1424-1438, 2006.
74. Warlaumont, S., Oller, D. K., Buder, E. H., Dale, R. and Kozma, R., *Data-Driven Automated Acoustic Analysis of Human Infant Vocalizations using Neural Network Tools*, Journal of the Acoustical Society of America, vol. 127, issue 4, pp. 2563-2577, 2010.
75. Weerasinghe, M., Gomm, J. B. and Williams, D., *Neural Networks for Fault Diagnosis of a Nuclear Fuel Processing Plant at Different Operating Points*, Control Engineering Practice, vol. 6, issue 2, pp. 281-289, 1 Feb. 1998.
76. Widrow, B. and Hoff, M. E., *Adaptive Switching Circuits*, 1960 IRE WESCON Convention Record, New York: IRE Part 4, pp. 96-104, 1960.
77. Wu, J. D., Wang, Y. H., Chiang, P. H. and Bai, M. R., *A Study of Fault Diagnosis in a Scooter using Adaptive Order Tracking Technique and Neural Network*, Expert Systems with Applications, vol. 36, issue 1, pp. 49-56, Jan. 2009.
78. Yen, G. G. and Lin, K. C., *Wavelet Packet Feature Extraction for Vibration Monitoring*, IEEE Transactions on Industrial Electronics, vol.47, no.3, pp.650-667, 2000.
79. Yu, L., Wang, S. Y. and Lai, K. K., *Foreign-Exchange-Rate Forecasting with Artificial Neural Networks*, New York: Springer, 2007.
80. Zhang, J., *Improved On-Line Process Fault Diagnosis through Information Fusion in Multiple Neural Networks*, Computers & Chemical Engineering, vol. 30, issue 3, pp. 558-571, 15 Jan. 2006.
81. Zhang, Y., Ng, C. C. and Sahoo, A., Hanxin Chen, and Ming J. Zuo, *Inspection of EMD Slots on Steel Blocks Using OmniScan*, Technical Re-

port, Department of Mechanical Engineering, University of Alberta, Edmonton, Alberta, T6G 2G8, June 15, 2007.