4-2018

ARTIFICIAL INTELLIGENCE APPROACH FOR CLASSROOM SCHEDULING

Farah M. T. Aiash

UAEU

جامعة الإمارات العربيـة المتحدة
United Arab Emirates University

United Arab Emirates University

College of Engineering

Department of Mechanical Engineering

ARTIFICIAL INTELLIGENCE APPROACH FOR CLASSROOM SCHEDULING

Farah M. T. Aiash

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science in Mechanical Engineering

Under the Supervision of Dr. Basem Fayez Yousef

April 2018

Declaration of Original Work

I, Farah M. T. Aiash, the undersigned, a graduate student at the United Arab Emirates University (UAEU), and the author of this thesis entitled "*Artificial Intelligence Approach for Classroom Scheduling"*, hereby, solemnly declare that this thesis is my own original research work that has been done and prepared by me under the supervision of Dr. Basem Fayez Yousef, in the College of Engineering at UAEU. This work has not previously been presented or published or formed the basis for the award of any academic degree, diploma or a similar title at this or any other university. Any materials borrowed from other sources (whether published or unpublished) and relied upon or included in my thesis have been properly cited and acknowledged in accordance with appropriate academic conventions. I further declare that there is no potential conflict of interest with respect to the research, data collection, authorship, presentation and/or publication of this thesis.

Student's Signature: ____________________ Date: 20/5/2018

Approval of the Master Thesis

This Master Thesis is approved by the following Examining Committee Members:

1) Advisor: Dr. Basem Fayez Yousef
   Title: Associate Professor
   Department of Mechanical Engineering
   College of Engineering
   Signature ____________ Date 30/4/2018

2) Co-Advisor (Committee Chair): Dr.Khalifa H. Harib
   Title: Associate Professor
   Department of Mechanical Engineering
   College of Engineering
   Signature ____________ Date 30/4/2018

3) Member: Dr.Rafic Ajaj
   Title: Assistant Professor
   Department of Mechanical Engineering
   College of Engineering
   Signature ____________ Date 30/4/2018

4) Member (External Examiner): Professor Saeed Zolfaghari /Dr. Tariq Darabse
   Title: Professor
   Department of Mechanical and Industrial Engineering
   Institution: Ryerson University
   Signature ____________ Date 30/4/2018

This Master Thesis is accepted by:

Dean of the College of Engineering. Professor Sabah AlKass

Signature ______________ Date 17/5/2018

for Dean of the College of Graduate Studies: Professor Nagi T. Wakim

Signature Ali Hassan Date 21/5/2018

Copy 8 of 9

Advisory Committee

1) Advisor: Dr. Basem Fayez Yousef

Title: Associate Professor

Department of Mechanical Engineering

College of Engineering

2) Co-advisor: Dr. Khalifa Hamad Harib

Title: Associate Professor

Department of Mechanical Engineering

College of Engineering

Abstract

Companies, factories, and academic institutes often rely on planning and controlling scheduling of production lines or classrooms to ensure efficient utilization of resources. Task scheduling is a complex nonlinear process, due to numerous constraints, parameters and frequent, sudden changes in the requirements. The aim of this project is to explore the utilization of artificial intelligent neural networks in the preparation of classroom scheduling by utilizing their adaptive attributes and learning ability to establish a procedure for classroom timetable preparation. A set of input vectors comprising five constraints are introduced to a Self-Organizing Feature Map (SOM) neural network for classroom sections classification and separation, using some cluster centers equal to the available rooms. The SOM demonstrated strong capability in clustering the sections into groups comprising courses with conflicts based on the defined constraints, hence identifying classes to be sequentially scheduled in one room. A second stage SOM is used to further split oversized clusters. Moreover, to fit newly created classrooms into the SOM generated timetable, the output from SOM is used to train a Feedforward Back Propagation (FFBP) neural network to extract the implicit course-classroom mapping as formulated by the SOM. The trained FFBP is used to accommodate the new courses without the need to re-cluster with SOM. The trained FFBP managed to prepare a conflict-free schedule successfully. The outputs of the integrated neural networks show that the proposed model can create an initial guess of a valid classroom schedule. It is envisaged that the procedure can be extended and implemented in fields other than academia such as factories, healthcare, and transportations.

Keywords: Classroom scheduling, artificial intelligent neural networks, Self-Organizing Feature Map neural network, Feedforward Back Propagation neural network.

Title and Abstract (in Arabic)

استخدام الذكاء الاصطناعي في جدولة الصفوف الدراسية

الملخص

تعتمدُ الشركات والمصانع والمعاهد الأكاديمية على تخطيطِ ومراقبة جدولة خطوط الإنتاج أو الفصول الدراسية؛ وذلك لضمان الاستخدام الأمثل للموارد المتاحة. جدولة المهام هي عملية معقدة غير خطية (Non-linear) رياضياً؛ يعود سبب تعقيدها إلى الكثير من القيود والعوامل والمتغيرات المفاجئة والمتكررة في المتطلبات. الهدف من هذا المشروع الكشف عن استخدام الشبكات العصبية الذكية الاصطناعية (Artificial Intelligent Neural Networks) في إعداد جدولة الفصول الدراسية من خلال الاستفادة من سمات التكيف والقدرة على التعلم واستخدامها لإعداد جدول الفصل الدراسي. يتم إدخال مجموعة من ناقلات الإدخال (Input vector) والتي بدورها تتألف من خمسة قيود إلى شبكة الخريطة العصبية ذاتية التنظيم (Self-Organizing Map) لتصنيف المواد الدراسية وفصلها إلى مجموعات، ويتم ذلك باستخدام عدد من المراكز العنقودية مساوية لعدد الفصول الدراسية المتاحة. أظهرت شبكة الخريطة العصبية ذاتية التنظيم (SOM) قدرتها العالية في تصنيف المساقات إلى مجموعات تشمل المساقات المتعارضة مع الأخذ بعين الاعتبار المحددات المذكورة. وبالتالي تحديد المساقات بطريقةٍ جدولةٍ متسلسلةٍ في الفصل الدراسي الواحد. وتستخدم شبكة الخريطة العصبية ذاتية التنظيم (SOM) في المزيد من التصنيف في حال وجود تجمعات كبيرة من المساقات المتكدسة. وعلاوة على ذلك، يتم اضافة مساقات جديدة -غير التي تم إنشاؤها-. وبناء على المعلومات المعدة في شبكة الخريطة العصبية ذاتية التنظيم (SOM) من قبل؛ نعتمد على مخرجات (Output vector) شبكة الخريطة العصبية ذاتية التنظيم (SOM) التي بدورها يتم استخدامها كمدخلات لتدريب الشبكة العصبية ذات التغذية الأمامية والانتشار الارتدادي (Feedforward Back Propagation (FFBP) neural network)؛ لاستخراج الفصل الدراسي المناسب للمساق الجديد كما صاغتها شبكة الخريطة العصبية ذاتية التنظيم (SOM). ويتم استخدام الشبكة العصبية ذات التغذية الأمامية والانتشار الارتدادي (FFBP) المدربة مسبقاً لاستيعاب المساقات الجديدة دون الحاجة إلى إعادة التصنيف باستخدام شبكة الخريطة العصبية ذاتية التنظيم (SOM). و قد تمكنت الشبكة العصبية ذات التغذية الأمامية والانتشار الارتدادي (FFBP) المدربة من إعداد جدول للفصول الدراسية خال من التعارضات بنجاح. تظهر مخرجات الشبكات العصبية المتكاملة أن النموذج المقترح

يمكن أن يخمن بشكل فعالٍ أولي   جدولة الفصول الدراسية. ومن المتصور أن يمتد هذا الإجراء وينفذ في ميادين اخرى غير الأوساط الأكاديمية مثل المصانع والرعاية الصحية والنقل.

مفاهيم البحث الرئيسية: جدولة الفصول الدراسية، الشبكات العصبية الذكية الاصطناعية، شبكة الخريطة العصبية ذاتية التنظيم(SOM)، الشبكة العصبية ذات التغذية الأمامية والانتشار الارتدادي (FFBP).

Acknowledgements

I am most grateful to my supervisor Dr. Basem Yousef for all his patience, guidance and enthusiasm. The door to Dr. Basem’s office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my work, but steered me in the right the direction whenever he thought I needed it. It was an honor for me to work under his exemplary supervision. I am also very grateful to Dr. Khalifa Harib for his support and encouragement. I also would like to acknowledge Dr. Khalid Al Arebi for his contributions in the early stages of this research.

Many thanks go to Dr. Tariq Darabseh, master program coordinator of the Mechanical department, for his guidance and support. I would like to thank my committee for their guidance and assistance throughout my preparation of this thesis. Also, my special thanks are extended to the Library Research Desk for providing me with the relevant reference material.

Finally, I must express my very profound gratitude to my parents, my spouse and my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Dedication

This thesis is dedicated to
my husband Ahmed
who’s love and encouragement made this thesis possible
and my parents and family
who have always supported me in everything I’ve attempted

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
SOM	Self-Organizing Map
FFBB	Feed Forward Back-propagation
NN	Neural Network
MLP	Multi-Layer Perceptron
GUI	Graphical User Interface
LMS	Least Mean Squares

List of Symbols

i	Synapse
x_i	Signal input
j	Neuron
w_{ji}	Weight
y_j	Output signal, actual output
$\emptyset(x), f$	Activation function
$x(n)$	Input vector
$w(n)$	Weight vector
x, y	Cartesian-coordinates
$i\ (\vec{x})$	Closest weight vector, (distance)
l	The total number of output neurons in the network
$d_{j,i}$	The lateral distance
(n)	Iterations
σ	The width of the Gaussian function, also it is called the neighborhood radius
σ_o	The initial σ
Ţ_1	The time constant
$h_{j,i(\vec{x})}(n)$	Neighborhood function
r_j	The position vector of the excited neuron j
r_i	The position vector of the winning neuron i
$\Delta\overrightarrow{w_j}$	Updated weight
η	The learning rate parameter

τ_2	Another time constant
$g\left(y_j\right)\overrightarrow{w_j}$	The forgetting term
E (n)	The error energy at iteration n
$e_j(n)$	The error signal
$d_j\ (n)$	The desired response or target output for neuron j at iteration n
$E_{avg}(n)$	The average square energy
$v_j(n)$	Induced local field of a neuron j is the output of the summation unit
∂	Partial differentiating
$\acute{\emptyset}\ \left(v_j(n)\right)$	Differentiated activation function with respect to $v_j(n)$
$\delta_j(n)$	Local gradient
k	new index to avoid any confusion between neuron *j*
o_j	The function signal at the output neuron *j*
α	Momentum constant

Chapter 1: Introduction

1.1 Overview

Scheduling problems have been the subject of research for decades. According to (Wren, 1995), scheduling is the constraint of resources to objects, being placed in space-time in such a way as to minimize the total cost of a set of the resources used. Scheduling activity is considered a fundamental and frequent action in many places such as hospitals, transportation, and academic institutions. This thesis is concerned with classroom scheduling using an Artificial Intelligence (AI) approach. The adaptability of this AI approach will open the doors to use it, not only in academic institutions, but also in hospitals, transportation, factories and other places or fields where efficient utilization of resources is needed.

1.2 Problem Statement and Purpose

Class scheduling for academic institutes is a fundamental educational management activity. With the number of students increasing, academic programs and other requirements upsurge the complexity of designing a conflict-free timetable. A class scheduling problem inherits the intricacy of both resources allocation and personal preferences. The manual timetabling process for larger academic organizations can be described as time-consuming, tedious and oftentimes challenging. This appears to make scheduling timetables an ideal candidate for the application of information technology. The aim of this project is to design an artificial neural network that can help in solving a complex university class scheduling problem. The proposed system uses a neural network-approach which has the competency to adapt to unforeseen scenarios and problems, thus creating a conflict-free schedule for students while using optimal classroom space and associated teacher resources.

1.3 Intended Outcomes and Deliverables

The outcomes that would be achieved in this project are numerous. Designing an integrated model of neural networks will lead to the generation of a conflict-free classroom timetable, but most importantly this model introduces a new approach in creating classroom scheduling. Due to the use of artificial intelligence, this approach has the potential to generate conflict free schedules in complex scenarios and institutions with greater ease than exists within previous models (Teoh & Wibowo, 2013) . An additional outcome of this project is the ability for it to be applied, not only to academic institutions, but also in hospitals, transportation, factories and other places or fields where efficient utilization of resources is needed.

An unanticipated outcome comes from the experience of implementing neural networks to produce a functioning model; it is not like any other experience due to the fact that it is professionally following a world a wide trend these days. Moreover, gaining skills that contribute to realizing the machining learning for any type of knowledge base data that is encountered in professional careers is invaluable.

In addition, this project meets the vision of United Arab Emirates in using artificial intelligence in its industry and sectors. Also, this project opens the doors widely for further questioning and research into ways the neural network can be used in different applications once we think of it in un-traditional way.

1.4 Relevant Literature

Class scheduling for an academic institution has become time-consuming, redundant, and tedious. For example, classes have been double booked with no instructors, students have been looked over, the seemingly available time slots have proven to be unavailable or not able to support the entire class period. All these make the process of achieving an efficient class scheduling frustrating and very difficult.

Many process constraints are encountered while preparing a working timetable those include scheduling classes, teachers, and rooms into a fixed number of conflict-free time slots. Accordingly, no teacher, class or room is used more than once during a specific time slot.

Traditionally, staff who scheduled classes utilized a trial-and-error approach which was to manually create a conflict-free timetable while optimizing the use of rooms and associated teaching resources. However, when put in practice, this approach has proven inefficient or unsuccessful.

In addition, increasing the number of students, different programs of study, and teachers will increase the complexity of the process of class schedule. These challenges make using an information technology program a good candidate to solve these ever-present scheduling problems.

Many research projects related to automated timetabling were reported in the literature (Gotlieb, 1963), where researchers have developed different approaches to solving the class schedule problem (Carrasco & Mizrach, 1986). At an early stage, the operation research optimization techniques were used extensively in solving the timetable problems (Aloul, 2007).

Some other papers have classified the basic scheduling approaches into two different approaches; the action-driven approach and the strategy-driven approach. The action-driven approach consists of heuristic algorithms and analytic methods. It is considered as commonly used approach for education institutions (Liebowitz, 1998). The strategy-driven approach aims to construct a flexible and easily adaptable timetabling system. This system takes into account the teaching staff preferences and un-availabilities according to information as obtained directly from staff. This approach includes the optimization techniques (Dimopoulou, 2004)

Recently, researchers have increasingly focused on the science of artificial intelligence to solve issues associated with scheduling. Since using the simulated annealing, tabu search, constraint satisfaction genetic algorithms and neural networks (Schaerf, 1999), which are techniques in artificial intelligence, there has been a significant improvement in the performance of solving timetable problems compared to the traditional operation research techniques and optimization. These AI methods attracted researchers due to their flexibility and adaptability for different scenarios (Abramson, 1992).

Examples of those methods are “Genetic algorithms (GA) which mimic the process of natural selection and can be used as a technique for solving complex optimization problems that have very large search spaces. Although the GAs can solve a complex timetable problem and are considered quite powerful in finding the global minimum from an enormous search space, their convergence is very much dependent on the initial solution (Azimi, 2005). This is due to the ambiguity in deciding the fitness function of the GA. Many approaches which seek to find optimal solutions to constraint-satisfaction problems by genetic algorithms have been suggested, but the majority of these methods are problem-dependent and consequently difficult to apply to the complexity posed by real-world situations (Deris, 1999).

Further research was conducted using genetic algorithms to solve the timetable problems which considered the flexibility preferences. Accordingly, a new method was created by combining an Ant Colony Optimization (ACO) with the Genetic Algorithm Operators method. Hence the Ant Colony Optimization (ACO) is a population-based metaheuristic that seeks to solve difficult combinatorial optimization problems (Birattari, 2011). This combination resulted in flexible timetables (Mahmud, 2014), however, the genetic algorithm could stop during some occasions which depend on the search area space. For example; in extremes situation where only one solution exists, the genetic algorithm will most likely fail since it does not work based on an event or action language (Ansari, 2014).

Another artificial intelligence based approach is knowledge-based. This system which is called "Assistant for Class Scheduling" uses the knowledge of an expert human scheduler to generate a class schedule. The system has a control strategy where it can prioritize the courses according to the preferred time of the corresponding teacher and the size of the classroom. Accordingly, courses with higher priority will be chosen first for scheduling. However, this approach has some limitations. For instance, it can adapt only for courses that are distributed with an even number of hours per week, courses with odd numbers of hours allocated per week have to be scheduled manually (Hwang, 1989). Also, courses requiring a special classroom are not considered in this method. In particular, this approach lacks the ability to adapt to scenarios that are variant of the norm (Qu, 2006). Also, in Hong Kong, an institution created an intelligent timetabling using a knowledge-based system built on a microchip. The knowledge, strategies, and heuristics of a small, centralized group of schedulers were modeled and subsequently represented in a readily available expert system shell which runs on a standard IBM-type microcomputer (Martinsons & Kong, 1993).

Furthermore, the cultural algorithms (Reynolds, 1994) are a class of evolutionary algorithms that use domain knowledge extracted during the evolutionary process to improve the performance of the search engine adopted. It is a new approach that is showing to be effective to solve the timetabling problem. Another promising approach is the Population-based algorithms which are better at exploring a search space than local search algorithms. In other words, it has potential in optimizing the solution of the timetabling problem (Abuhamdah, Ayob, & Kendall, 2013).

In addition, Simulated Annealing algorithm has a potential to create a classroom timetable (Teoh & Wibowo, 2013). The simulated annealing name comes from the principles of metallurgy, which boiled and cooled metals to achieve a stable crystal lattice structure with minimal energy state. The algorithm begins by generating an initial random solution. After that, an adjacent solution is generated and these two solutions are evaluated by an objective function (Gonzalez, 2007). As stated by Basir (2013) the use of simulated annealing will give an optimum solution to the problem. This makes simulated annealing an attractive option for the problem of optimization, (Basir, Ismail, & Norwawi, 2013).

Another model that can be used to solve problems associated with curriculum-based course timetabling that was introduced is the Adaptive Tabu Search algorithm (Zhipeng, 2010). Tabu search is a metaheuristic search method employing local search methods used for mathematical optimization (Glover, 2018). The proposed algorithm follows a general framework composed of three phases: initialization, intensification and diversification. The initialization phase creates a feasible initial timetable using a fast heuristic. To reduce the number of soft constraint while still satisfying the hard constraints, an adaptively combined intensification and diversification is used. The proposed hybrid system showed that it has the potential to solve a course timetabling problem. Because the basic ideas are quite general, it would also be applicable to other similar problems (Zhipeng, 2010).

Another technique used to suggest a solution for class scheduling is the graph partitioning algorithm combined with simulated annealing. A graph partitioning algorithm is a mathematical algorithm that is defined by data represented in the form of a graph G = (V,E), with V vertices and E edges. The graph partitioning algorithm was used to represent the relation between the constraints and the time-slots which can be represented by an edge-weighted graph. The simulated annealing was used as a "noise term" to update spin configuration in the graph partitioning algorithm (Yu, 1990).

Furthermore, neural networks models like: Interactive Activation and Competition, Potts Neural Network, and Modified Hopfield Neural Network were previously introduced to solve scheduling issues. The first model uses a hybrid form of neural network to create the Interactive Activation and Competition networks. The structure of the model organizes classes sequentially from the network in the region with the largest number of restrictions first. At the same time, the network configures the parallel combination of resources most appropriate for the class in question, under simultaneous interaction of complex restrictions. The restrictions then go through adjustments in synaptic weights before the next class is selected for scaling. It results with significantly slower network growth (linear) according to the size of the problem and is flexible to organize restrictions more realistically. The second network is the Potts Neural Network, which is a derivation of the Hopfield Neural Network discussed below. In the Potts network, a neuron multistate is used (in place of the usual two neuron stages), as well as a factorization, which provides a substantial reduction in the number of neurons (Carrasco & Pato, 2001). The third model is the Hopfield network with modifications. The main advantage of this tool lies in its potential for fast computational power when implemented in hardware, and also the parallel nature of the ANNs (Smith, Abramson, & Duke, 2003) .The results achieved using the modified Hopfield Neural Network proved it comparable to the best technical heuristics. Thus, the network was shown capable of producing solutions to complex problems of time allocation. One advantage of this method is its speed (Taborda, 2004).

In a comparative study of simulated annealing, tabu search with local search and genetic algorithms in solving the school timetabling problems for two Italian high schools conducted by Colorni (1998), found that Tabu search produced the best results followed by genetic algorithms and simulated annealing (Colorni, 1998).

Another computational study was done by Smith et al. (2003) using Hopfield neural network to solve the school timetabling problem. It was used for nine high schools. The performance of the Hopfield neural network on this data set is compared to simulated annealing and tabu search. The neural network performed better than the other methods, followed by simulated annealing (Smith, Abramson, & Duke, 2003). The comparison above shows that neural networks are more useful to solve the problems associated with complex timetabling (Pillay, 2010) .

It is clear that a fair amount of research has been conducted in the use of artificial intelligence to solve the problem of scheduling; however, few other studies were found to use neural networks to solve the issue of class scheduling. The (Smith, Abramson, & Duke, 2003) using the Hopfield network. The research in this thesis proposes an alternative network application solution to this issue through the use of

SOM and FFBP. Additional investigation into new methods should continue to be conducted, as they may contribute further to the scheduling field.

1.5 Report Structure

This report is well organized to explain all concepts clearly by moving from the implementation of the neural networks, to the analysis, and finally to the results of the designed model. Chapter 1 focuses on problem statements, purposes, intended outcomes and deliverables and the background literature review. In Chapter 2 an overview of neural network is highlighted and the mathematical and architectural models of SOM NN and FFBP NN are discussed in detail. Creation of data sets, as well as methodology to solve the scheduling problems are proposed and discussed in Chapter 3. Chapter 4 illustrates the results and the discussion for each phase of this research. Finally, this thesis concludes in Chapter 5 with summarization of the work done; here it mentions the final results and future plans for further researches.

Chapter 2: Introduction to ANN

The artificial neural network (ANN) is the proposed approach to be used in this thesis to find a new, alternative solution to the challenge posed by classroom scheduling by examining and utilizing the adaptation feature in ANN. In this chapter, the basic concept of ANN is introduced with its architecture. Then, the mathematical details of the selected ANN types that will be used in this thesis are explained.

2.1 Basic Concept of ANN

The artificial neural network theory was inspired by the structure of the human brain. The human brain is a highly complex and non-linear system. It can process a vast amount of information simultaneously. Hence, the human brain uses parallel interconnection neurons in processing the data, allowing the neurons to interact in parallel through multiple layers of neurons in the brain. A neuron sends output and receives input. Each neuron can receive values from all neurons in the previous layer, and it can send values to all neurons in the next layer. The continuity of sending and receiving values between neurons is called learning and memorizing. As a result, the brain will be able to make the proper decision.

The ability of decision making in the brain is gained from memorizing and learning from previous cases that are similar to the situation the brain is trying to make a decision for. Scientists tried to mimic the brain’s neurons architecture to develop more effective and efficient engineering systems. They created the Artificial Neuron Networks which are commonly known as “neural networks.” An artificial neural network is made up of layers of artificial neurons or processing elements. The processing element has the natural tendency to store information, also known as experimental knowledge, and make it useful when we need it.

Neural networks resemble the brain via two aspects:

1. Gaining knowledge: knowledge is built up in the network from its environment through the learning process.
2. The functioning of connectivity: interneuron connection strengths are the synaptic weights which are used to store the learned or developed knowledge.

In brief, the learning process is the process where the synaptic weights are modified to attain the desired design objective.

2.2 Benefits of Neural Network

The aptitude to derive meaningful results from massive or distorted data is a remarkable feature in ANN. It gives reasonable outputs from inputs not encountered during the training (learning), which is referred to as generalization. As a result, the neural network can find good approximate solutions to complex large-scale problems. Other advantages are:

Nonlinearity: Neural networks are made up of an interconnection of nonlinear neurons which makes an ANN able to approximate any nonlinear continuous function to the anticipated solution. This property is highly essential mainly if the underlying physical mechanism responsible for generating the input signal is inherently nonlinear.

Input-Output Mapping: Supervised learning, or learning with a teacher, is a prevailing paradigm in neural networks. It is a system where the input and the desired output data are provided. When a set of paired data is trained to generate consistent output for the response to new data, this is called a supervised learning algorithm. Hence, it involves the modification of the synaptic weights of the trained network.

Adaptivity: Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment. In particular, a neural network trained in a specific context can be retrained easily to handle changes in the conditions of the operating environment.

2.3 Models of a Neuron

A neuron, or the processing unit, is the fundamental element to the operation of a neural network. It consists of three main parts as shown in Figure 2.1:

Synapses or connection links: each connector or synapse is characterized by weight or strength; the synapse i connects the signal input x_i with the neuron j. The relationship between the input signal x_i and the neuron j is presented by multiplication between the weight w_{ji} and the input signal x_i .

Summing junction: Adding all input signals weighted by the respected synaptic strength.

Activation function: Limiting the amplitude to determine a neuron’s output in a neural network. It maps the resulting values between 0 to 1 or -1 to 1 etc. (depending upon the function). The most common activation functions are listed in the Table 1.

Bias: is similar to the constant b of a linear function y = ax + b. It allows one to move the line up and down to better fit the prediction with the data. Two different kinds of parameters can be adjusted during the training of an ANN: the weights and the value in the activation functions. Due to the impracticality of adjusting both parameters, a bias neuron is invented. The bias neuron lies in one layer, and is connected to all the neurons in the next layer, but none in the previous layer and it always emits 1. Since the bias neuron emits 1 the weights, connected to the bias neuron, are added directly to the combined sum of the other weights Eq. 2.1 (Rojas, 1996).

$$v_i = \sum w_{ji}\, x_m \qquad Eq.(\,2.1)$$

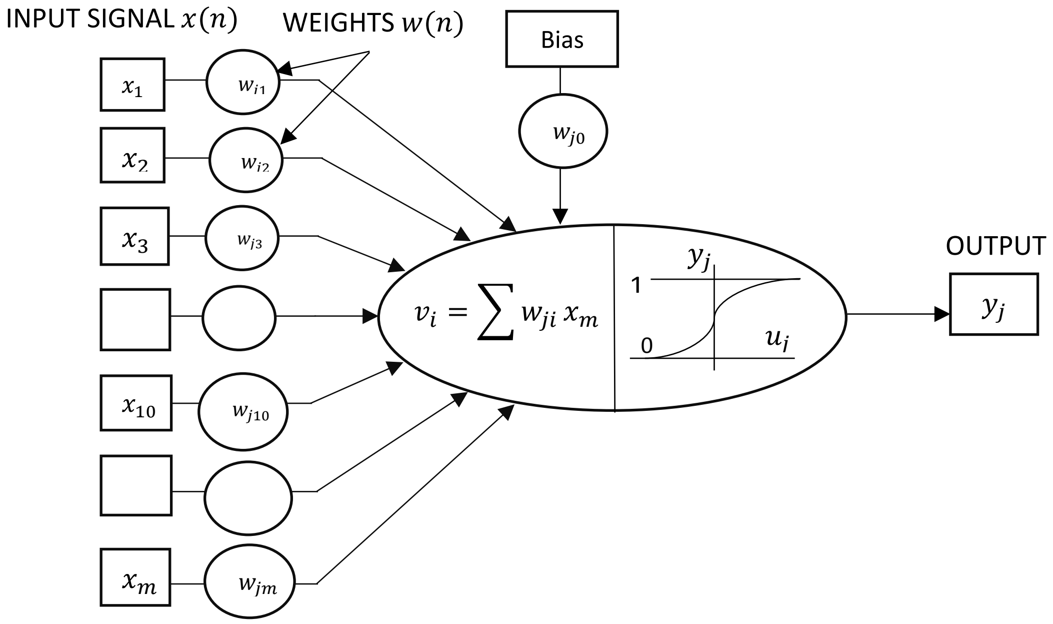

Figure 2.1: The basic structure of the neuron

Table 1: Different types of activation function

Function name	Formula	Values range
Exponential	$\emptyset(x) = e^{-ax}$	$(0, \infty)$
Sigmoid	$\emptyset(x) = \frac{1}{1 + e^{-ax}}$	(0,1)
Hyperbolic Tangent	$\emptyset(x) = \frac{2}{1 + e^{-2x}} - 1$	$(-1,1)$
Step	$\emptyset(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$	[0,1]

Thus, the operation performed by neuron j can be mathematically expressed as

$$y_j = f\left(w(n)^T x(n)\right) \qquad Eq.(\,2.2)$$

Where f is the activation (transfer) function, y_j is the output of neuron j, the superscript T represents the transpose of $w(n)$ which is the interconnection weight vector and $x(n)$ is the input signal vector for iteration n.

2.4 Self-Organizing Feature Map (SOM)

Self-organizing feature map neural-network is a type of artificial intelligence that is trained using unsupervised-learning to produce lower dimensional clustered regions. The self-organizing feature map was initially proposed by Rosenblatt in 1958 (Lek & Guégan, 1999). The idea of the self-organizing map is inspired from human brain: “The brain is organized in many places in such a way that different sensory inputs are represented by topologically ordered computational map” (Haykin, 2009).

The self-organizing map network is based on competitive learning systems. The network output neurons compete among themselves to be activated or fired. Hence only one neuron per group will be the winning neuron. The way of persuading a winner output neuron among a group of outputs is to use lateral inhibitory connections. The lateral inhibitory connection is when the neuron dominates the field and inhabits neighboring neurons (Sayers, 1991).

The neurons of the self- organizing map are sited in a frame called lattice. The lattice can be one, two or higher dimensional maps. The neurons become selectively adjusted to various input patterns. Accordingly, the location of winning neurons become ordered in an expressive coordinate system for different input features, which are created over the lattice (Haykin, 2009).

2.4.1 Self-Organizing Feature Map Structure

The structure of the self-organizing feature map can be presented and understood with the use of an illustration such as in the Figure 2.2 shows a tiny Kohonen network of 3x3 output nodes/layer connected to two input nodes/layer. Each output node has a specific topological position which represented as unique x, y coordinates in the 2-D lattice (output layer).

The mechanism behind Kohonen’s network is straightforward; when an input pattern embodies to the network, the response of each neuron is measured, and the one which produced the maximum response, as well as the adjacent neurons, are modified in such a way to generate a better response to that input pattern. After many iterations, the system should ideally reach a state where no more significant change in the neuron location appears.

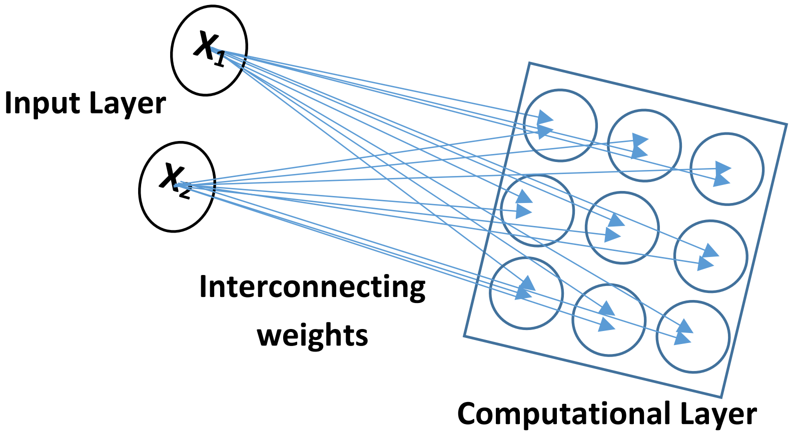

Figure 2.2: Self-organizing feature map

2.4.2 Self-Organizing Feature Map Algorithm

The self-organizing feature map process can be divided into four main stages:

Initialization: where all the connection weights are initialized with random values.

Competition: where each input pattern and it's corresponding weights compute their respective values of a discriminant function[1]. As a result, the neuron with the largest discriminant function is the winner.

In the first step, let the input space dimension to be m then the input pattern can be written as $\vec{x} = [x_1\ x_2\ \ldots.x_m]^T$. And the connection weight between the input layer and the computational layer is $\overrightarrow{w_j} = [w_{j1}\ w_{j2}\ \ldots.w_{jm}]^T; j = 1,2, \ldots.., l$,where l is the total number of output neurons in the network. The next step is to find the best match between $\vec{x}$ and $\overrightarrow{w_j}$. To find the best match we need to compute $\overrightarrow{w_j}^T\ \overrightarrow{x}\ for\ j = 1,2, \ldots..., l$ and select the largest value; hence, the maximum value of $\overrightarrow{w_j}^T\ \vec{x}$ is nothing but the minimum value of the Euclidian distance between $\overrightarrow{x}\ and\ \overrightarrow{w_j}$. By using the index $i\ (\vec{x})$, the formula below will give the value of the $winning\ neuron$. Finally, acknowledge that the corresponding weights vector to $i\ (\vec{x})$ is the closest weight vector.

$$i\ (\vec{x}) = \arg min_j \left\|\vec{x} - \overrightarrow{w_j}\right\| \qquad Eq. (2.3)$$

In brief, a continuous input space of activation pattern is mapped into a discrete output space by process of competition.

Cooperation: where the winning neuron can spot the spatial location of a topological neighborhood of excited neurons, or more specifically, winning neurons locate the center of a topological neighborhood of an excited/ cooperated neuron.

In the beginning, assume that i is the winning neuron and $h_{j,i}$ is the topological neighborhood centered around i and encompassing neuron j. Naturally, the topological

[1] A function of several variates used to assign items into one of two or more groups. The function for a particular set of items is obtained from measurements of the variates of items which belong to a known group

neighborhood function should decrease with the $d_{j,i}$ which is the lateral distance between the winning neuron i and the excited/ neighbors' neurons j.

Before implementing the topological neighborhood function it should satisfy three properties:

- Symmetric about $d_{j,i} = 0$
- Monotonically decaying function with distance $d_{j,i}$
- Decaying to zero at $d_{j,i}$ → ∞.

The typical function which may fulfill the mentioned properties is the Gaussian function, accordingly, $h_{j,i}$ can be expressed as the following:

$$h_{j,i(\vec{x})} = exp\left(-\frac{{d_{j,i}}^2}{2\,\sigma^2}\right) \qquad Eq. (2.4)$$

Where σ is the width of the Gaussian function, Figure 2.3, also it is called the neighborhood radius (Guthikonda, 2005). Note that the Gaussian function does not depend on the winner neuron's location; hence, it is translation invariant.

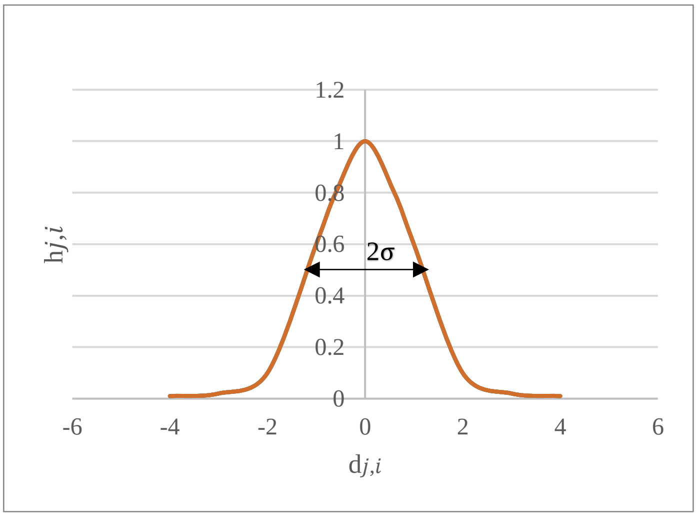

Figure 2.3: Gaussian curve-neighborhood function

Another unit property of the SOM is σ in the Gaussian function. This σ varies with time where time is the network iterations(n). As the iteration $\{n: n =$

$0,1,2, \ldots, \infty\}$ progresses, the σ is going to decrease with time. Thus, as σ decreases, the neighborhood shrinks gradually. Figure 2.4 explains clearly the shape of the neighborhood after it shrinks during the cooperation process when the iteration progressively narrows down the neighborhood of the winner neuron. Mathematically σ can be presented as

$$\sigma(n) = \sigma_o \exp\left(-\frac{n}{\text{Ʈ}_1}\right) \qquad Eq.(2.5)$$

Where $\sigma_o$is the initial σ and Ʈ_1 is the time constant.

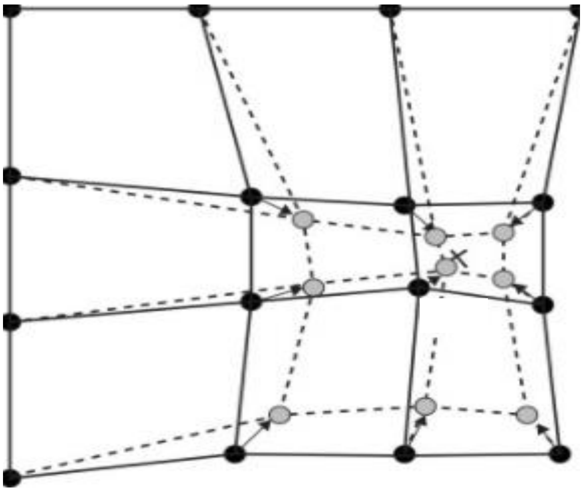

Figure 2.4: SOM network during the process of cooperation (Juha, 1999)

As a result, the neighborhood function will be

$$h_{j,i(\vec{x})}(n) = \exp\left(-\frac{{d_{j,i}}^2}{2\,\sigma^2(n)}\right), n = 0,1,2, \ldots \qquad Eq.(2.6)$$

In case of 1-D lattice, the distance is: $d_{j,i} = |j - i|$. For 2-D the distance, ${d_{j,i}}^2 = \|\vec{r_j} - \vec{r_i}\|^2$, where r_j is the position vector of the excited neuron j and r_i is the position vector of the winning neuron i. In fact, for higher dimensions the 2-D equation mentioned above is valid, but r_j and r_i will not consist of just two elements. It will consist of multiple elements depending upon the number of chosen dimensions for the lattice.

Adaptation: where the excited neurons associated with the input pattern keep decreasing the differences in the values of the discriminant function for the connected weights, such that the winning neuron’s response to any subsequent application for similar input pattern will become enhanced.

The learning mechanism behind the adaption process in SOM is the Hebbian learning. Hebbian learning is when the pre-synaptic and post-synaptic activities are correlated; when correlation occurs the synaptic connection will be strengthened. When correlation is absent, the synaptic connection is weakened. Hebbian learning is used to update the weights in SOM, yet it needs to be modified due to some limitations to suit the SOM. Therefore, the following term $g(y_j)\overrightarrow{w_j}$ is introduced to avoid the limitation, which is the saturation in the synaptic weight which it accrued during the continuity of feeding the same input pattern. So, $g(y_j)\overrightarrow{w_j}$ is called the forgetting term in Hebbian hypothesis, where $y_j$is a positive scaler function and for simplicity let $g(y_j) = \eta y_j$ which is a linear function.

The weight needs to be adjusted, not only for the winner neuron, but also for the neighbor neurons which are the excited neurons. Accordingly, $y_{j=}h_{j,i(\vec{x})}$, the topological neighborhood is maximum when the j neuron is the winner and as the lateral distance from the winning neuron progressively increases then y_j will progressively decrease.

Heibbian Hypothesis for adaptation (weight update):

$$\Delta\overrightarrow{w_j} = \eta\, y_j\vec{x} - g(y_j)\overrightarrow{w_j} \qquad Eq.(2.7)$$

Where η is the learning rate parameter, since $g\left(y_j\right) = \eta y_j$, thus, Eq. (2.7) can be written as $\Delta\overrightarrow{w_j} = \eta\ y_j\vec{x} - \eta y_j\overrightarrow{w_j}$. In order to include the winner neurons and the excited neurons, consider $y_{j=}h_{j,i(\vec{x})}$, then

$$\Delta\overrightarrow{w_j} = \eta\ h_{j,i(\vec{x})}\left(\vec{x} - \overrightarrow{w_j}\right) \qquad Eq. (2.8)$$

Which proves that w_j will be adjusted such that it should move closer to $\vec{x}$. Thus, during the learning phase $w_j$will align itself with $\vec{x}$, hence $\Delta\overrightarrow{w_j} = 0$

Using discrete-time formulation $\Delta\overrightarrow{w_j}$ can be written as

$$\overrightarrow{w_j}(n+1) = \overrightarrow{w_j}(n) + \eta(n)h_{j,i(\vec{x})}(n)\ (\vec{x} - \overrightarrow{w_j}) \qquad Eq. (2.9)$$

$$\eta(n) = \eta_0\, exp\left(-\frac{n}{\tau_2}\right), n = 0,1,2, \dots \qquad Eq. (2.10)$$

τ_2: Another time constant

So, the ultimate tendency is to align $\overrightarrow{w_j}$ with $\vec{x}$ for all winner neurons, yet all other neurons will also learn, but at a slower rate hence $h_{j,i(\vec{x})}(n)$ will be dropping down for the un-excited neurons. Ultimately, Eq. (2.10) is responsible for the topological ordering in SOM.

The two phases of the adaptive process (practical consideration) are:

1- Self-organizing (ordering): for topology arrangement. Learning rate should start with large value $\eta(n) \approx 0.1$ then it decreases to 0.01,$\eta_0 = 0.1$,$\tau_2 = 1000$: ,n=1000 iteration. Thus, topological neighborhood $h_{j,i(\vec{x})}$ starts with a large number of neurons then it decreases gradually; thus, neighborhood will keep shrinking till it is restricted to a very small neighborhood (Singupta, 2003).

2- Convergence phase: all neurons obtained in the topological stage will keep converging (tuning) till it reduces the error as much as possible. To complete the convergence, the number of iterations must be at least 500 times the number of neurons. For example, if we have 4X4 topological the number of iteration is 500*16=8000 iterations (Singupta, 2003).

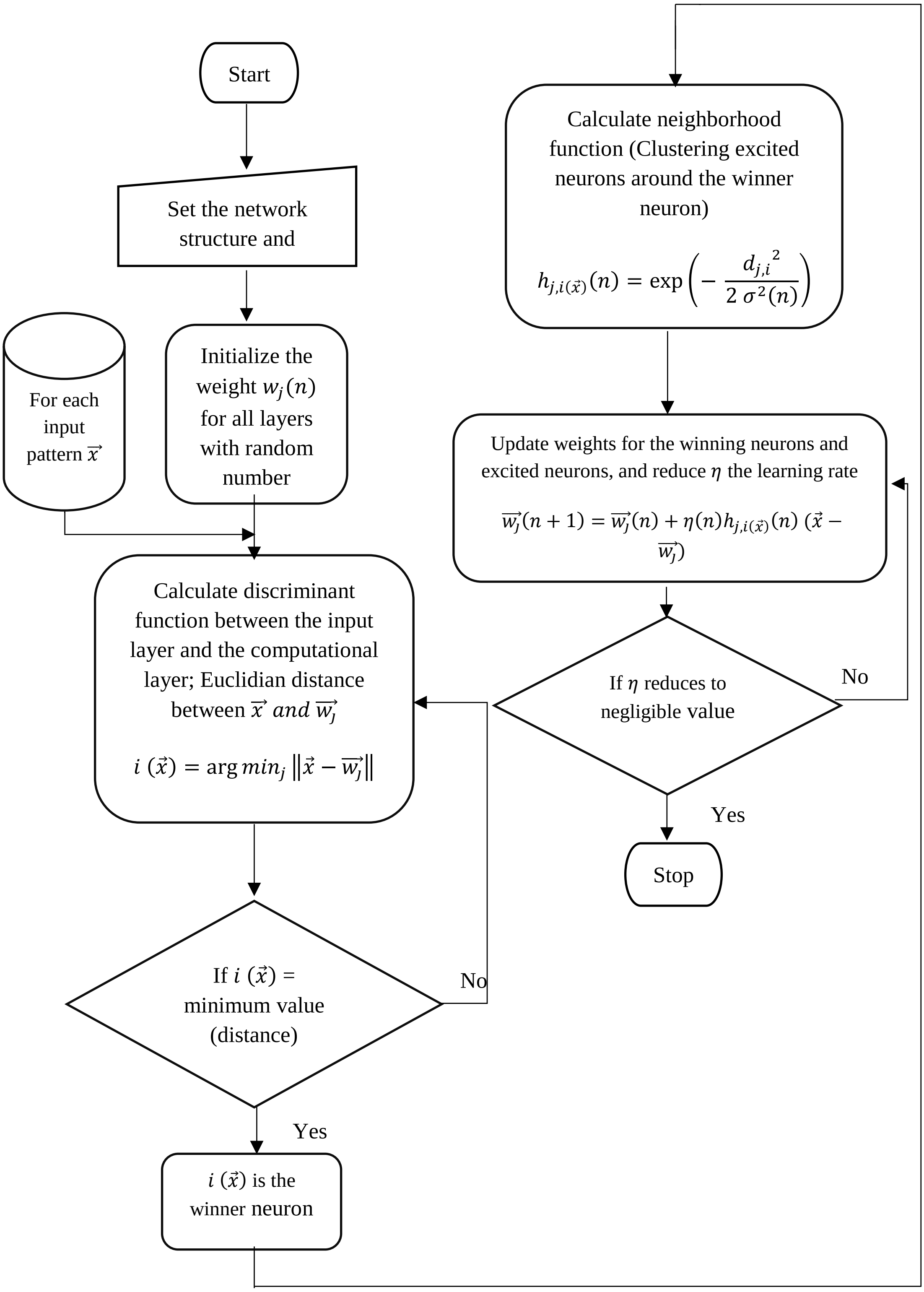
Start
Set the network structure and
For each input pattern $\vec{x}$
Initialize the weight $w_j(n)$ for all layers with random number
Calculate discriminant function between the input layer and the computational layer; Euclidian distance between $\vec{x}$ and $\vec{w_j}$
$i(\vec{x}) = \arg min_j \|\vec{x} - \vec{w_j}\|$
If $i(\vec{x})$ = minimum value (distance)
No
Yes
$i(\vec{x})$ is the winner neuron
Calculate neighborhood function (Clustering excited neurons around the winner neuron)
$h_{j,i(\vec{x})}(n) = \exp\left(-\frac{{d_{j,i}}^2}{2\,\sigma^2(n)}\right)$
Update weights for the winning neurons and excited neurons, and reduce η the learning rate
$\vec{w_j}(n+1) = \vec{w_j}(n) + \eta(n) h_{j,i(\vec{x})}(n)(\vec{x} - \vec{w_j})$
If η reduces to negligible value
No
Yes
Stop

Figure 2.5: Flowchart of self-organizing map neural network algorithm

2.5 Multi-Layer Perceptron Neural Network (MLP)

Multi-layer perceptron is considered one of the first neural network's type. Many neural networks depend on the structure of MLP. MLP is formed by cascading neurons (perceptrons) in several layers. The input vector is fed into each perceptron in the first layer, the output of the first layer's perceptrons has formed the input to the second layer's perceptrons, and so on, see Figure 2.1. Nodes of MLP are fully connected between layers. The arrangement and the type of neurons depend on the network type. The main parts of MLP network are:

1. **Input neurons** are carrying some action or information about the external environment. Input neurons do not perform any computation, but only pass the input vector to subsequent neurons.
2. **Output neurons** receive signals from the preceding neurons and transform it using formulas 2.1 and 2.2. Those values represent the output of the whole neural network.
3. **Hidden neurons** are the basis of the neural network. Those neurons receive the signal from the input neurons or preceding hidden neurons, process it by formulas 2.1 and 2.2 and then pass result signals to the subsequent (hidden or output) neurons.

2.6 Back-Propagation Neural Network (BP)

Back-propagation neural network is one of the most widely used neural networks. The back-propagation neural network is a multilayer feedforward network trained according to error back-propagation algorithm. The idea of the back-propagation network is to adjust the synaptic weight values and threshold values to achieve the minimum error sum of the square in the learning phase of the network. More details are shown in the sub-sections below (Borglin, 2011).

2.6.1 Back-Propagation Neural Network (BP) Structure

Back-propagation neural network is consist of two parts: First part is the Forward propagation of operating (function) signal: the input signal is propagated from the input layer, via the hidden layer, to the output layer. During the forward propagation of operating signal, the weight value and offset value of the network are maintained constant, and the status of each layer of the neuron will only exert an effect on that of next layer of the neuron. In case that the expected output cannot be achieved in the output layer; then it can be switched into the backpropagation of error signal.

The second part is Backpropagation of error signal: the difference between the real output and expected output of the network is defined as the error signal; in the backpropagation of error signal, the error signal is propagated from the output end to the input layer in a layer-by-layer manner. During the backpropagation of error signal, the weight value of a network is regulated by the error feedback. The continuous modification of weight value and the offset value is applied to make the real output of network closer to the desired one (Li, Cheng, Shi, & Huang, 2012).

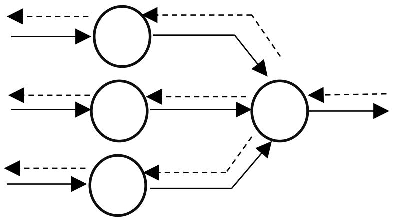

Figure 2.6: Illustration of signal flow for (operating signal and error signal) in multi-layer perceptron

2.6.2 Back-Propagation Neural Network (BP) Algorithm

Algorithm Formulas

Back-propagation is a training method used for a multi-layer neural network. It is also called the generalized delta rule. It is a gradient descent method which minimizes the total squared error of the output computed by the net (Rojas, 1996). Thus, it measures the performance using the sum of error squares function also it called the instantaneous error energy.

$$E(n) = \frac{1}{2}\sum_{j\in c} e_j^2\,(n) \qquad Eq.(\,2.11)$$

Where E (n) is the error energy at iteration n, c is the set of all neurons in the output layers and $e_j(n)$ is the error signal at each output neuron j at iteration n for all neurons in the output layer. If $d_j\,(n)$ denotes the desired response or target output for neuron j at iteration n, and $y_j(n)$ is the actual output, then

$$e_j(n) = \ d_j\,(n) - y_j(n) \qquad Eq.(2.12)$$

Hence, the average square energy is

$$E_{avg}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n) \qquad Eq.(2.13)$$

Where N is the total number of iterations of training patterns.

The instantaneous error energy $E(n)$, and therefore the average error energy $E_{avg}(n)$, is a function of all the free parameters (i.e. synaptic weights and bias levels) of the network. For given training set, $E_{avg}(n)$ represents the cost function as a measure of learning performance. The objective of the learning process is to adjust the free parameters of the network to minimize $E_{avg}(n)$. To do this minimization, an approximation similar in rational to the derivation of the LMS algorithm is used. Specifically, we consider a simple method of training in which the weights are updated on a pattern-by-pattern basis unit one epoch, that is, one complete presentation of the entire training set has been dealt with. The adjustments to the weights are made in accordance with the respective error computed each pattern presented to the network. (Haykin, 2009). In addition, the following equations are introduced to derive the BP neural network algorithm's formulas.

Induced local field of a neuron j is the output of the summation unit.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \qquad Eq. (2.14)$$

Where m is the number of neurons in the previous layer.

$$y_j(n) = \emptyset_j \left(v_j(n) \right) \qquad Eq. (2.15)$$

Where $\emptyset(n)$ is the activation function.

The back-propagation algorithm applies a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$ which is proportional to the partial derivative $\frac{\partial E(n)}{\partial w_{ji}(n)}$. Form the equations above $\frac{\partial E(n)}{\partial w_{ji}(n)}$ can be calculated using the chain rule of calculus.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} . \frac{\partial e_j(n)}{\partial y_j(n)} . \frac{\partial y_j(n)}{\partial v_j(n)} . \frac{\partial v_j(n)}{\partial w_{ji}(n)} \qquad Eq. (2.16)$$

Differentiate both sides of Eq. (2.11) with respect to $e_j(n)$, then

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \qquad Eq. (2.17)$$

Differentiate both sides of Eq. (2.12) with respect to $y_j(n)$, then

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \qquad Eq. (2.18)$$

Next, differentiating both sides with respect to $v_j(n)$, then

$$\frac{\partial y_j(n)}{\partial v_j(n)} = (\acute{\emptyset}) \left(v_j(n) \right) \qquad Eq. (2.19)$$

Finally, differentiating Eq. (2. 14) with respect to $w_{ji}(n)$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_j(n) \qquad Eq. (2.20)$$

The use of all equations between Eq. (2.17) and Eq. (2.20) in Eq. (2.16) yields

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\acute{\emptyset}\left(v_j(n)\right) y_j(n) \qquad Eq. (2.21)$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defining as the delta rule.

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \qquad Eq. (2.22)$$

Where η is the learning-rate parameter of the back-propagation. The minus sign shown in the equation above is an indication for gradient descent, hence, the gradient descent is an optimization technique for minimizing multidimensional smooth convex objective functions (Vishwanathan, 2008) in weight space. Combining Eq. (2.21) and Eq. (2.22) will result.

$$\Delta w_{ji}(n) = \eta\, e_j(n)\acute{\emptyset}\left(v_j(n)\right) y_j(n) \qquad Eq. (2.23)$$

Hence, the local gradient term $\delta_j(n)$ is defined by the product of the corresponding error signal $\partial e_j(n)$ for the output neuron j and derivative $\acute{\emptyset}\left(v_j(n)\right)$ of the associated activation function.

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial e_j(n)}.\frac{\partial e_j(n)}{\partial y_j(n)}.\frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\acute{\emptyset}\left(v_j(n)\right)$$

$$Eq. (2.24)$$

Then equation Eq. (2.23) can be re-written as

$$\Delta w_{ji}(n) = \eta\ \delta_j(n)\, y_j(n) \qquad Eq. (2.25)$$

From equations Eq. (2.24) and Eq. (2.25) it was noted that a critical factor involved in the calculation of the weight adjustment $\Delta w_{ji}(n)$ is the error signal $e_j(n)$ at the output neuron j. In this context, two distinct cases are identified. Case 1, when neuron j is an output node. Case 2, neuron j is hidden node. Note that although hidden neurons are not directly accessible, they share responsibility for error made at the output of the network.

Case 1 Neuron *j* is an output node.

Merely the neuron j is located in the output layer of the network. It is supplied with the desired response of its own, error signal $e_j(n)$ can be computed easily from Eq. (2.12) associated with this neuron. Accordingly, the local gradient $\delta_j(n)$ can be found using Eq. (2.24).

Case 2 Neuron *j* is a hidden node.

When neuron j is located in hidden layer, no specific desired response for that neuron. Consequently, the error information term for neuron j is determined recursively in term of the error information term of all neurons to which that hidden neuron j is directly connected as follows

$$\delta_j(n) = \acute{\emptyset}_j \left(v_j(n)\right) \sum_k \delta_k(n) w_{kj}(n) \qquad Eq. (2.26)$$

The equation above introduced a new index k to avoid any confusion between neuron j which is used as hidden neuron in Case 2, hence neuron k is an output node. $\delta_j(n)$ for hidden layer is derived as follows.

Eq. (2.24) is re-written as $\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}$

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \ \acute{\emptyset}_j \ \left(v_j(n)\right) \qquad Eq. (2.27)$$

Where neuron j is hidden as before mentioned. Hence Eq. (2.11) is re-written with k index

$$E(n) = \frac{1}{2}\sum_{k\in c} e_k^2\,(n) \qquad Eq.(2.28)$$

Differentiating Eq. (2.28) with respect to the function signal $\partial y_j(n)$.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \qquad Eq.(2.29)$$

Then using the chain rule to solve the Eq. (2.29)

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \qquad Eq.(2.30)$$

Recall Eq. (2,12) and change the index from j to k

$$e_k(n) = \ d_k\,(n) - y_k(n) \qquad Eq.(2.31)$$

$$= \ d_k\,(n) - \acute{\emptyset}_k\,\big(v_k(n)\big) \qquad Eq.(2.32)$$

Hence

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\acute{\emptyset}_k\,\big(v_k(n)\big) \qquad Eq.(2.33)$$

Also, recall Eq. (2.14) –induced local field-and change the index to k

$$v_k(n) = \ \sum_{j=0}^{m} w_{kj}(n) y_j(n) \qquad Eq.(2.34)$$

Where m is the total number of inputs (excluding the bias) applied to neuron k.

Next, differentiating Eq. (2.34) with respect to $y_j(n)$ yields.

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \qquad Eq.(2.35)$$

Thus, substituting Eq. (2.33) and Eq. (2.35) in Eq. (2.30)

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\,\acute{\emptyset}_k\left(v_k(n)\right)w_{kj}(n) \qquad Eq.(2.36)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k \delta_k(n)\;w_{kj}(n) \qquad Eq.(2.37)$$

Hence Eq. (2.37) used the definition of the local gradient $\delta_k(n)$with the index k From all the above we get the back-propagation formula Eq. (2.26) for the local gradient $\delta_j(n)$ for neuron j in hidden layer.

Activation Function

The knowledge of the activation function and its derivative is required to compute δ for each neuron of the multi-layer perceptron. For this derivative to exist, the function $\emptyset(.)$ Need to be continuous. In other words, the function $\emptyset(.)$ Need to be differentiable. In this study, the logistic function is used as activation function.

$$\emptyset_j\left(v_j(n)\right) = \frac{1}{1+\exp\left(-av_j(n)\right)} \qquad a\ > 0\ and -\infty < v_j(n) < \infty$$

$$Eq.(2.38)$$

Where $v_j(n)$ is the induced local field of neuron j. As a result, of the non-linearity in the Eq. (2.38) the amplitude of the output lies inside the range $0 \le y_j \le 1$. Differentiate Eq. (2.38) with respect to $v_j(n)$.

$$\acute{\emptyset}_J\left(v_j(n)\right) = \frac{a\exp\left(-av_j(n)\right)}{[1+\exp\left(-av_j(n)\right)]^2} \qquad Eq.(2.39)$$

Re-write Eq. (2.39) using $y_j(n) = \emptyset_j\left(v_j(n)\right)$ to eliminate some terms. So $\acute{\emptyset}_J\left(v_j(n)\right)$ is expressed as

$$\acute{}\emptyset_J\left(v_j(n)\right) = \frac{1+a\exp\left(-av_j(n)\right)-1}{[1+\exp\left(-av_j(n)\right)]^2} \qquad Eq.(2.40)$$

$$= \frac{1 + a \exp\left(-av_j(n)\right)}{[1 + \exp\left(-av_j(n)\right)]^2} - \frac{1}{\left[1 + \exp\left(-av_j(n)\right)\right]^2}$$

$$\acute{\emptyset}_j\left(v_j(n)\right) = ay_j(n)\left[1 - y_j(n)\right] \qquad Eq.(2.41)$$

For neuron j located in the output layer, $y_j(n) = o_j(n)$. Hence, local gradient for neuron j can be expressed as

$$\delta_j(n) = e_j(n)\, \acute{\emptyset}_j\, \left(v_j(n)\right) \qquad Eq.(2.42)$$

$$= a[d_j(n) - o_j(n)]\, o_j(n)\, [1 - o_j(n)]$$, Neuron j is an output node

Where o_j is the function signal at the output neuron j, and d_j is the desired response for it. On the other hand, for an arbitrary hidden neuron j. we may express the local gradient as

$$\delta_j(n) = \acute{\emptyset}_j\, \left(v_j(n)\right) \textstyle\sum_k \delta_k(n) w_{kj}(n) \qquad Eq.(2.43)$$

$$= ay_j(n)\, [1 - y_j(n)] \textstyle\sum_k \delta_k(n) w_{kj}(n),$$ Neuron j is hidden

For a sigmoid activation function, the synaptic weights are changed the most for those neurons in the network where the function signals are in their midrange. Thus, it is the feature of back-propagation learning that contributes to its stability as a learning algorithm.

Rate of Learning η and Momentum Constant α

A new parameter is introduced to BP-algorithm which is momentum term α. The purpose behind the addition of the momentum constant term α is to control the feedback loop acting around $\Delta w_{ji}(n)$. As explained earlier, Eq. (2.23) provides an approximation to the trajectory in weight space computed by the steepest descent. The smaller η is the learning rate, the smoother the trajectory and slower the convergence of the network to the optimum solution. On the other hand, increasing the learning rate η will lead to instability in the learning process hence the network will become oscillatory about the optimum weight due to the large steps in modifying the weights. As a result, the scientists modified the delta rule by adding the new term α momentum constant. Thus, the generalized delta rule is

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta\ \delta_j(n)\ y_j(n) \qquad Eq.(2.44)$$

2.6.3 Summary of the Back-Propagation Algorithm

The feedforward and back-propagation stages are shown in the following overview of the back-propagation algorithm.

Step 0. Set the learning parameters η to 0.1 and the momentum constant α to small values from 0.1 to 0.5

Determine the number of hidden layers as well as the number of neurons per layer.

Determine the maximum number of iteration

Set the minimum system error $E_{avg}(n)$

Step 1. Initialize the weights and biases for all layers to small random values between±1 or [0, +1] depends on the activation function.

Step 2. While stopping condition is false, do steps 3- 10, described below.

Step 3. For each training pair $(x(n), d(n))$ do steps 4- 10, where $x(n)$ is the input signal vector at iteration and $d(n)$ desired response vector at iteration n.

Feedforward:

Step 4. Determine the response vector for all the neurons in the *first layer*

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)x_i(n) \qquad Eq. (2.45)$$

$$y_j{}^{(l)}(n) = \emptyset_j\left(v_j(n)\right) \qquad Eq. (2.46)$$

Where $w_{ji}(n)$ is the interconnection weight vector for first hidden layer neurons $\emptyset_j$ is the activation function as given by Eq. (2.38) which is used by the first layer.

Step 5. Determine the response of the neurons in each of the following hidden layers, as well as output layer, using

$$v_j{}^{(l)}(n) = \sum_{i=0}^{m_0} (w_{ji}{}^{(l)}(n)y_i{}^{(l-1)}(n)) \qquad Eq. (2.47)$$

Where $y_i{}^{(l-1)}(n)$ is the output *function* signal of neuron i in the previous layer $l - 1$ at iteration n and $w_{ji}{}^{(l)}$ is the synaptic weight of neuron j in layer l that is fed from neuron i in layer $l - 1$.

$$y_j{}^{(l)}(n) = \emptyset_j\left(v_j(n)\right) \qquad Eq. (2.48)$$

Where $\emptyset_j$ is the activation function in layer l.

Step 6. Determine the mean squared error associated with pattern n using

$$E(n) = \frac{1}{2}\sum (d_j(n) - y_j(n))^2 \qquad Eq. (2.49)$$

Step 7. Determine the average (normalized) system error using

Where N is the total number of training patterns

$$E_{avg}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n) \qquad Eq.(2.50)$$

Back-propagation of error:

Step 8. Compute the error information terms and calculate the weight correction term for all neurons included in the output layer and layer.

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\,\acute{\emptyset}_j\left(v_j^{(L)}(n)\right) & \text{for neuron } j \text{ in output layer L} \\ \acute{\emptyset}_j\left(v_j^{(L)}(n)\right)\sum_k \delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n) & \text{for neuron } j \text{ in hidden layer } l \end{cases}$$

$$Eq.(2.51)$$

Where the prime in $\acute{\emptyset}_j$ denotes differentiation with respect to the

argument.

Update weights:

Step 9. Adjust the synaptic weight of the network l accordigng to the generalized delta rule:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n)$$

$$Eq.(2.52)$$

Step 10. Testing for stopping condition:

If the chosen maximum number of iteration n is reached or if the normalized system error calculated in step 7 is smaller than the pre-set value in step 0, then STOP; otherwise continue.

Flowcharts for the feedforward back-propagation neural network in learning and operation mode are given in Figures 2.4 and 2.5 respectively.

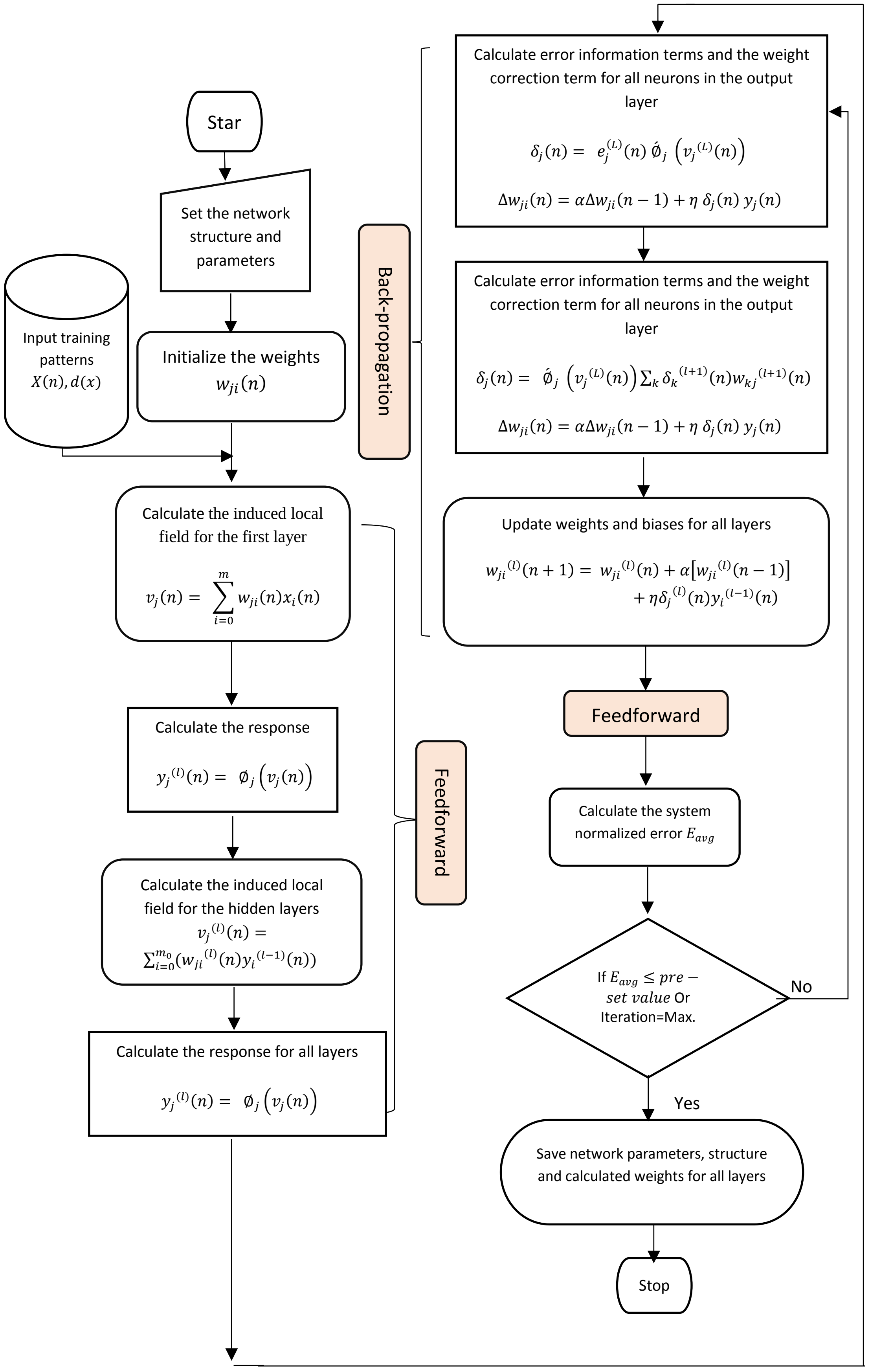
Star
Set the network structure and parameters
Input training patterns $X(n), d(x)$
Initialize the weights $w_{ji}(n)$
Calculate the induced local field for the first layer
$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)x_i(n)$
Calculate the response
$y_j^{(l)}(n) = \emptyset_j\left(v_j(n)\right)$
Calculate the induced local field for the hidden layers
$v_j^{(l)}(n) = \sum_{i=0}^{m_0}(w_{ji}^{(l)}(n)y_i^{(l-1)}(n))$
Calculate the response for all layers
$y_j^{(l)}(n) = \emptyset_j\left(v_j(n)\right)$
Feedforward
Back-propagation
Calculate error information terms and the weight correction term for all neurons in the output layer
$\delta_j(n) = e_j^{(L)}(n)\,\acute{\emptyset}_j\left(v_j^{(L)}(n)\right)$
$\Delta w_{ji}(n) = \alpha\Delta w_{ji}(n-1) + \eta\,\delta_j(n)\,y_j(n)$
Calculate error information terms and the weight correction term for all neurons in the output layer
$\delta_j(n) = \acute{\emptyset}_j\left(v_j^{(L)}(n)\right)\sum_k \delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n)$
$\Delta w_{ji}(n) = \alpha\Delta w_{ji}(n-1) + \eta\,\delta_j(n)\,y_j(n)$
Update weights and biases for all layers
$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n)$
Feedforward
Calculate the system normalized error E_{avg}
If $E_{avg} \leq pre-set\ value$ Or Iteration=Max.
No
Yes
Save network parameters, structure and calculated weights for all layers
Stop

Figure 2.7: Flowchart for a neural network in learning mode (Yousef, 2001)

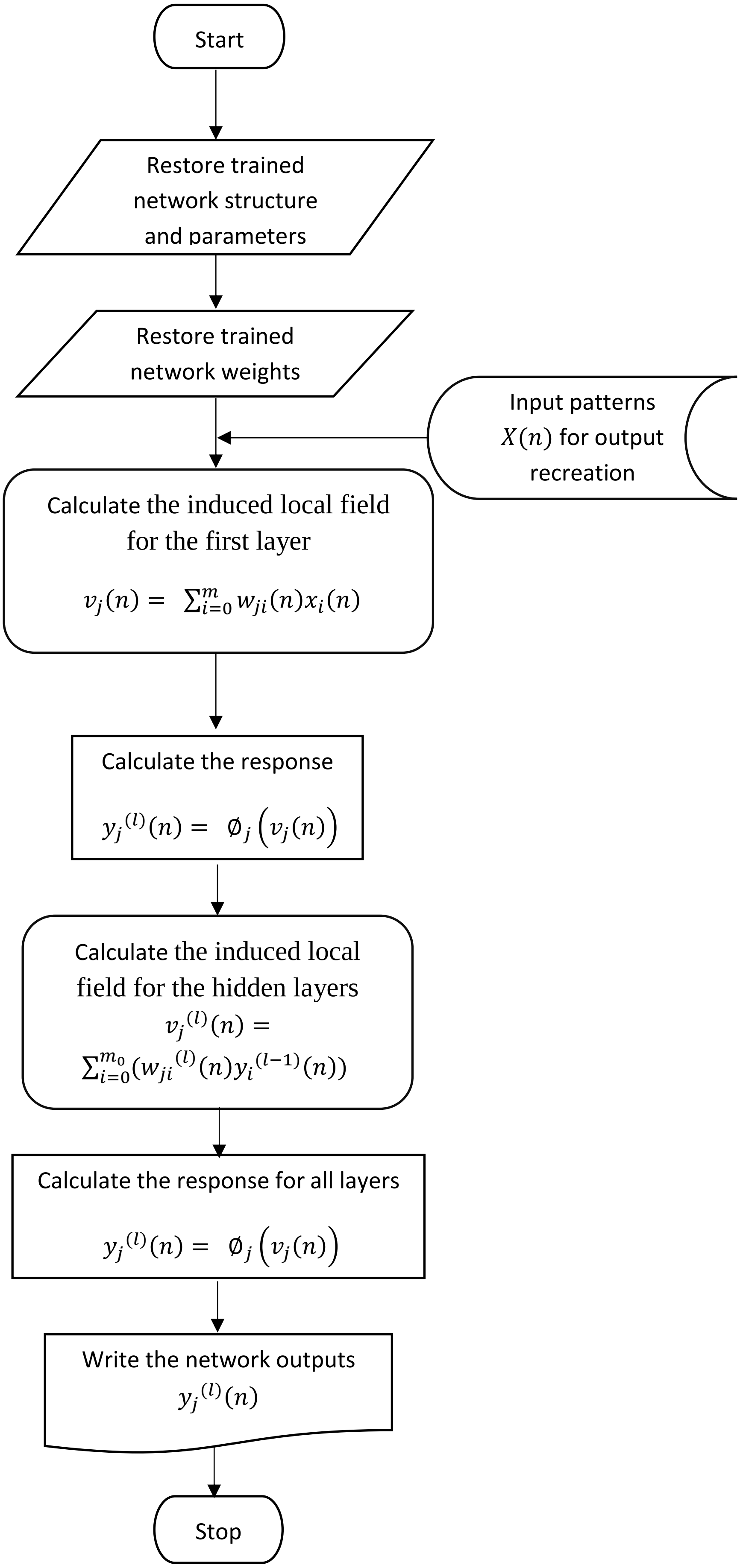
Start
Restore trained network structure and parameters
Restore trained network weights
Input patterns $X(n)$ for output recreation
Calculate the induced local field for the first layer
$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)x_i(n)$
Calculate the response
$y_j^{(l)}(n) = \emptyset_j\left(v_j(n)\right)$
Calculate the induced local field for the hidden layers
$v_j^{(l)}(n) = \sum_{i=0}^{m_0}(w_{ji}^{(l)}(n)y_i^{(l-1)}(n))$
Calculate the response for all layers
$y_j^{(l)}(n) = \emptyset_j\left(v_j(n)\right)$
Write the network outputs
$y_j^{(l)}(n)$
Stop

Figure 2.8: Flowchart for a neural network in operation mode (Yousef, 2001)

Chapter 3: Methodology

This chapter explains the methodology of modeling neural networks to solve the classroom scheduling problem. Data generation is initiated given five constraints. Furthermore, two neural networks are proposed to solve the classroom scheduling, which are: (1) the self-organizing feature map (SOM) neural network and (2) the feedforward back-propagation (FFBP) neural network. Detailed problems formulation and networks modeling are explained in this chapter.

3.1 Overview of ANNs

Making machines that can mimic the abilities of the human brain has been a dream for centuries. The idea came true with the computer revolution and demanding on data processing machines. Therefore, engineers created what is called 'Machine Learning,' which is the science of designing intelligent machines. The tools used to make Machine Learning are called neural networks (Rojas, 1996). A Neural Network can be thought of as a black-box which can correlate process inputs to its outputs based on a mapping relationship that is captured by the Neural Network during its training phase. According to Philip, training is a process where the machine parameters are modified in such a way that it will correlate with the needed output values. If the user defines the desired output values, the training is called supervised training. Otherwise, if the network picks the output values automatically from the data itself, the process is called unsupervised training (Philip, 2001).

One of the key benefits of neural networks is that they have the ability to process a large number of data with the same accuracy regardless of some factors like time and place. Furthermore, neural networks can find patterns from events which may appear as random; for instance, weather prediction. A neural network can predict the unseen scenarios relationships by mapping in some data that humans cannot capture. Consequently, using the Artificial Intelligence-based approach to solve the complexity of classroom scheduling is well worth investigating. Especially, because classroom scheduling holds so many parameters and variables.

3.2 Classroom Scheduling Problem

Classroom scheduling is a process whereby classrooms are allocated to a set of courses within the school hours so that it will meet specific constraints. The constraints in classrooms can be divided into two types: hard constraints, e.g. instructor cannot teach more than one course at the same time, and soft constraints, e.g. the instructor is able to submit a time preference (morning or evening) for class timing (Mahmud, 2014). In other words, designing a valid schedule should at least fulfill the hard constraints, adding soft constraints will add more flexibility in schedule but it won't cause a major issue if it was not exist in the schedule. Although a classroom timetable/schedule that meets both the hard and soft constraints can serve the objectives effectively, however meeting only the hard constraints can result in preparing a feasible classroom schedule (Edmund, 2006).

Also, partially meeting the hard constraints can produce a feasible initial guess for a sufficiently working schedule. In this research, a neural network- based approach will be adopted to prepare an initial guess for a preliminary classroom schedule that can meet specific hard constraints. To establish a feasible schedule a user needs to define/formulate a set of constraints that may depend on the circumstances of the workplace. For the purpose of conducting this research, the following constraints will be considered:

1) Teacher Conflict: Teacher can have more than one course to teach. However, a teacher cannot teach more than one course at the same time.
2) Course Conflict: A set of courses from the same level that must all be taken in the same year. For example, third level students in mechanical engineering in UAEU need to take Mechanics of Material (MECH305) and Geometric Modeling (MECH315) in the same semester to avoid any delay in their study plan). Hence for some groups of courses, no two courses from the same group can be scheduled at the same time. For details see Appendix.
3) Time Restriction: Some courses need a specific time; for example, a laboratory course needs to be scheduled for three consecutive hours to prevent the interruption of laboratory work. Due to this, such courses are often conducted at late times during the day.
4) Classroom Requirements: Classrooms cannot be assigned to more than one course within a specific time interval. Also, the classroom capacity and equipment needed for the class should be considered.

The goal of this project is to develop a methodology for solving a complex scheduling problem considering as many scheduling parameters/ constraints as possible. An Artificial-Intelligence based approach will be adopted to enable intelligent class scheduling. Thus, the black-box below shows the functional structure of the final product. At this point, executes of the product are not yet specified; this will allow a flexible selection of different networks to run the inputs. Also, when a new technology becomes available, the input can be substituted efficiently while keeping the same function of the product to achieve the desired output.

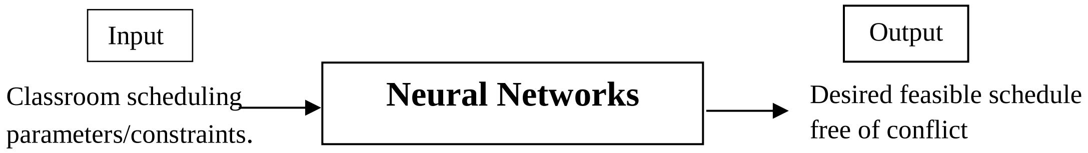
Input
Classroom scheduling parameters/constraints.
Neural Networks
Output
Desired feasible schedule free of conflict

Figure 3.1: Black box for the final product

Typically, a neural network can be constructed using systematic steps, which are defining the input and the output, training ANN models and validating and testing ANN models.

Step 1: Defining the input and the output. In this research designing / creating data is required, thus a set of input, x_n, will be defined and prepared to enter the ANN. The input data point vector will include all the constraints mentioned above. The creation of input vectors will be discussed in the following section.

Step 2: Training ANN model. Defining the data samples will play a crucial rule in choosing a proper neural network to train the data points. More details will be shown in the sections below.

Step 3: Validation and testing ANN model. This step is to verify the accuracy of the trained ANN by comparing the output against a set of the new data sample, noting that this step is needed for a supervised network. In contrast, the unsupervised network cannot be validated since the user does not know what to expect. Below is Figure 3.2 which is a flowchart simplifying the creation of ANN.

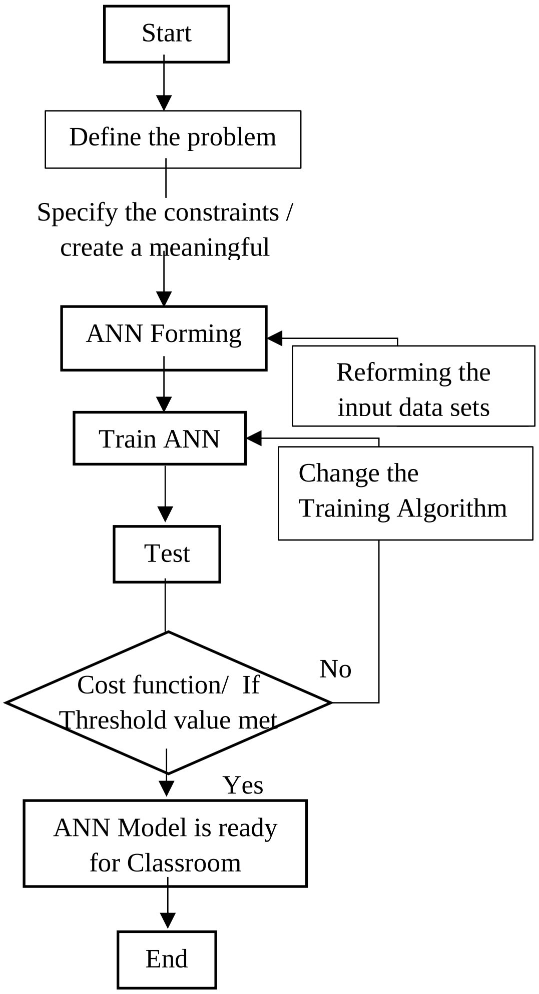
Start
Define the problem
Specify the constraints / create a meaningful
ANN Forming
Reforming the input data sets
Train ANN
Change the Training Algorithm
Test
Cost function/ If Threshold value met
No
Yes
ANN Model is ready for Classroom
End

Figure 3.2: Flowchart for ANN Creation

3.3 Creation of Data Point Samples

The datasets of a classroom schedule have been created by coding the input parameters. For example, the alphabetic letters [A: Z] in the course code are mapped into numbers [1:26], hence, the course code is transferred into numbers. Table 2 shows the mapping between the constraints and the input parameters, and between the input parameters and the numbers.

Table 2: Mapping constraints and input parameters

Constraint	**Input parameter**	**Code**
Teacher Conflict	Prof. ID	Numbers from 1 to 20
Time Restriction	Class time (AM/ PM)	Number 1 for AM
		Number 0 for PM
Course conflict	Course level	Number 3 for Third year
		Number 4 for Fourth year
		Number 5 for Fifth year
		Number 6 for sixth year
	Course name	For example, MECH348 represents Fluid Mechanics lab. The first digit represents the course level and the second and third digits the course name.
Classroom requirements	Course type (Theory / Laboratory)	Theory is 1
		Laboratory is 0

Moreover, the created datasets will form the shape of an input vector; see the input matrix Eq. (3.3) below. Note that the raw data used in this report is from a study plan of Mechanical and Electrical Department of Engineering College at UAE University. See Appendix for more details. Creating data points is essential in the selection of the proper neural network "training stage". Some assumptions were made to ease the training procedure, guarantee accurate output values and alleviate testing and debugging the datasets if anything goes wrong during the training stage. Hence, 78 data points were made. The table below illustrates the raw input points used to create the needed data. Thus, it forms the classroom schedule's requirement.

Table 3 demonstrates the number of involved professors which are 20 and that each professor will teach four subjects/courses. Also, it shows the course level, course type, and timing. In addition, the course name in letters is displayed to relate the courses to the professor easily. Note that the rest of the data is shown in Appendix.

Table 3: Original data with constraints

	Four lectures each Prof.				**Four lectures each Prof.**				**Four lectures each Prof.**				**Four lectures each Prof.**			
	Original data															
Prof ID	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
Course Level	3	4	4	3	3	4	5	5	3	4	6	4	3	4	5	5
Course (Theory/ Lab)	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
course name	48	29	30	11	6	33	40	42	10	50	15	17	11	29	31	30
Time (AM/PM)	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1
Course Name (letters)	Fluid Mech lab	Thermo. System	Thermal Eng. lab	Applied Thermo	Manuf. process	Intro to CAM	Sel.topics in Manufact.	Intro to Comp	Dynamic	System Dynamic lab	Adv. Dynamic and Control	Kinematic	Applied Thermo	Thermo. System	Intro. to Robotics	SEL Topics in Mech.

3.3.1 Data Normalization

Formulating data points and converting it into the input vector is the primary step to start the training. However, normalizing the input vector must take place before introducing the input vector to the network. The normalization removes geometrical biases towards some of the dimensions of the data vectors which will be inputted into the SOM neural network. This is done to insure that every bit of data gets treated in a "fair" manner. Also, this needs to be done to the data before it is entered into the backpropagation neural network. The reason for this normalization in the BP neural networks is that activation functions (e.g. sigmoid, hyperbolic tangent and Gaussian), produce a result that lies in ranges of [0,1] or [-1,1]. Thus, it is a must to normalize the input values, to insure it will be within the domain of [0,1] or [-1,1]. In addition, another way of posing this is to realize that all learning algorithms depend on numerical properties, so one should try to avoid small numbers, large numbers, and large differences (Nicholas, 2012).

As with all functions, if the input values are not in the domain, the result is not guaranteed to be appropriate (Nicholas, 2012). There are some ways to normalize data, for example, Z-score, the coefficient of variation or feature scaling. The most straightforward method for our data is feature scaling. If all input variable belong to some interval $\chi \epsilon \ [M\_min, M\_max]$, then the normalization formula (Mendelssohn, 1993) is:

$$x_{new} = \frac{x - M_{min}}{M_{max} - M_{min}} \qquad Eq. (3.1)$$

Pre-processing the data and training it in the neural network will result in an output vector. Consequently, the output vector needs post-processing or de-normalizing in order to interpret and present the output data in the understandable matter. The following formula is used for de-normalizing:

$$x = x_{new}(M_{max} - M_{min}) + M_{min} \qquad Eq. (3.2)$$

Table 4 specifies sample of data points after normalization- the rest of normalized data is presented in Appendix. Now, the data is ready for training, each row in the matrix below shows the five input parameters. Each column represents one set of an input vector. In this report, we are using 78 input vectors with each vector comprising professor ID, course level, course type, course name and class timing.

Table 4: Data after normalization

	Four lectures each Prof.				Four lectures each Prof.				Four lectures each Prof.				Four lectures each Prof.			
Prof ID	0	0	0	0	0.0526	0.052	0.052	0.052	0.105	0.105	0.105	0.10	0.1579	0.157	0.157	0.157
Course Level	0	0.333	0.3333	0	0	0.333	0.666	0.666	0	0.333	1	0.33	0	0.333	0.666	0.666
Course (Theo./ Lab)	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
Course Name	0.510	0.308	0.3191	0.117	0.0638	0.351	0.425	0.446	0.106	0.531	0.159	0.18	0.117	0.308	0.329	0.319
Time (AM/PM)	0	1	0	1	1	0	1	0	1	0	0	1	0	0	1	1

$$\text{Input Matrix} = \begin{bmatrix} 0 & 0 & 0 & 0. & 0.0526 & 0.0526 & 0.0526 & 0.0526 & 0.1053 & 0.1053 & 0.1053 & 0.1053 \\ 0 & 0.3333 & 0.333 & 0 & 0 & 0.333 & 0.6667 & 0.6667 & 0 & 0.333 & 1.0 & 0.333 \\ 0 & 1 & 0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 0 & 1.0 & 1.0 \\ 0.5106 & 0.3085 & 0.3191 & 0.1170 & 0.0638 & 0.3511 & 0.4255 & 0.4468 & 0.1064 & 0.5319 & 0.1596 & 0.1809 \\ 0 & 1.0 & 0 & 1.0 & 1.0 & 0 & 1.0 & 0 & 1.0 & 0 & 0 & 1.0 \\ 0 & 0.8700 & 0.90 & 0 & 0 & 0 & 0 & 0 & 0.2 & 1.0 & 0 & 0 \end{bmatrix}^{\Lambda}$$

Eq. (3.3)

3.4 ANN Formation

The most challenging and critical phase of this thesis is the building of the neural network models. Mainly, there are two significant challenges of this project. The first is how to interpret the multidimensional nonlinear data in the first draft of the schedule. Thus, the data needs to be directed to form groups that have a common mien. Each formed group is considered a classroom and each classroom will hold five unique, unpredictable features (or constraints). The second challenge faced is how to handle the unseen scenarios. For example, when dynamic changes happen at the beginning of any academic semester, such as opening a new section, closing an existing section and overlapping between some classes within the first draft.

3.5 Implementing SOM Neural Network

The solution starts with defining and understanding the behavior of input data to the neural network. Thus, analyzing the data, to be distributed in distinct regions or zones, may reveal a reasonable solution for the first challenge. Therefore, introducing Self-Organizing Feature Map neural network to define those regions is an appropriate solution for the following reasons:

- The presented data in this report has a clear non-linear relationship due to the different types of constraints.

- SOM can reduce the size of the problem from a five-dimensional (multidimensional) map to a two-dimensional map, while maintaining all the information about the features (or constraints) valued. Hence, each input vector consists of five different constraints.
- SOM neural network uses unsupervised learning, so the training samples contain only input patterns. As a result, the system does not need to be well defined, which is the case with our data.

SOM NN comprises two layers. The first layer is the input which consists of the data points (sections) and neurons and the second is an output layer in the form of a two-dimensional map that locates classrooms according to their degree of similarity after clustering. The dimension of the output layer depends on the amount of data being analyzed. The higher the volume of data, the larger the output layer.

As for the architecture of SOM neural networks, input data is propagated within the system through forward connections, where connections originate in the input layer and implant in the subsequent layer (output layer). Furthermore, the lateral connections which appear between neurons of the same layer, can be seen in the output layer, as introduced by Kohonen. Thus, it represents the cooperation stage in SOM training.

After the learning process is completed, the output layer results in clusters (groups). Each cluster has a centroid, which is the neuron around which the data points are grouped; see the Figure 3.3. It is worth mentioning that the position of the centroid neuron of the cluster is represented as a located in the X-Y plane of the network map, which is also the weight vector of the centroid neuron itself. Hence, the number of neurons designate the number of suggested classrooms, which in our research is 18 classrooms.

Furthermore, each ball in Figure 3.3 represents one section, each of which is comprised of professor ID, course level, course type, course name and class timing. The colors of the data points cloud in the input layer are a representation of the courses based on its constraints or features. As a result, the data points with the same color signify the sections with similarities in their features. These data points, which have the same color, are clustered to form the classrooms in the output layer by SOM NN. A related point to consider is that the neurons in the input layer had no defined color; they are shown in gray color in Figure 3.3. However, after clustering in the output layer, the neurons (center of clusters) are converted to match the color of their group; which is simply the function "clustering" of SOM NN.

A further splitting is done for large clusters if the classroom is assigned to more than four courses.

Note that the information in Figures 3.3 and 3.4 below is simply a graphical representation which serves as a model for the neural network our research endeavors to create. In real mathematical modeling of SOM, the neurons migrate between the similar features. In contrast, the data points remain fixed.

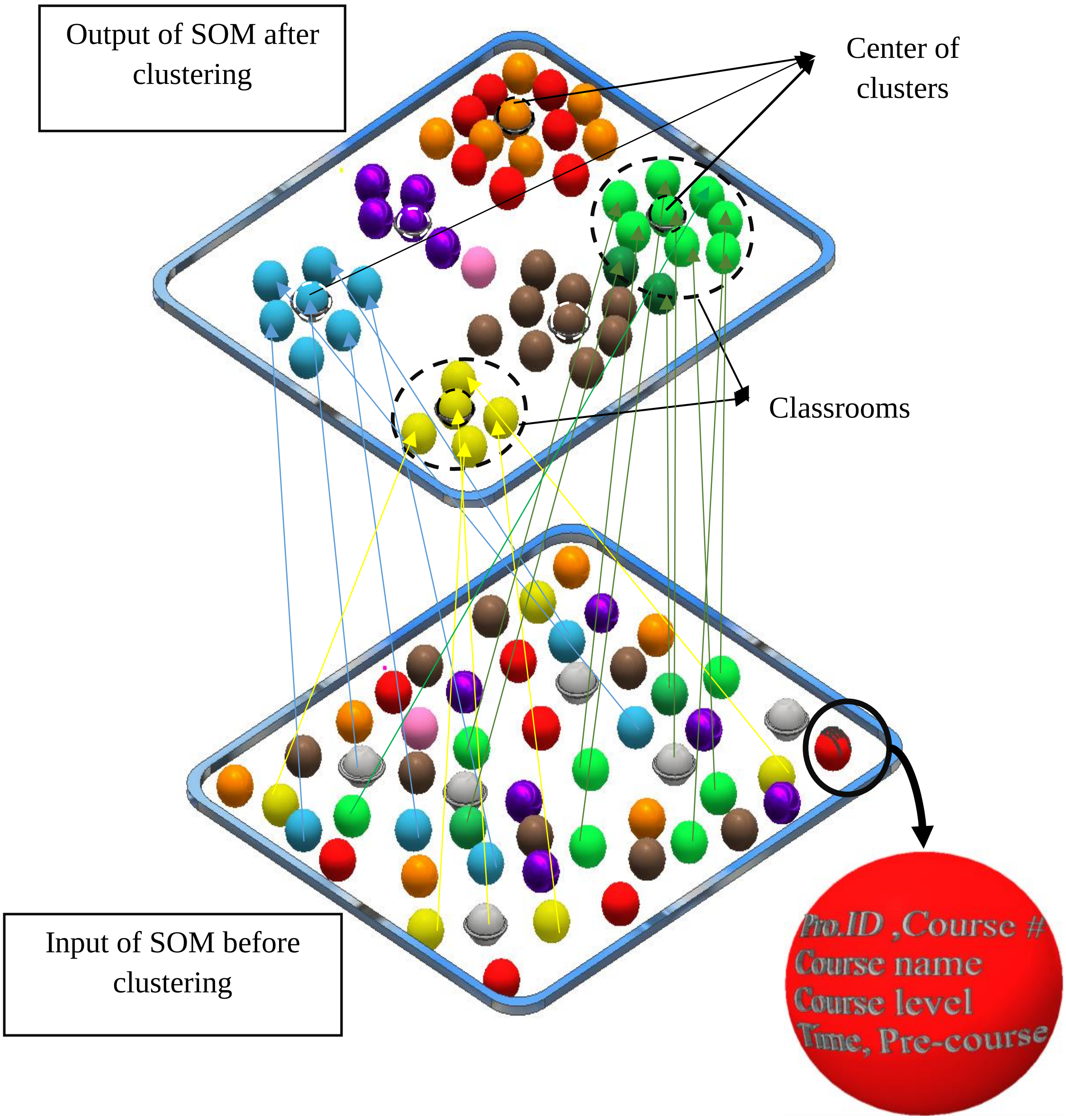

Figure 3.3: Graphical representation of first stage of sections (courses) separation by SOM Neural Network

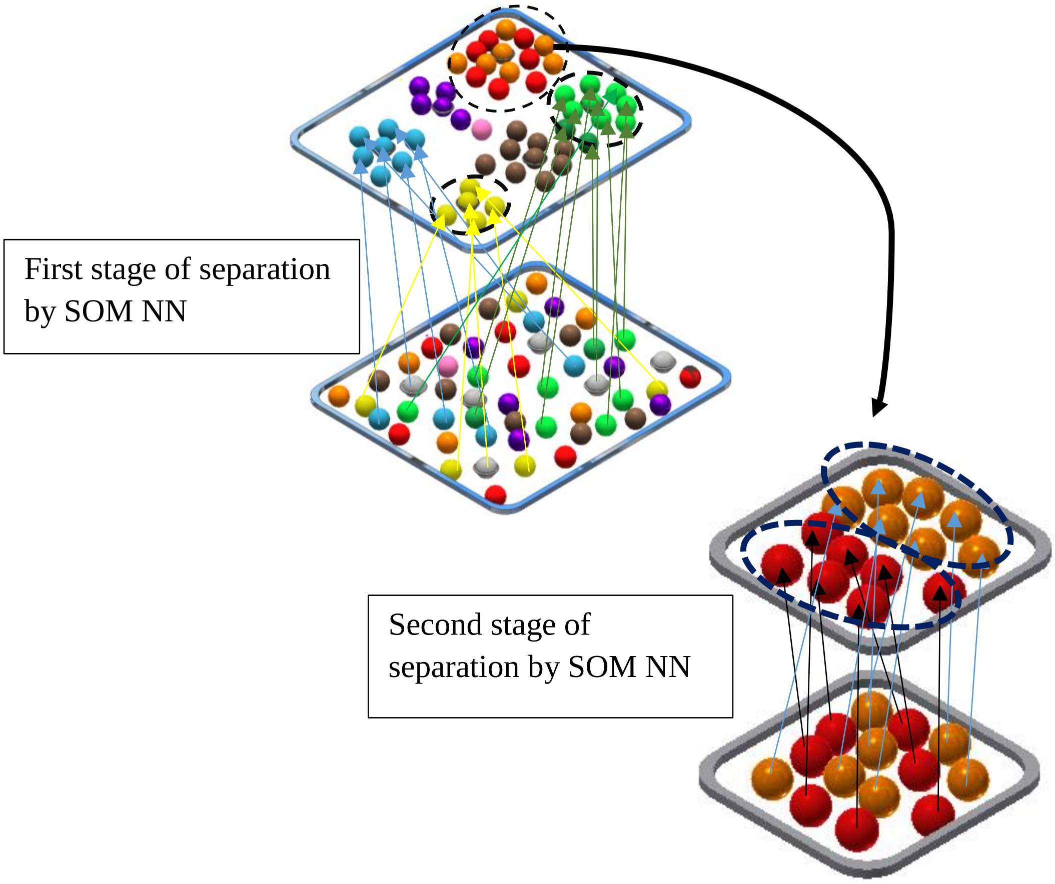
First stage of separation by SOM NN
Second stage of separation by SOM NN

Figure 3.4: Graphical representation of SOM separations

3.6 Implementing FFBP Neural Network

To preserve the produced classroom timetable done by SOM neural network and to allow for new sections or updates to the classroom schedule to be made with minimal changes, the feedforward back-propagation neural network is introduced. The clustered output data done in SOM network is now the input data for FFBP neural network.

As previously explained in Chapter 2, feedforward neural network consists of at least three layers: an input layer, hidden layer, and output layer. The network of all three layers is fully connected. Neurons in the input layer are linked to neurons in the hidden layer, and in-return the hidden layer's neurons are fully connected to the output layer's neurons; this is why it is called the feedforward part of FFBP network. Concerning back-propagation of an error signal, which is a training method for neurons to adapt their weights to attain new knowledge, all neurons are fully connected through all layers and propagate from the output layer to the input layers. This insures the input data passes through all layers, including the hidden layers. In general there is no particular technique to determine the number of the hidden layer. Thus, from experiments researchers advise to use two rules to launch the FFBP model: (i) number of hidden layers equals one; and (ii) the number of neurons in that layer is the mean of the neurons in the input and output layers.

As the Figure 3.5 illustrates, learning mode is when the FFBP neural network takes the input data (courses) and propagates it in a forwarding fashion. The patterns resulting from the hidden layers arriving at the output layer are then compared with the desired (associated) output pattern (classrooms numbers) to calculate an error signal. Then the error signal for each target output pattern is propagated from the output layer to the input layer, to update the weights in each layer of the network. After the training phase, the network can be tested on a new set of samples to see how well it classifies new patterns (new course).

Operation mode, which is the trained FFBP neural network, classifies new data. Thus, the network will be able to identify whether a particular data point (i.e., new course) belongs to classroom 1, 2, or 3, etc.

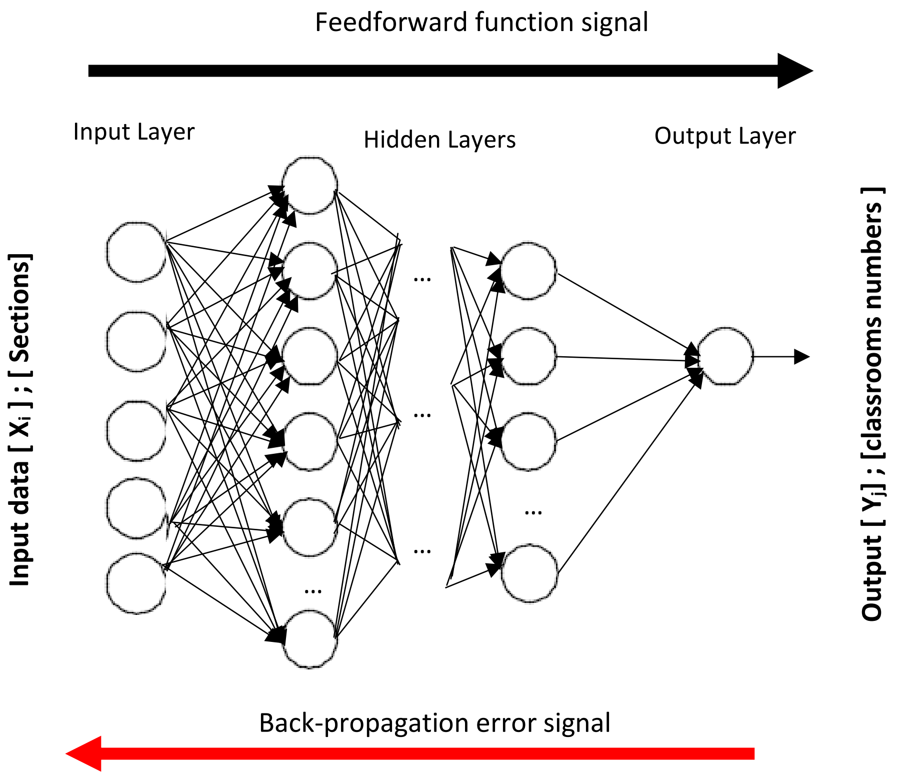
Feedforward function signal
Input Layer
Hidden Layers
Output Layer
Input data [X_i] ; [Sections]
Output [Y_j] ; [classrooms numbers]
Back-propagation error signal

Figure 3.5: Learning mode in FFBP NN

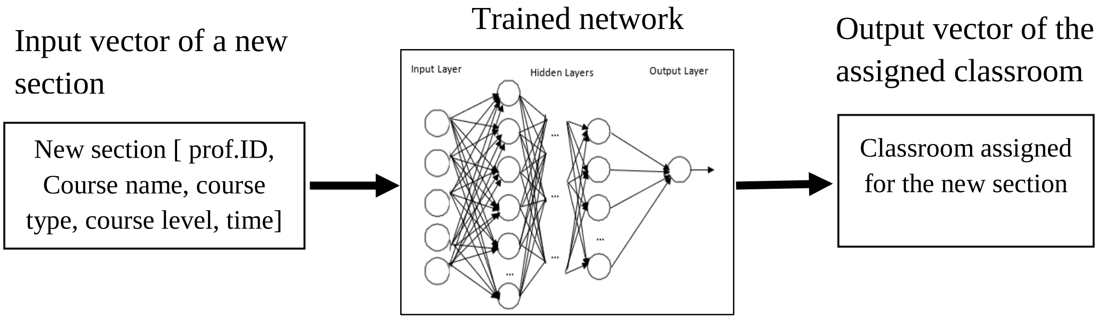
Input vector of a new section
New section [prof.ID, Course name, course type, course level, time]
Trained network
Input Layer
Hidden Layers
Output Layer
Output vector of the assigned classroom
Classroom assigned for the new section

Figure 3.6: Graphical representation of the operational mode in FFBP NN

In summary, Figure 3.6 illustrates a set of input vectors comprising five constraints are introduced to a Self-Organizing Feature Map (SOM) neural network for classroom section classification and separation, using some cluster centers equal to the available rooms. The SOM demonstrated robust capability in clustering the sections into groups, comprising courses with conflicts based on the defined constraints, hence it identified classes to be sequentially scheduled in one room. A second stage SOM is used to further split the big clusters. Moreover, to fit newly created classrooms into the SOM generated timetable, the output from SOM is used to train a Feedforward Back Propagation (FFBP) neural network which then shows where the new section can be allocated without disrupting the already existing schedule created by the SOM. The trained FFBP is used to accommodate the new sections without the need to re-cluster with SOM. In combination, the SOM and the trained FFBP managed to prepare a conflict-free schedule successfully. Figure 3.7 represents the suggested artificial intelligence approach for classroom scheduling graphically.

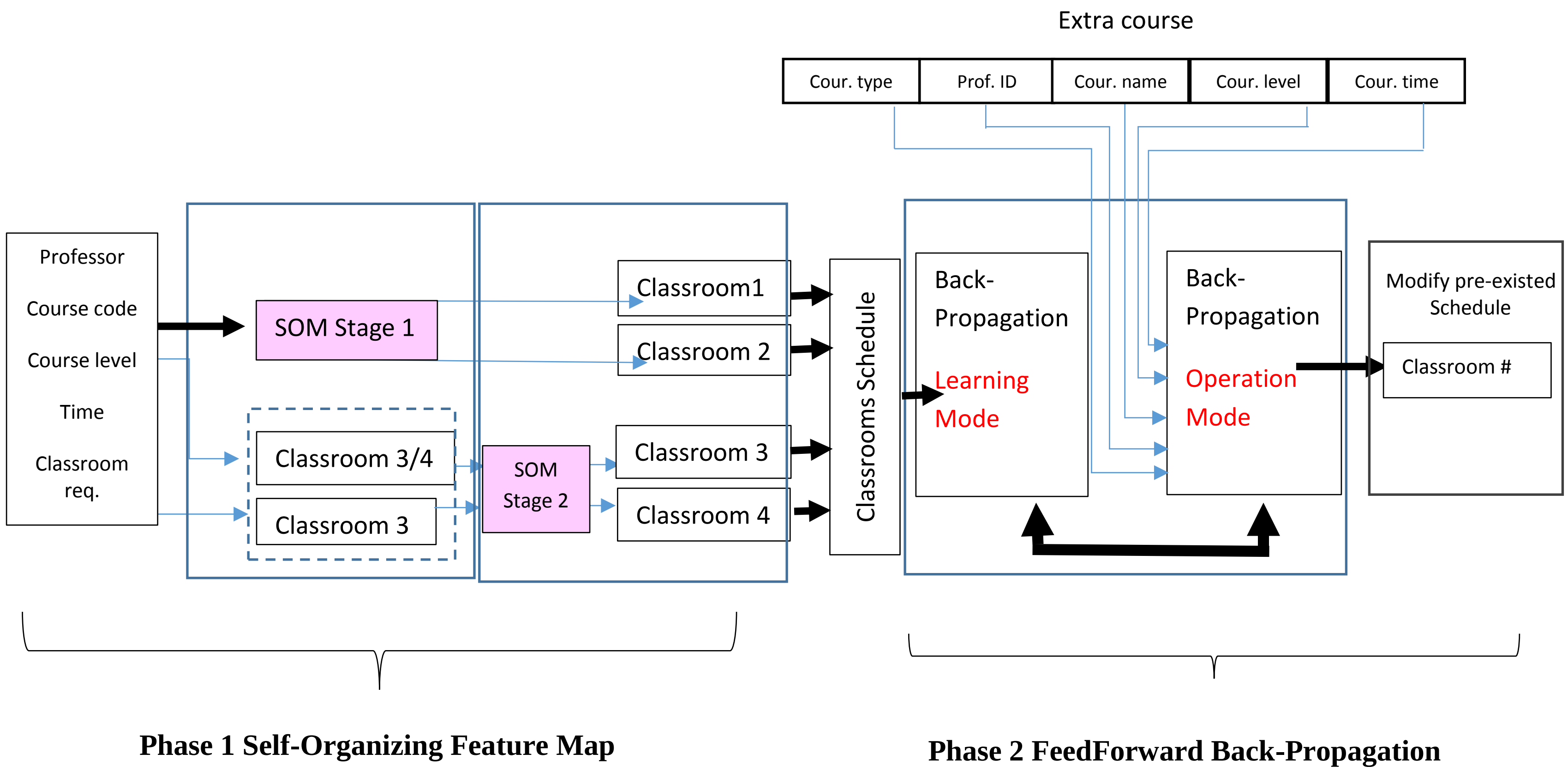

Figure 3.7: Chart representation of AI approach for classroom scheduling

Chapter 4: Tests Results and Discussion

The neural networks algorithms implemented in this research are written in Matlab. In this chapter, mathematical models and tests are described, and the results of implementing SOM NN and FFBP NN are presented and discussed.

4.1 Mathematical Modeling of SOM for Classroom Scheduling

The SOM NN takes the set of input vectors (sections) and maps it with the neurons of a two-dimensional hexagonal grid. Each neuron in the 2D grid is assigned a weight vector $w_{jD} = (w_{j1}, w_{j2}, \dots, w_{jD})$ which has similar dimensionality as the input vector, where j = (1, 2, …, l) and l is the total number of neurons in the network and D is dimension of the input vector. Hence, the input pattern of our model has five dimensions (constraints), $D = 5$. According to (Fisher, 2006), the weights represent the centers of clusters in the 2-D map. Additionally, the number of neurons, j, is determined as the following: "$Number\ of\ neurons\ \leq\ number\ of\ classrooms$". Thus, the number of classrooms must be sufficient to satisfy the given number of sections; moreover, the minimum number of neurons should be at least equal to the number of rooms to accommodate the given courses. According to the generated data, the number of data points (input vectors) is 72. Thus, each professor will teach four courses. However, for the sake of making the network more realistic, some scenarios is added which is: two professors are assigned to teach only three sections. Hence 6 additional input data are added so that the total number of input vectors is 78. Accordingly, the number of neurons j can be determine as the following:

$$Number\ of\ neuron = \frac{Number\ of\ opened\ sections\ for\ the\ current\ semester}{Number\ of\ sections\ that\ can\ occupy\ one\ classroom}$$

$$= \frac{72}{4} = 18\ classrooms$$

As the above formula reveals (i.e., the number of classrooms needed to house all courses and sections), the total number of neurons l = 18. As the number of constraints = 5, the weight matrix number will be $18\ X\ 5 = 90$.

$$w_{jD} = \begin{bmatrix} w_{1\,1} & w_{1\,2} & w_{1\,3} & w_{1\,4} & w_{1\,5} \\ w_{2\,1} & w_{2\,2} & w_{2\,3} & w_{2\,4} & w_{2\,5} \\ w_{3\,1} & w_{3\,2} & w_{3\,3} & w_{3\,4} & w_{3\,5} \\ w_{4\,1} & w_{4\,2} & w_{4\,3} & w_{4\,4} & w_{4\,5} \\ w_{5\,1} & w_{5\,2} & w_{5\,3} & w_{5\,4} & w_{5\,5} \\ w_{6\,1} & w_{6\,2} & w_{6\,3} & w_{6\,4} & w_{6\,5} \\ & & \vdots & & \\ w_{18\,1} & w_{18\,2} & w_{18\,3} & w_{18\,4} & w_{18\,5} \end{bmatrix} \quad Eq. (4.1)$$

Where w_{jD} is the weight matrix, j represents the classrooms [$j = 1, 2,3, \dots 18$], and D represents the constraints [$D = 1,2, \dots ,5$].

Figure 4.1, clearly exemplifies the interconnection between the input vector (input layer) and the computational layer (output layer).

$$\begin{array}{r c} Classroom\ 1: & \begin{bmatrix} w_{1\,1} & w_{1\,2} & w_{1\,3} & w_{1\,4} & w_{1\,5} \\ w_{2\,1} & w_{2\,2} & w_{2\,3} & w_{2\,4} & w_{2\,5} \\ w_{3\,1} & w_{3\,2} & w_{3\,3} & w_{3\,4} & w_{3\,5} \\ w_{4\,1} & w_{4\,2} & w_{4\,3} & w_{4\,4} & w_{4\,5} \\ w_{5\,1} & w_{5\,2} & w_{5\,3} & w_{5\,4} & w_{5\,5} \\ w_{6\,1} & w_{6\,2} & w_{6\,3} & w_{6\,4} & w_{6\,5} \\ & & \vdots & & \\ w_{18\,1} & w_{18\,2} & w_{18\,3} & w_{18\,4} & w_{18\,5} \end{bmatrix} \end{array}$$

$$\begin{array}{r|ccccc} Classroom\ 1: & w_{1\,1} & w_{1\,2} & w_{1\,3} & w_{1\,4} & w_{1\,5} \\ Classroom\ 2: & w_{2\,1} & w_{2\,2} & w_{2\,3} & w_{2\,4} & w_{2\,5} \\ Classroom\ 3: & w_{3\,1} & w_{3\,2} & w_{3\,3} & w_{3\,4} & w_{3\,5} \\ Classroom\ 4: & w_{4\,1} & w_{4\,2} & w_{4\,3} & w_{4\,4} & w_{4\,5} \\ Classroom\ 5: & w_{5\,1} & w_{5\,2} & w_{5\,3} & w_{5\,4} & w_{5\,5} \\ Classroom\ 6: & w_{6\,1} & w_{6\,2} & w_{6\,3} & w_{6\,4} & w_{6\,5} \\ \vdots & & & \vdots & & \\ Classroom\ 18: & w_{18\,1} & w_{18\,2} & w_{18\,3} & w_{18\,4} & w_{18\,5} \end{array}$$

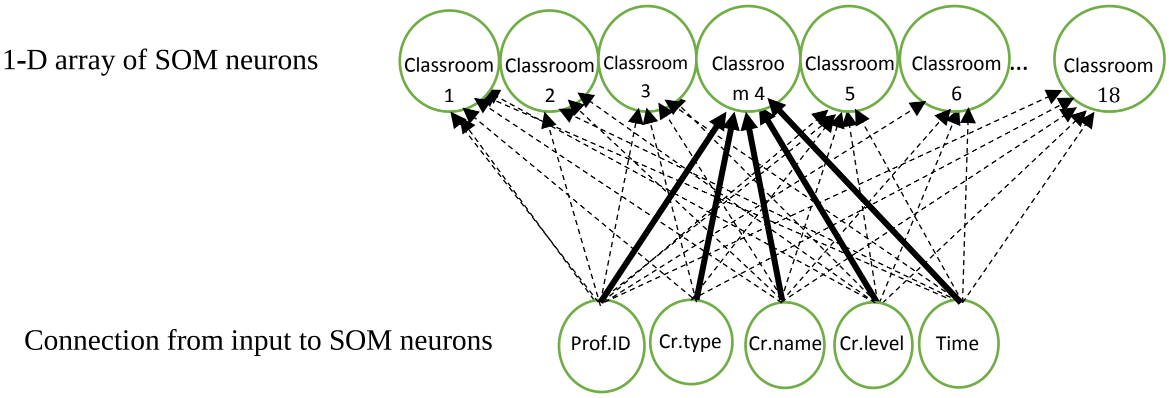

Figure 4.1: SOM NN-mathematical model

Competition Phase: By using the Euclidian distance Eq. (2.46), the winning input vector can be found by calculating the distances between the input vectors and the weight of the neurons (classrooms) such that $d\ (classroom\ j)$ is minimum.

$$d(classroom1)^2 = [(prof.ID - w_{11})^2 + (Cr.type - w_{21})^2 + (Cr.name - w_{31})^2 + (Cr.level - w_{41})^2 + (Time - w_{51})^2]$$

$$d(classroom2)^2 = [(prof.ID - w_{12})^2 + (Cr.type - w_{22})^2 + (Cr.name - w_{32})^2 + (Cr.level - w_{42})^2 + (Time - w_{52})^2]$$

$$d(classroom3)^2 = [(prof.ID - w_{13})^2 + (Cr.type - w_{23})^2 + (Cr.name - w_{33})^2 + (Cr.level - w_{43})^2 + (Time - w_{53})^2]$$

$$\vdots$$

$$d(classroom\ 18)^2 = [(prof.ID - w_{1\ 18})^2 + (Cr.type - w_{2\ 18})^2 + (Cr.name - w_{3\ 18})^2 + (Cr.level - w_{4\ 18})^2 + (Time - w_{5\ 18})^2]$$

Cooperation Phase: After the competition phase, the cooperation phase is determined by calculating the neighborhood function (distance between the excited neurons and the winner neuron). The neighborhood function should satisfy two requirements: it must be symmetric, and decreases monotonically with the increase of the distance.

$$h_{j,i(\vec{x})}(n) = \exp\left(-\frac{{d_{j,i}}^2}{2\,\sigma^2(n)}\right), n = 0,1,2, \dots\ is\ the\ iterations$$

Hence, ${d_{j,i}}^2$ is the 2-D distance, ${d_{j,i}}^2 = \left\|\vec{r_j} - \vec{r_i}\right\|^2$, where, r_j is the position vector of the excited neuron j and r_i is the position vector of the winning neuron i.

Adaptation phase: Next the weights must be updated for the winning neurons and excited neurons, and reduce the learning rate, η. Let the learning rate start from $\eta_{(t=0)} = 0.6$ and decrease till $\eta_{(t=t(n))} = 0.01$ until the changes become less than the predefined threshold. This is then the cost function:

$$\overrightarrow{w_j}(n+1) = \overrightarrow{w_j}(n) + \eta(n) h_{j,i(\vec{x})}(n) \left(\vec{x} - \overrightarrow{w_j}\right) \qquad Eq. (4.2)$$

Which is simply

$$\overrightarrow{w_j}(new) = \overrightarrow{w_j}(old) + \eta h_{j,i(\vec{x})} \left(\vec{x} - \overrightarrow{w_j}(old)\right) \qquad Eq. (4.3)$$

4.2 SOM Neural Network Parameters for Tests in MATLAB

The behavior of SOM was explained earlier in Chapter 2. However, understanding and selecting the proper parameters will ensure a better performance and faster convergence of the neural network. Unfortunately, there is no definite and explicit method to select optimal parameters for the used SOM NN model. Accordingly, general trends from previous research, as well as trial and error methods were followed to find the best parameters values.

MATLAB Software: Matlab is an open source tool with high-performance language for technical computing. It has built-in functions for different types of neural networks, hence, it uses Graphical User Interface (GUI) and/or Lines commands (Mathworks, 1994-2018). However, using the command lines allows the luxury to fine tune the network parameters easily. Therefore, in this research the command lines in Matlab are used to run the SOM tests using the generated data above. Hence, the line commands are shown in Appendix.

Topology in SOM: Picking a suitable topology will result in a 2-D map with well-clustered data. To this end, different tests were done using 11 input vectors from the generalized data sets on three different topologies (grid topology, hexagonal topology, and random topology; see Fig. 4.2). Sometimes the shape of the chosen topology can be justified, sometimes it cannot. For example, in our case, the use of hexagonal topology was a more natural fit based on the already consisting shape of our data points. Furthermore, hexagonal topology has the highest number of adjacent neurons per neuron (López, 2014), which gives more flexibility in tuning the clusters as compared to the grid topology. Hence, the preferred use of hexagonal topology was confirmed after several tests, as compared with the random and the grid topologies.

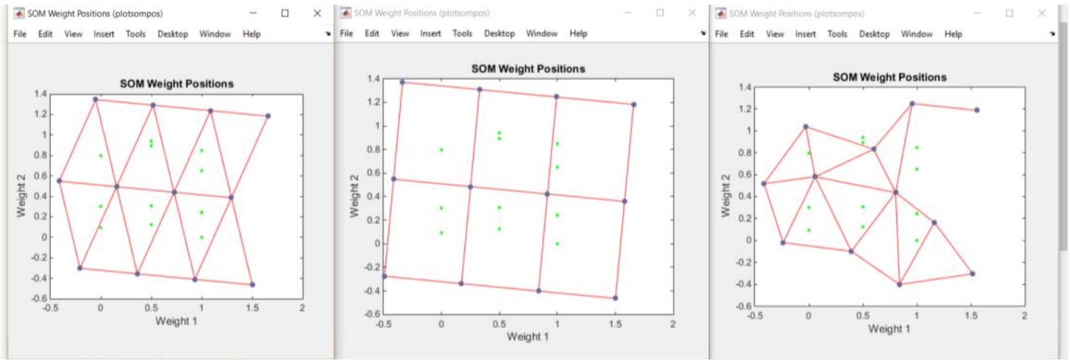

Figure 4.2:Three different types of topologies, HexTop, GridTop and RandTop

No. of Epochs: The number of iteration must be at least 500 times the number of neurons. In our case 500 X 18=9000 iterations, hence, the iteration should start from 9000. Recall that an epoch in learning means using all the training samples once. So, after trial and error method, the approximate number of needed epochs to converge was 100,000 epochs. Hence, the suggested parameters are taken from two resources either from different researchers or found by try and error experimentally.

Learning Rate η: The learning rate η started with 0.6 and decreased till it reached a steady state in term of the changes in the neurons convergence. No significant changes were noticed in the clustering map when η reached 0.1.

Initial neighborhood radius σ: The radius should start out as the radius of the network, and approach zero, at which time the radius is simply the winner node. Any nodes found within the radius of the winner are adjusted to make them more like the input vector. The value of σ decreases with the number of iterations (for full discussion see Chapter 2). Hence, σ started with 6 and gave the best results at 4. Thus, the result is that the neighboring neurons tend to have similar weight vectors and to be responsive to similar input vectors. This result was found experimentally via Matlab.

4.3 SOM Results and Discussion

The purpose of using SOM in this thesis is to cluster the data points. Thus, the SOM distributed the data points (sections) over the neurons (clusters), to guarantee a conflict-free schedule for each cluster. Hence, each cloud point (or cluster of data) has shared features which reveal the points of potential conflict and highlight sections that need to be scheduled in the same classroom. It is worth mentioning that the suggested solution is considered an initial estimate for the classroom schedule.

4.3.1 Matlab Analysis Plots

After tens of runs and tests, Figures 4.3 and 4.4 show the data points before and after clustering distributed over 6X3 hexagonal topology in SOM. The weight position plot below shows the data points (sections) as green dots in terms of two first features: professors IDs and course type. The neurons’ weight vectors are plotted in dark-blue dots according to their first two weights only. The red lines indicate which neurons are neighbors. Note how the neurons spread out with neighboring neurons representing the adjacent arrows of the sections' features in space. The 2-dimensional map appears folded in some places in the plot because it is spread over five dimensions of the sections features. Nonetheless, the neurons distribution in the map is expected to be well organized; since the input data, or first two features, are well distributed from the beginning. However, the main target from clustering the data is to extract features from each cluster (classroom) separately.

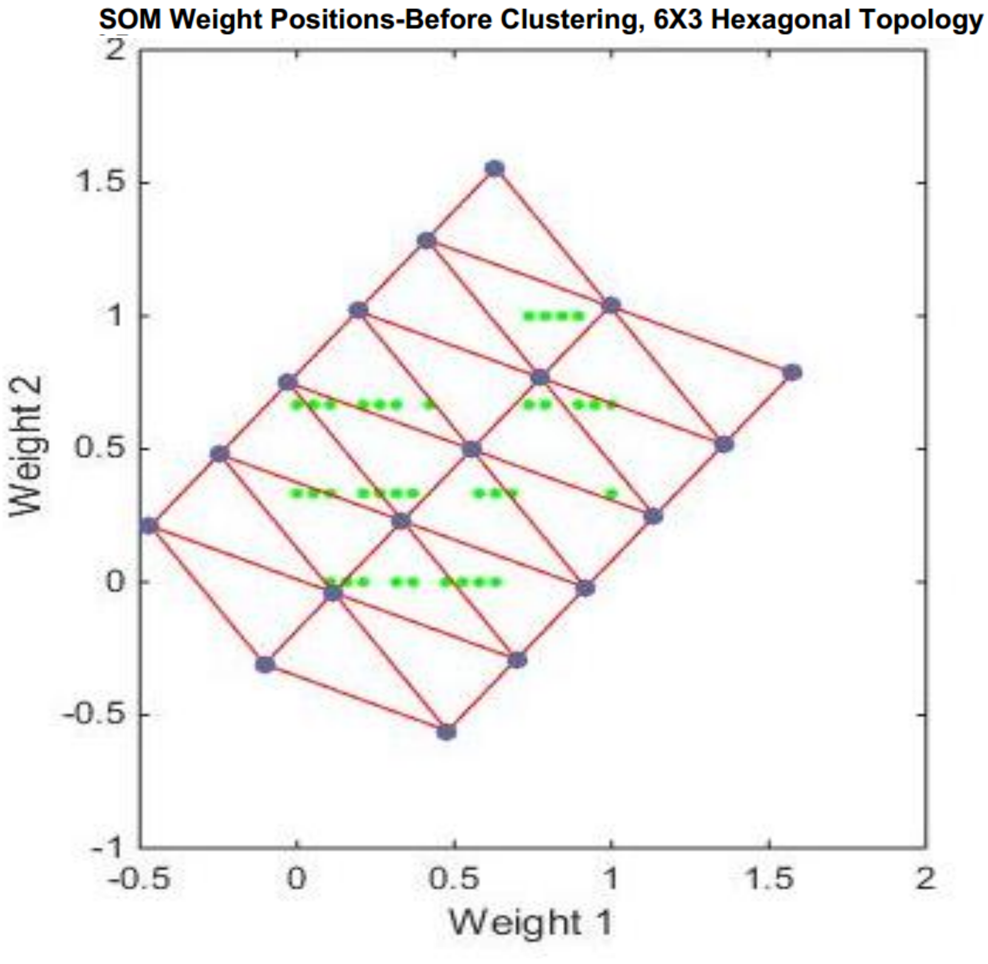

Figure 4.3: SOM before training

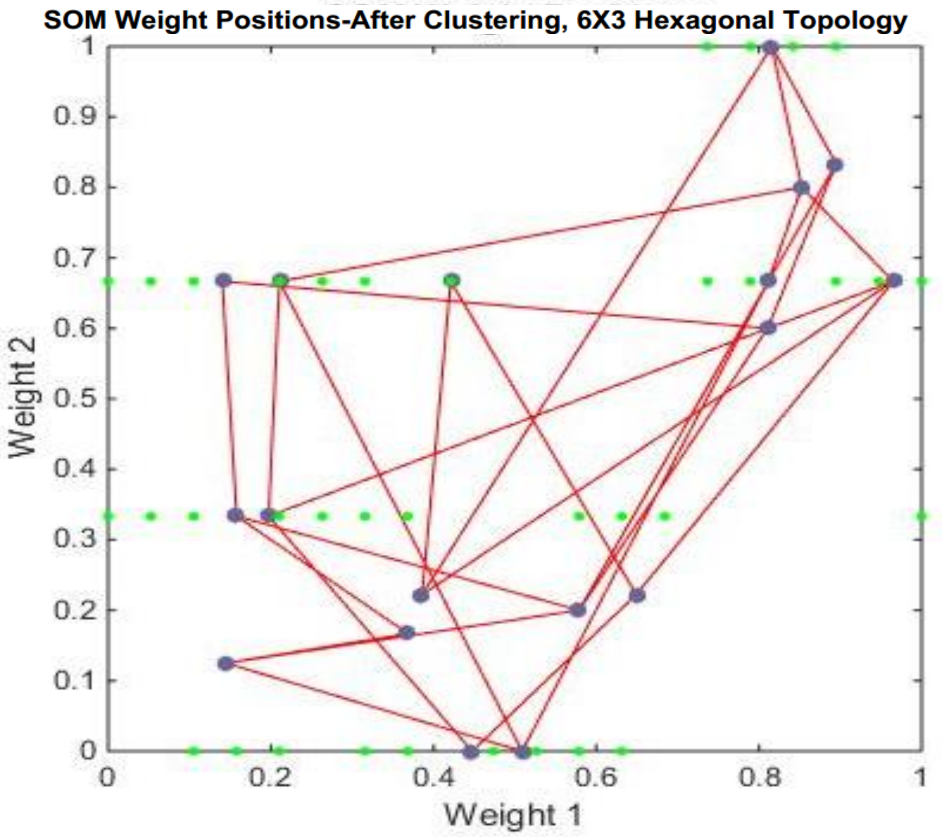

Figure 4.4: SOM after training

Additionally, the SOM Topology, the Distribution of clusters (classrooms), the SOM neighbor weight distance plot, and the weight input planes shown below are Matlab visualization tools which help in interpreting the data points after clustering.

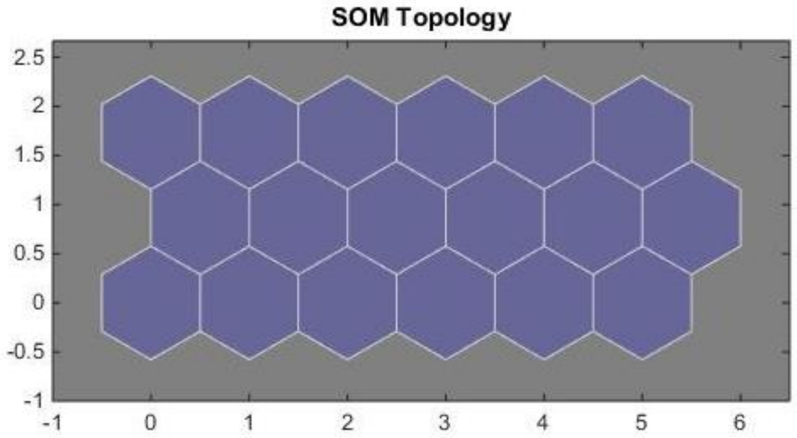

Figure 4.5: (6X3) 2D-hexagonal topology

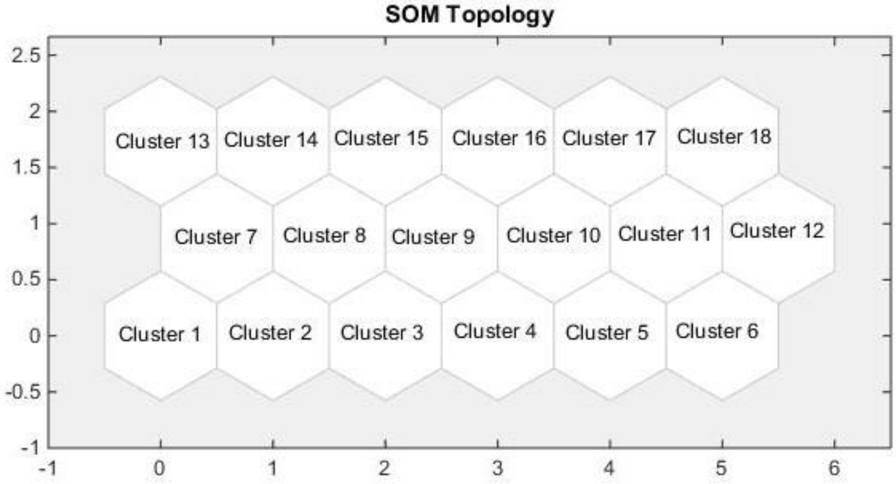

Figure 4.6: Clusters (Classrooms) distribution in (6X3) SOM topology

Firstly, the two figures (Figure 4.5 and Figure 4.6) above are self-explanatory. In contrast, in Figure 4.7 the SOM neighbor weight distances requires more interpretation. So, to interpret Figure 4.7 the following colors and description should be defined: 1. Neurons are represented by blue hexagons; 2. Red lines connect neighboring neurons; 3. Dark-colored regions represent larger distances between neurons; and 4. Light-colored regions represent smaller distances between neurons. It is clear according to the plot below, that the clusters, which are indicated as lighter colors, are distributed more consistently in most of the map. Yet, one or two clusters are presented with relatively large distances between the neighbors' weights, as indicated by the darker colors. Note that the neighbor weight distances plot is consistent with the position weight plot in Figure 4.4.

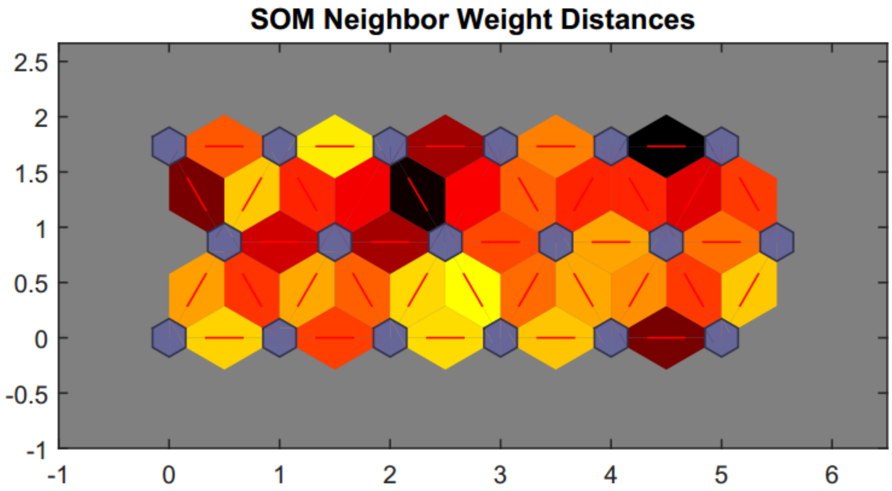

Figure 4.7: SOM neighbor weight distance for sections clustering

Moreover, the SOM weight plane plots in Fig. 4.8 are used to visualize the strength of weights that connect each input to each of the neurons (Robertson, 2014). For our experiment, five inputs were used; therefore, five subplots were generated for each input. The five input features included: the professors ID #, course level, course type, course name and course timing. This figure was generated after 100,000 iterations.

Lighter colors in the plots represent larger weights, whereas darker colors represent smaller weights. Similar connection patterns of the inputs indicate a high correlation. Inputs 3 and 5 appeared to be similar in some locations and were interpreted as highly correlated. Input from variables 1, 2, and 4 appeared to contribute the smallest amount of cluster separation in the data sets, as they appear to be the least similar and are less correlated. Although the SOM weight plane plot suggests a possible relationship might exist between the inputs, this concept does not pertain to our thesis, since it was pre-defined that the relationship between the inputs is independent.

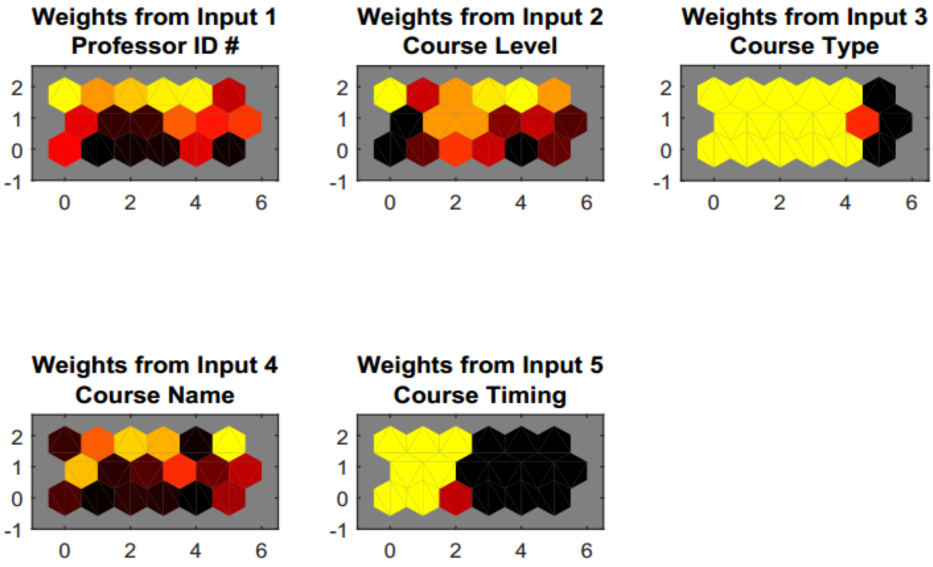

Figure 4.8: Weight input plane plot

In this thesis the analysis was done on each cluster independently, in order to extract the dominant features; this point will be elucidated at the next stage of analysis.

An additional useful visual plot provided by the MATLAB SOM function is the SOM sample hits plot seen in Figure 4.9. The sample hits plot counts the number of data points associated with each neuron. In an ideal situation, a relatively even distribution across the neurons is desired. However, the distribution was clustered

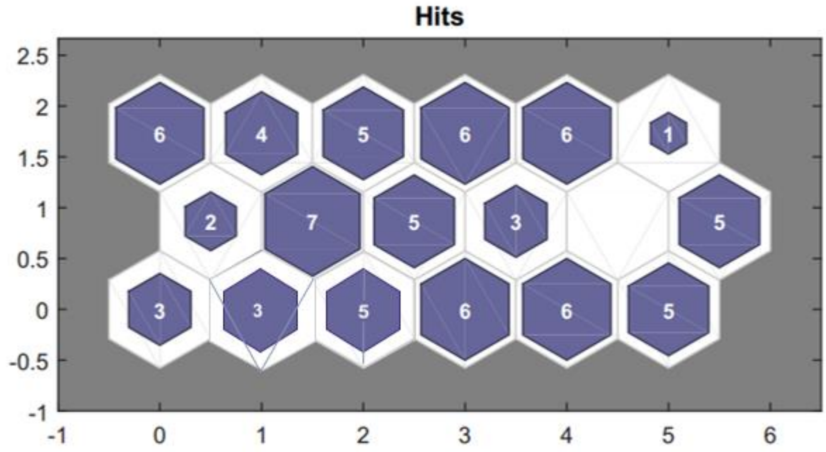

Figure 4.9: SOM sample hits- input points (sections) after clustering

unevenly throughout the map, which indicates that similar data were separated over different regions.

4.3.2 Classroom Scheduling Constraints Analysis

SOM First Stage

At this stage a deeper analysis is required to disclose the output of each cluster. Starting with cluster 1, which will be considered as classroom 1, the hits figure above indicates that cluster 1 consists of three sections and, according to the weight input plane plot below, input 3 and input 5 exert the most control on the data cloud points of classroom 1, with input 1 (Professor ID) also exerting a lesser degree of control. Hence, input 3, which is the course type, and input 5, which is the course timing, have the main effect on cluster 1 (Classroom 1), making them the dominant features.

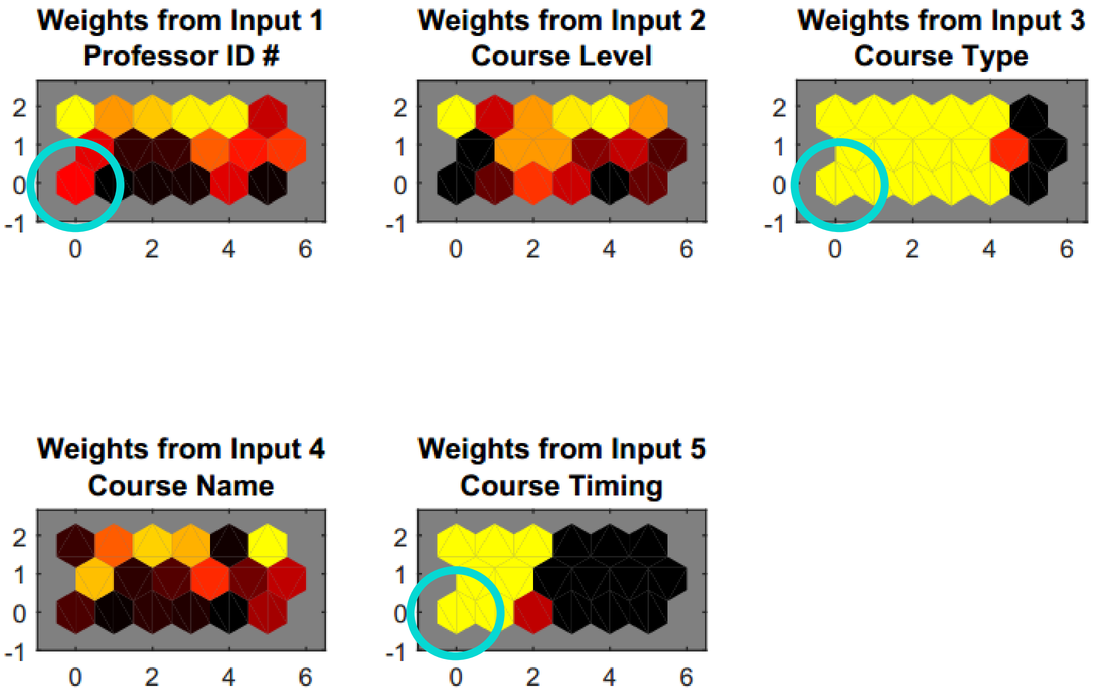

Figure 4.10: Weight input plane plot of cluster # 1

Furthermore, Table 5 and Table 6 illustrate the data before and after normalization for cluster 1 (Classroom 1), making it very easy to extract the dominant features. It is obvious that course level, course type and timing all have similarities across the three sections. Note that the existence of those three courses in the same group course level gaurantees a conflict-free schedule for third year students. Also, it appears that professor 11 is teaching two courses in classroom 1, which guarantees that there will not be a clash in timeslots for classroom 1 for these two sections.

Table 5: Cluster 1 (Classroom1) sections normalized

.Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
1	0.473684211	0	1	0.224719101	1
1	0.526315789	0	1	0.292134831	1
1	0.526315789	0	1	0.337078652	1

Table 6: Cluster 1 (Classroom 1) sections de-normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
1	10	3	1	25	1	Engineering Electromagnetics
1	11	3	1	31	1	Computer Programming
1	11	3	1	35	1	Digital Logic Design

A similar case is evident in cluster 2, as evidenced in Tables 7 and 8. These tables show that fourth year students will be able to take three courses in the same classroom at different times. Similar cases are repetitve in most of the clusters, so this observation can be considred as a point of strength in SOM, revealing that this is a good initial estimation for classroom scheduling.

Table 7: Cluster 2 (Classroom 2) sections normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
2	0.263157895	0.333333333	1	0.078651685	1
2	0	0.333333333	1	0.235955056	1
2	0.052631579	0.333333333	1	0.314606742	1

Table 8: Cluster 2 (Classroom 2) sections normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
2	1	4	1	26	1	Thermo-fluid System
2	2	4	1	33	1	Intro to CAM
2	6	4	1	12	1	Machine Design II

Also, it was noticed that the co-requisite sections were grouped in the same cluster, which is cluster 4 (classroom 4), thus it guarantees that students are able to register for two courses-- the co-requisite --with minimum conflict. For example, the table below shows the case of co-requisite courses MECH 409 (Dynamics System and Control) and MECH 417 (Kinematics), which are co-requisite courses for MECH 450 (System Dynamics Lab). Note, cluster 4 went through the second stage of SOM because it was overloaded with 6 sections; find the results of this in Table 9 below. Tables 10-13 show classroom #4 before and after separation and de-normalization.

Table 9: Cluster 4 (Classroom # 4) Overloaded with 6 sections before separation

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
4	0.10526316	0.3333333	1	0.04494382	0
4	0.10526316	0.3333333	1	0.134831461	0
4	0.2631579	0.3333333	1	0.02247191	0
4	0.31578947	0.3333333	1	0.134831461	0
4	0.36842105	0.3333333	1	0.235955056	0
4	0.21052632	0.3333333	1	0.314606742	0

Table 10: Cluster 4 (Classroom #4) part A after separation

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
4	0.10526316	0.3333333	1	0.04494382	0
4	0.10526316	0.3333333	1	0.134831461	0
4	0.2631579	0.3333333	1	0.02247191	0

Table 11: Cluster 4 (Classroom #4) part A de-normalized

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**	**Course Name (Letter)**
4	3	4	1	9	0	Dynamics system & control
4	3	4	1	17	0	Kinematics
4	6	4	1	7	0	Machine Design I

Table 12: Cluster 4 (Classroom #4) part B after separation

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**
4	0.315789474	0.333333333	1	0.134831461	0
4	0.368421053	0.333333333	1	0.235955056	0
4	0.210526316	0.333333333	1	0.314606742	0

Table 13: Cluster 4 (Classroom #4) part B de-normalized

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**	**Course Name (Letter)**
4	5	4	1	33	0	Intro to CAM
4	8	4	1	26	0	Thermo-fluid System
4	7	4	1	17	0	Kinematics

Another case that should be highlighted is found in clusters 6 and 12. Cluster 6 (classroom 6) and Cluster 12 (classroom 12) are distinct, as their course types are lab, not lecture. Tables 14, 15, 16 and 17 demonstrate the sections before and after normalization. The SOM could classify the labs and separate them from the lectures completely. Hence, laboratory rooms can be thought of as specialized equipment rooms with three consecutive hours sections; to prevent the interruption of laboratory work. Due to this, such sections are often conducted at late times during the day. Thus, this case helps alleviate the challenge often posed by the scheduling of laboratories with other academic classrooms. In addition, the two clusters 6 and 12 representing the labs in Mechanical Engineering and Electrical Engineering respectively, were distinguished by the SOM, allowing successful scheduling of the Mechanical labs and the Electrical labs.

Table 14: Cluster 6 (Classroom #6) normalized

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**
6	0	0.333333333	0	0.280898876	0
6	0.157894737	0	0	0.483146067	0
6	0.157894737	0	0	0.483146067	0
6	0.263157895	0.333333333	0	0.393258427	0
6	0.315789474	0.333333333	0	0.101123596	0

Table 15: Cluster 6 (Classroom #6) de-normalized

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**	**Course Name (Letter)**
6	1	4	0	30	0	Thermal Engineering lab
6	4	3	0	48	0	Fluid Mechanics lab
6	4	3	0	48	0	Fluid Mechanics lab
6	6	4	0	40	0	Design and Manufacturing Lab
6	7	4	0	50	0	System Dynamics lab

Table 16: Cluster 12 (Classroom #12) normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
12	0.473684211	0	0	0.056179775	0
12	0.578947368	0	0	0.449438202	0
12	0.578947368	0.333333333	0	0.314606742	0
12	0.631578947	0	0	0.786516854	0
12	0.684210526	0.333333333	0	0.629213483	0

Table 17: Cluster 12 (Classroom #12) de-normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
12	10	3	0	10	0	Electric Circuits I lab
12	12	3	0	45	0	Digital Logic Design Lab
12	12	4	0	33	0	Instrument and control lab
12	13	3	0	75	0	Electronic Circuits Lab
12	14	4	0	61	0	Microprocessors Lab

Moreover, an additional section--a lecture type course, but located in a lab--was added to measure the SOM clustering aptitude. In this case, the SOM succeeds in separating this point in one cluster only, cluster 18 (classroom 18). This is confirmation that SOM can differentiate easily between lectures and labs sections. See Tables 18 and 19.

Table 18: Cluster 18 (Classroom #18) normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
18	0.421052632	0.666666667	0	0.898876404	0

Table 19: Cluster 18 (Classroom #18) de-normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
18	9	5	0	85	0	graduation project I

To increase the flexibility of classroom scheduling and allow students to have more options to create their own schedules, multiple open sections of the same courses and different professors are created as a common scenario in university scheduling. This will give students more options to register in a course, thus it helps the student who has a conflicting morning session to register for an evening session or register for a specific course on a different day. Tables 20 and 21 show identical courses clustered in same classroom with two different professors, which guarantees that there will be no conflict between the two sections because logically two sections cannot be given at the same time in the same place. As a result, students will get the opportunity to select between the two timeslots. This case can be found also in clusters 14, 15, 16 and 17. See Appendix for more details.

Table 20: Cluster 10 (classroom #10) normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)
10	0.6315789	0	1	0.752808989	0
10	**0.6315789**	**0.3333333**	**1**	**0.516853933**	**0**
10	**0.6842105**	**0.3333333**	**1**	**0.516853933**	**0**

Table 21: Cluster 10 (Classroom #10) de-normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
10	13	3	1	72	0	Electro-Mechanical Devices
10	13	4	1	51	0	Microprocessors
10	14	4	1	51	0	Microprocessors

SOM Second Stage

Designing conflict-free scheduling is a challenging task, due to many variables and scenarios. The sections distribution over the class map (SOM) provides a partial solution. Although the section distribution is satisfying, some classrooms are overloaded with 6 sections and sometimes 7 sections in one day. This can be solved by a second stage of SOM clustering (refer to Chapter 3). For example, Tables 22, 23 and 24 show classroom 8 overloaded with 7 sections and the dominant features vary between course type, course level, course timing and professors' IDs.

Table 22: Cluster 8 (Classroom #8) before second stage SOM (separation)

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**
8	0	0.666666667	1	0.101123596	1
8	0	0.666666667	1	0.078651685	1
8	0.0526316	0.6666667	1	0.4044944	1
8	0.0526316	0.6666667	1	0.4157303	1
8	0.3157895	0.6666667	1	0.505618	1
8	0.4210526	0.6666667	1	0.1685393	1
8	0.4210526	0.6666667	1	0.1797753	1

Table 23: Cluster 8 (Classroom #8) after second stage SOM normalized

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**
8	0	0.666666667	1	0.101123596	1
8	0	0.666666667	1	0.078651685	1
8	0.0526316	0.6666667	1	0.4044944	1
8	0.0526316	0.6666667	1	0.4157303	1
Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**
8	0.3157895	0.6666667	1	0.505618	1
8	0.4210526	0.6666667	1	0.1685393	1
8	0.4210526	0.6666667	1	0.1797753	1

Table 24: Cluster 8 (Classroom #8) after second stage SOM de-normalized

Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**	**Course Name (Letter)**
8	1	5	1	14	1	Heat Engine
8	1	5	1	12	1	Intermediate heat Transfer
8	2	5	1	41	1	Non-Conventional Manufact.
8	2	5	1	42	1	Intro to Composites Design
Neuron # (Cluster #)	**Prof ID**	**Course Level**	**Course (Theory/ Lab)**	**Course Name**	**Time (AM/PM)**	**Course Name (Letter)**
8	7	5	1	31	1	Introduction to Robotics
8	9	5	1	20	1	Selected Topic in Bio.
8	9	5	1	21	1	Biomechanics

To provide further explanation, the overloaded classrooms cannot fit the given timeframe for each classroom in one day. Therefore, the suggested solution is to re-cluster the same section (going through second stage in SOM), which will then be split across two days. For example, the first new cluster will be on Sunday, and the second new cluster will be on Monday, and so forth. [Find all the overloaded clusters after second stage of SOM in Appendix] After splitting the overloaded cluster into two days, the professors IDs then become the dominant feature organizing the new clusters. This separation by professors IDs is useful as it avoids having the same professor teaching two sections in the same classroom at the same time. Consequently, the SOM was able to show a significant effect of Professor IDs feature through the second stage of separation.

So far, the SOM network was able to overcome the restrictions of course conflicts and classroom requirements. Additionally, it was noted that sections with same course level were clustered all together in the same groups, and the classrooms were separated to form rooms for lab sections and classrooms for regular lectures. This may form an initial estimation for classroom schedule.

In order to introduce the first draft of the schedule, a new constraint is added which are the detailed timeslots shown in Table 25. This will result in the Tables 26-32 below, which are considered an initial draft of randomly selected professors' timetables. Professor ID #1 was found in classroom2, classroom 6 and classroom 8 and professor ID # 4 was found in rooms 3 and 6. Table 22 and Table 24 display the first draft of the timetables and shows that there is no conflict in the common classroom 2.

Table 25: Detailed time slots per classroom

Classroom #	Timing
Slot #1	8- 10 AM
Slot # 2	10-12 AM
Slot #3	12-2 PM
Slot # 4	2-4 PM

Table 26: Professor ID #1 timetable

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
2	1	4	1	26	1	Thermo-fluid System
6	1	4	0	30	0	Thermal Engineering lab
8	1	5	1	14	1	Heat Engine
8	1	5	1	12	1	Intermediate heat Transfer

Table 27: First draft of Professor ID #1 timetable

Professor ID # 1				
Day/Time	.8-10	.10-12	.12-2	.2-4
Sunday	Thermo-fluid System (classroom # 2)		Thermal Engineering lab (Classroom # 6)	
Monday	Heat Engine (Classroom #8)	Intermediate heat Transfer (Classroom #8)		
Tuesday	Thermo-fluid System (classroom # 2)			
Wednesday	Heat Engine (Classroom #8)	Intermediate heat Transfer (Classroom #8)		

Table 28: Professor ID #2 timetable

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
2	2	4	1	33	1	Intro to CAM
8	2	5	1	41	1	Non-Conventional Mnaufact
8	2	5	1	42	1	Intro to Composites Design
9	2	5	1	40	0	Selected topics in Manufact.

Table 29: First draft of Professor ID #2 timetable

Professor ID # 2				
Day/Time	.8-10	.10-12	.12-2	.2-4
Sunday	Non-Conventional Mnaufact (classroom # 8)	Intro to CAM (classroom#2)	Selected topics in Manufact. (Classroom # 9)	
Monday			Intro to Composites Design (Classroom #8)	
Tuesday	Non-Conventional Mnaufact (classroom # 8)	Intro to CAM (classroom#2)	Selected topics in Manufact. (Classroom # 9)	
Wednesday			Intro to Composites Design (Classroom #8)	

Table 30: Professor ID #6 timetable

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
0.058824	0.631579	0	1	0.752809	1	Intro to CAM
0.176471	0.631579	0.333333	1	0.516854	1	Machine Design I
0.176471	0.684211	0.333333	1	0.516854	0	Machine Design I
9	6	5	1	45	1	Maintenance Engineering

Table 31: First draft of Professor ID #6timetable

Professor ID # 6				
Day/Time	.8-10	.10-12	.12-2	.2-4
Sunday	Machine Design I (classroom # 4)	Maintenance Engineering (Classroom # 9)		
Monday	Machine Design II (classroom#2)		Design and Manufacturing Lab (Classroom #6)	
Tuesday	Machine Design I (classroom # 4)	Maintenance Engineering (Classroom # 9)		
Wednesday	Machine Design II (classroom#2)			

Table 32: Cluster 2 (Classroom #2) sections normalized

Neuron # (Cluster #)	Prof ID	Course Level	Course (Theory/ Lab)	Course Name	Time (AM/PM)	Course Name (Letter)
2	1	4	4	26	1	Thermo-Fluid System
2	2	4	1	33	1	Intro to CAM
2	6	4	1	12	1	Machine Design II

As the tables above reveal, it is possible to generate via SOM first draft schedules for each classroom, though finalization of individual professor timetables are still best adjusted manually proceeding the first SOM draft. Periodically, the SOM first draft places the same professor within the same classroom for all of his or her sections, however at other times, this must be adjusted for manually. Not all professors are able to have all of their courses in the same classroom though due to variation in SOM prioritization. The SOM may at times prioritize other variables over professors ID, such as course type, course level, and timing. This shows that the SOM is able to accommodate an additional variable, which in this case are the time slots.

At this stage, a conflict-free schedule has been constructed by the SOM. However, to fit newly created classrooms into the SOM generated timetable, the output from SOM is used to train a Feedforward Back Propagation (FFBP) neural network to extract the implicit course-classroom mapping as formulated by the SOM.

4.4 Mathematical Modeling of FFBP NN for classroom scheduling

The output of the SOM NN which is the classroom number for each section is considered as the input for FFBP NN. Each input vector has 5 elements (constraints) assigned to a specific room as previously discussed. The structure below shows the back-propagation neural network model for our data. The weight vectors are randomly initiated at the beginning. Note that two hidden layers were used; the first hidden layer carries 20 neurons and the second hidden layer carries 30 neurons. The structure below shows the details FFBP NN map as well as and the weight matrix size for the first hidden layer is (5 X 30) and the second size is (30X20)

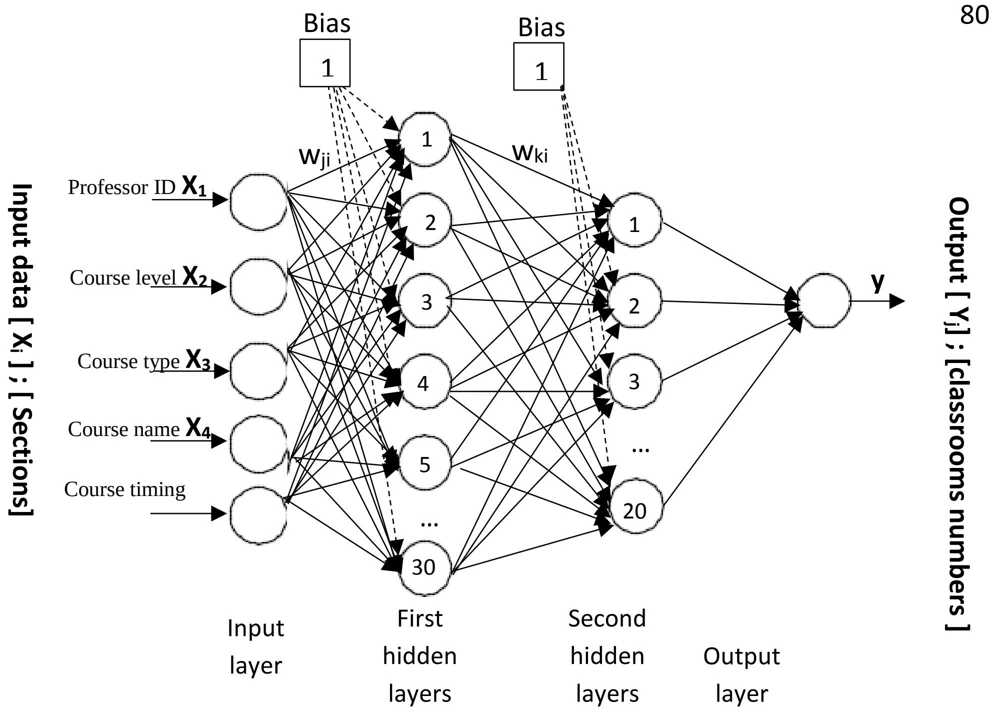

Figure 4.11: Backpropagation neural network with two hidden layers

4.5 Back-Propagation Parameters for Tests in MATLAB

BPFF NN codes were developed by using two hidden layers. Regardless of the number of hidden layers, the same structure and assumptions can be used to construct any code. The description of the main coding points is as follows:

Initiating Weights: The weights for each hidden layer were generated by assigning a random number between -0.5 and 0.5. The size of the weight's matrix is flexible based on the total number of the used input data and the number of neurons in the hidden layers. A bias layer of value 1 is included automatically to the weight matrix.

Initiating learning rate: The learning rate is responsible for the rate at which each single weight is modified after one learning cycle. The learning rate is usually between 0 and 1, and the closer it is to 0, the smaller the steps needed to modify each weight. The best learning rate for the used network was determined through trial and error which is 0.1, this results was found experimentally.

Modifying the input parameters: As previously mentioned, the output of SOM is the input to FFBPNN. Hence, the classroom numbers need to be normalized before entering the FFBPNN to avoid big differences between numbers as explained previously in Chapter 2.

Log-sigmoid transfer function: As the input vectors (sections) have non-linear relationship between its inputs, a non-linear log-sigmoid transfer function was used in this code. In this function the resulting values from multiplication of the inputs and weights fall between 0 and 1. The result of this function is processed as an input to the next layer, or as a final result in the case of the final layer.

No. of Epochs: For the set of 64 data points (70% of the total number of the data sets) with a batch size of 5, each iteration processes 5 input vectors for a total of 16 such iterations to create an entire set. Each set is called one epoch, which helps to direct the convergence. However, there is also a premature termination criterion depending on the mean squared error, which was set as 0.01. Based on this, the number of epochs in this test is 746626 after the convergence.

4.6 FFBP NN Results and Discussion

4.6.1 Learning Phase:

In this phase the neural network is carried out using the 64 data sets. As described earlier, the training parameters that were set continue running until it reaches below 0.1. Various network configurations were tested during the training phase with two different values of learning rates, 0.05 and 0.1. The back propagation learning activity is time consuming, due to the fact that after each learning cycle the network sends back the weight changes to every single weight in the system for the two hidden layers. Additionally, because the used learning rate is relatively small, the modifications to the weights will be small and thus it will take more learning cycles and more time to change all the weights to reach to the optimal solution. The average time required for the two hidden layers to be trained is 1:30 hours. Figure 4.12 illustrates the training results for two hidden layer networks.

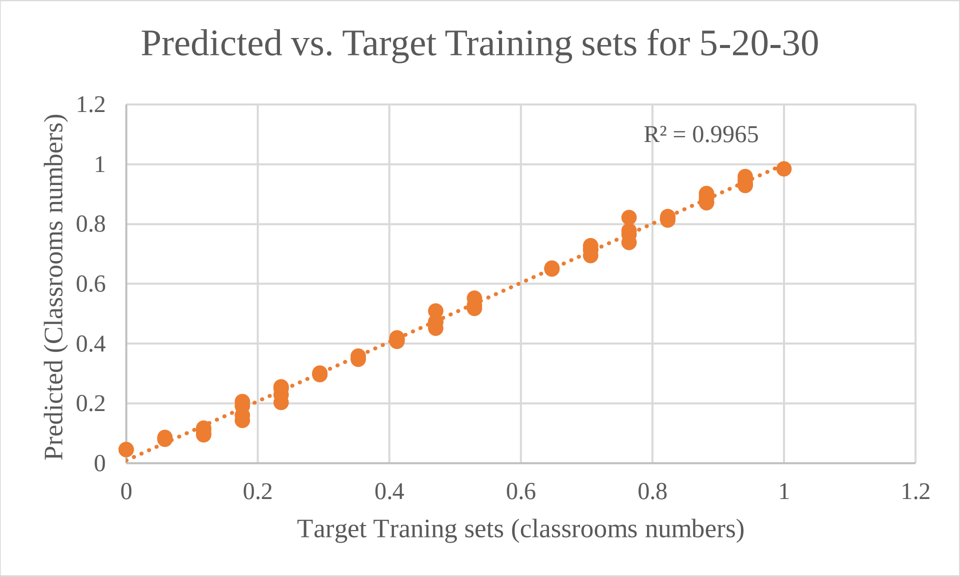

Figure 4.12: Predicted vs. target training sets

4.6.2 Testing and Validating Phase

The testing and validation phase must be done by using an independent test set. Hence, the independent test set is a set similar to the input set, but not a part of the training set. In our case 20% of the data set was used for testing and validating the network (15 untrained data points). The testing was done by using the acquired weights from the trained network in calculating the classrooms numbers for the 15 remaining untrained data sets. In the calculations of the classrooms numbers, the inputs were multiplied by the weights resulting from the training sets, and the results were then activated through the log-sigmoid function and proceeded to the next layer. The final results were compared to the tested sets. The accuracy of the predicted results was measured through the coefficient of determination R^2, where the maximum R^2 reached was 0.9814 for different sets of neurons each layer. See Figure 4.13.

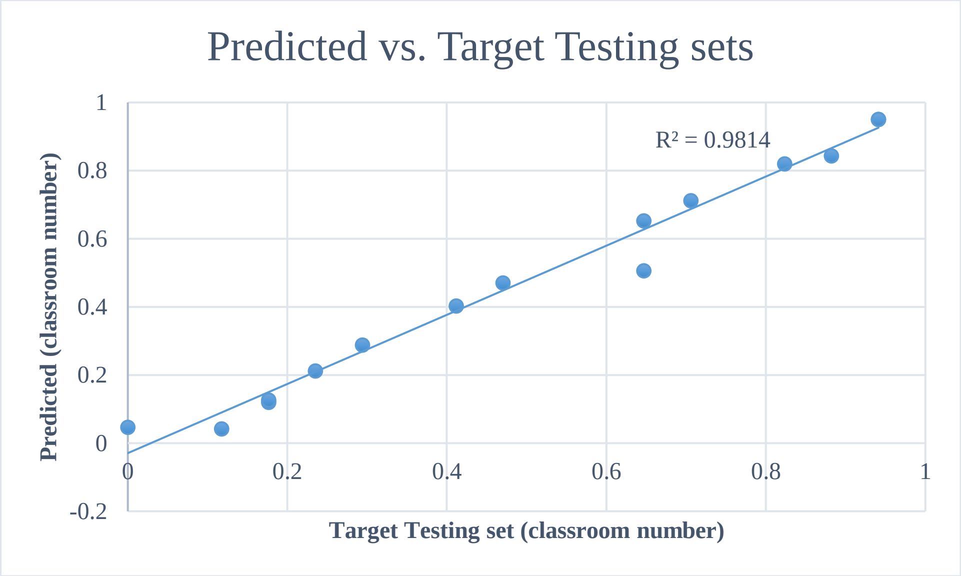

Figure 4.13: Predicted vs. target testing sets

4.6.3 Results and Discussion

Different types of network architectures were tested to find the optimal convergence. It was shown that the two layer configuration system with 30 neurons in the first layer and 20 neurons in the second layer with a learning rate of 0.1 gives the best convergence with R^2 of 0.94. The results showed that the optimal case required a total of 663 weights with a 0.1 learning rate to have a higher convergence. Decreasing the learning rate to 0.05 caused a negative impact on the accuracy, which is referred to as over-fitting. The increase of the network size also causes a reduction in the accuracy of the prediction due to the increase of the total weights that need to be modified. As a result, a number of modeling parameters were selected depending on the forecast horizon and degree of accuracy. Figure 4.14 shows high accuracy in the prediction results for training and test sets by FFBP network 5-30-20.

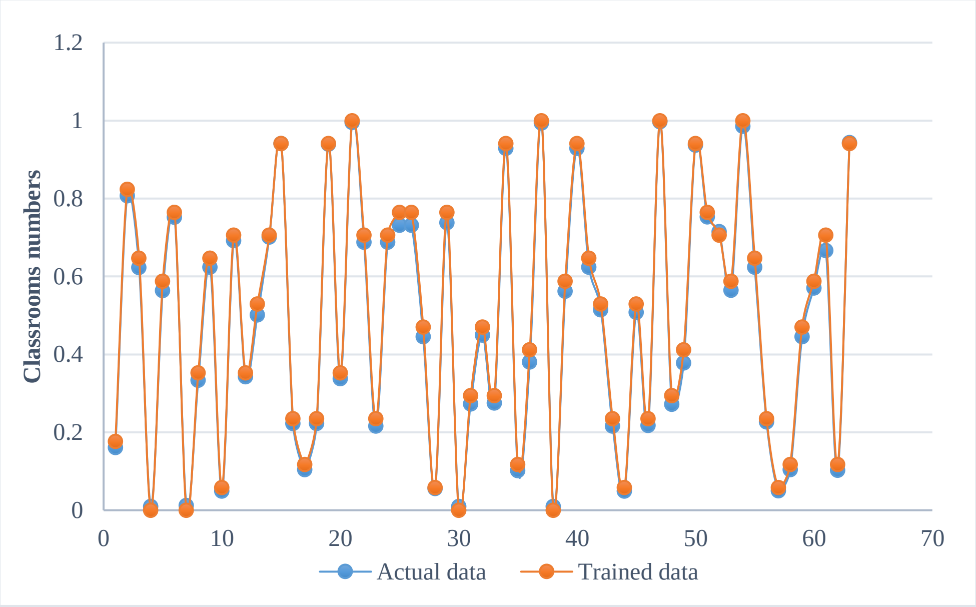

Figure 4.14: Prediction results for training and test set by FFBP network 5-30-20

The tests reveal that for accommodating a new section, FFBP NN is capable of fitting the new section into an existing classroom. The scenario of opening a new section after all the professors' timetables have been set or after one or two weeks have elapsed from the beginning of the semester is very common situation, occurring often in many institute for innumerable reasons. Therefore, re-clustering the whole set of data to fit the newly created classroom is not a practical solution because each SOM run can propose a new set of clusters, which will completely change the schedule for each classroom. As a result, the FFBP network will help to solve the problem by allocating the new section to fit in a suitable classroom without causing any conflict in the schedule.

4.7 Discussion in Summary

Based on the results of this thesis research, it can be concluded that the SOM is able to cluster the given sections and provide a good initial estimation for classroom scheduling. At first examination, the clusters show that the dominant features are course type and course timing. This is a benefit of the SOM, as grouping course levels in the same classroom provides the opportunity for students of the same year level to register in the needed sections without facing any conflicts (i.e., in the case of co-requisite courses).

However, each cluster presents an individual case, as seen in the example of the labs. Due to this common feature, the network was able to clearly distinguish these sections and group them into one region. To measure the accuracy of the network, an anomaly was added. The anomaly was a lecture class MECH 585 (Graduation Project-I) that, on this occasion, needed to be taught in a laboratory. The result was that the network separated the section into a different cluster, placing it into a completely unique section. Furthermore, a second stage of SOM was used to separate the overloaded clusters, resulting in two new clusters which are split into two days.

To further test the efficacy of this system, a new constraint--detailed timeslots--was added to create a first draft of a timetable for random professors. This was done to prove that it is possible to generate first draft classroom schedules via the SOM. Although the timetables were finalized manually, the first draft SOM results still were able to create clustered sections for each professor that revealed a conflict free timetable.

Lastly, a new scenario was put forward, which was fitting a newly opened section into an existing classroom without the need of re-cluster all the sections, which would result in a completely new classroom schedule. After trial, it was found that the FFBP NN was able to allocate the new section in the proper classroom that carries same features as the new section without changing the rest of the pre-existing room allocations.

Chapter 5: Conclusion and Future Work

This thesis has examined a new approach to solve university classroom scheduling problems using an artificial intelligence technique. Classroom scheduling is a very complex task due to many parameters and frequent changes in requirements. The proposed methodology is divided into two main phases. The first phase uses a Self-Organizing Feature Map (SOM) to cluster the generated input patterns, which consists of five different constraints. This set of data points were generated from United Arab Emirates University (UAEU) study plans for Mechanical and Electrical Engineering courses. The second phase uses a back-propagation algorithm to modify the SOM-generated timetable in order to accommodate newly created sections without requiring a complete change of the existing schedule.

In the first phase of this research, 78 data points (sections) were used as input vectors to SOM network. The output results with clustered sections assigned to each neuron (classroom) carry similar features, which are considered as a mark for conflict. In the case of the SOM producing an overloaded classroom, additional clustering was done to remove the overloaded sections and separate them to two days or two different classrooms. The second phase of scheduling occurs when the first draft schedule needs to be modified. For example, if a new section is opened after the initial schedule is created. In these cases, the Back-propagation neural network is used to fit the new section into the created timetable.

The Matlab software was used to write and run the networks. All code was written using the command lines in Matlab. Although Matlab has a Neural Network tool box (nntool), it was not easy to manipulate and test with the parameters of the network. That is why the command lines were used.

After several runs and fine-tunings of the networks, the results reveal that the proposed model can create an initial guess of a valid classroom schedule. The SOM NN was able to cluster the sections according to their similarities, which revealed the areas of conflict. For example, the SOM NN split the lab sections and the lecture sections, demonstrating that it was able to identify that the lab sections needed to be treated carefully when it comes to scheduling. A similar case found in this investigation is that the SOM also separated the course levels into groups, which highlights the SOM's recognition of similarities within these sections. The benefit of this distinction is that this grouping of sections by level prevents students from experiencing delays in their study plans. Further, it was noticed that the SOM grouped the professors who teach the same sections together, which also emphasized the fact that SOM was able to show the regions where scheduling may have conflicts. Also, in many cases the SOM was able to prioritize the features in such a way that the strongest (dominant) feature took the lead and had the most significant effect on a specific group of sections, i.e. in the case of scheduling lab sections. Additionally, when a classroom became overloaded, a further splitting was done to overcome this issue. After this, the resultant from the splitting separates the scheduled sections into two different days in the week, thus alleviating the overload in the particular classroom.

To further test the efficacy of this system, a new constraint--detailed timeslots--was added to create a first draft of a timetable for random professors. This was done manually to prove that it is possible to generate conflict-free first draft classroom schedules.

Another neuron network was used to modify the produced classroom scheduling without the need to change the whole content of the already existing schedule. This neural network is FFBP NN. As the FFBP does not stop learning, but instead continues to adapt to changing inputs, this allows the network to adjust to unexpected environmental changes, such as fitting a newly opened section into a pre-existing schedule.

The findings above show that when a proposed conflict occurred in professor ID, time restrictions, course conflict and classroom requirements, the system was capable of finding a solution. The proposed model enables the easy generation of conflict-free classroom timetables and it is predicted that the procedure can be extended and implemented in fields other than academia such as factories, healthcare, and transportation. The successes of using the artificial intelligence approach for classroom scheduling proves that the concepts in this research are valid.

Recommendations for further study: A comparative study is recommended to justify the superiority of this approach to other heuristic-based or mathematical-based models available in the literature. Additional features and analysis are recommended to investigate the differences in the SOM and to perform better clustering. Also, more tests and applications of this model need to be implemented to further prove its efficacy. . This model (artificial intelligence approach) has particular application for UAE, as it has potential to benefit the growing number of industries within the UAE, such as the healthcare field and transportation industry, along with many others.

References

Abramson, J. A. (1992). A Paralle Genetic Algorithm for Solving The School Timetabling Problem. *Australian Computer Science Conference*, (pp. 1-11). Hobart.

Abuhamdah, A., Ayob, M., & Kendall, G. (2013). Population based Local Search for university course timetabling. *Springer Science+Business Media*, Vol. 40, 44-53.

Aloul, A. W. (2007). Solving the University Class Scheduling Problem Using Advanced ILP Techniques. *Department of Computer Engineering, American University of Sharjah (AUS)*, 87-93.

Ansari, A. (2014). Genetic Algorithm to Generate the Automatic Time-Table – An Over View. *International Journal on Recent and Innovation Trends in Computing Communication*, Vol 2, 3480-3483.

Azimi, Z. N. (2005). Hybrid heuristics for Examination Timetabling problem. *Applied Mathematics and Computation*, Vol. 163, 705–733.

Basir, N., Ismail, W., & Norwawi, N. M. (2013). A Simulated Annealing for Tahmidi Course Timetabling. *The 4th International Conference on Electrical Engineering and Informatics (ICEEI 2013)* (pp. 437 – 445). Negeri Sembilan, Malaysia: Elsevier.

Birattari, M. D. (2011, May 16). *Encyclopedia of Machine Learning*. Retrieved from Springer: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_22

Borglin, J. (2011). *Classification of hand movments uring multi-channel EMG.* Gothenburg, Sweden: Chalmers University of Technology.

Carrasco, M. P. & Pato, M. (2001). A Pott Neural Network Heuristic for the class/Teacher Timetabling Problem. *4th Metaheuristics International Conference*, (pp. 139-142). Portugal.

Colorni, D. M. (1998). Metaheuristics for High School Timetabling. *Computational Optimization and Applications, Kluwer Academic*, Vol. 9, 275-298,.

Deris, S. S. (1999). Incorporating constraint propagation in genetic algorithm for university timetable planning. *Engineering Applications of Artificial Intelligence*, Volume 12, 241-253.

Dimopoulou, P. M. (2004). An automated university course timetabling system developed in a distributed environment: A case study. *European Journal of Operational Research*, Vol. 153, 136-147.

Edmund, K. B. (2006). *Case-based heuristic selection for timetabling problems.* Springer Science + Business Media, LLC 2006.

Fisher, A. F. (2006). Chapter 12 – Towards Automatic Risk Analysis for Hereditary Non-Polyposis Colorectal Cancer Based on Pedigree Data. In A. F. Fisher, *Outcome Prediction in Cancer* (pp. 319–337). Liverpool: Elsevier.

Glassey, C. R., & Mizrach, M. (1986). A decision support system for assigning classes to rooms. *Informs*, Vol.10, 92-100.

Glover, F. (2018, February 18). *Tabu search*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Tabu_search

Gonzalez, T. F. (2007). *Handbook of approximation algorithms and metaheuristics.* Taylor & Francis.

Gotlieb, C. (1963). The construction of class-teacher timetables. *IFIP congress 62* (p. 73). Popplewell editor.

Guthikonda, S. M. (2005). *Kohonen Self-Organizing Maps.* Wittenberg University.

Haykin, S. (2009). *Neural Network and Learning Machines* . Ontario, Canada: PEARSON.

Hwang, C. K. (1989). A Knowledge Base Approach to class scheduling problem with a developed system ACS. *Computer Society Press*, 658-663.

Juha, V. J. (1999). Self-organizing map in Matlab: the SOM Toolbox. *Proceeding of Matlab* (pp. 35-40). Finland: Laboratory of Computer and Information Science, Helsinki University of Technology,.

Khader, A. T. (1994). A knowledge based approach to class based school timetabling. *UMI* .

Lek, S., & Guégan, J. F. (1999). Artificial neural networks as a tool in ecological modelling, an introduction. *El Sevier* , Vol. 120, 65-73.

Li, J., Cheng, J., Shi, J.-y., & Huang, F. (2012). *Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement.* Changchun China: Aviation University of Air Force.

Liebowitz, J. K. (1998). Classroom scheduling: An application and tools. *The Journal of Computer Information Systems*, Vol. 38, Iss. 3.

López, R. E. (2014). Grid topologies for the self-organizing map. *Elsevier*, Vol. 56, 35-48.

Mahmud, A. (2014). ACO with GA Operators for Solving University. *3rd International conference on informatics, electronics & vision 2014* (pp. 1-16). Dhaka: IEEE.

Martinsons, M. G., & Kong, C. K. (1993). Intelligent timetabling using a microcomputer. *The International Journal of Educational Management*, Vol. 7, 1-12.

Mathworks. (1994-2018, January Sunday). *Matlab*. Retrieved from www.mathworks.com: https://www.mathworks.com/products/matlab.html

Mendelsohn, L. (2018, January 16). Preprocessing Data for Neural Networks. Retrieved from Vantagepointsoftware: https://www.vantagepointsoftware.com/mendelsohn/preprocessing-data-neural-networ

Nicholas, N. K. (2012). *Forecasting of wind speeds and directions with artificial neural networks.* Lappeenranta, Fenland: Lappeenranta University of Technology .

Philip, S. N. (2001). *Studies in Artificial Neural Network Modeling.* Kochi- India: Cochin University of Science and Technology.

Pillay, N. (2010). *An Overview of School Timetabling Research.* KwaZulu-Natal: School of Computer Science, University of KwaZulu-Natal.

Qu, E. K. (2006). Case-based heuristic selection for timetabling problems. *Springer Science + Business Media*, Vol. 9, 115-132.

Reynolds, R. (1994). *An introduction to cultural algorithms.* Michigan.

Robertson, K. B. (2014). *Knowledge Discovery and Information Extraction on the Open Internet Using MATLAB and Amazon Web Services.* INTECH.

Rojas, R. (1996). *Neural Networks.* Berlin: Springer.

Sayers, C. (1991). *Self Organizing Feature Maps and Their Applications to Robotics.* Pennsylvan: University of Pennsylvan Scholarly Commons.

Schaerf, A. (1999). A Survey of Automated Timetabling. *Kluwer Academic Publishers*, Vol. 13, 87–127.

Singupta, I. I. (2009,September 22). Neural Network and Applications . Retrieved from www.youtube.com:

https://www.youtube.com/watch?v=xbYgKoG4x2g&list=PL3EA65335EAC29EE8

Smith, A. K., Abramson, D., & Duke, D. (2003). Hopfield neuralnetwork for timetabling: formulations methods and compatitive results. *Computer and Industrial Engineering*, Vol. 44, 283-305.

Sprain, L., Endres, D., & Rai-Peterson, T. (2010). Research as a transdisciplinary network process. *Communication Monograph, 77*(4), Vol.77, 441- 444 .

Stattrek, L.M. (2017, October 10). *stat trek*. Retrieved from www.stattrek.com: http://stattrek.com/statistics/dictionary.aspx?definition=z_score

Taborda, A. W. (2004). *Neural Networks Applied on Educational Timetabling Problems: an Overview.* Porto Velho: Federal University of Rondônia.

Teoh, C. K., & Wibowo, A. (2013). Review of state of the art for metaheuristic techniques. *Springer Science+Business Media Dordrecht*, Vol 44, 1-21.

Vishwanathan, A. S. (2008). *Introduction to machine learning.* Cambridge, United Kingdom: the press syndicate of the university of cambridge.

Wren, A. (1995). A genetic algorithm for public transport driver scheduling. In A. Wren, *Computers & Operations Research* (pp. 101-110). Berlin: Springer.

Yousef, B. F. (2001). *Neural network approach to modeling the laser material-removal process.* London, Ontario, Canada.: University of Western Ontario, .

Yu, T. (1990). *Time-table scheduling using Neural Ketwork Algorithms.* California : California State University at San Bernardino,.

Zhipeng, J. K. (2010). Adaptive Tabu Search for course timetabling. *European Journal of Operational Research*, Vol. 200, 235–244.

List of Publications

Basem F. Yousef and Farah Aiash “Mechanism for surgical Tool Manipulation”, 9th IEEE Asian Control Conference, (ASCC 2013) Istanbul-Turkey, pp.2713-2718, June 2013.

Appendix

Table 33: Raw data with constraints (1)

	Four lectures each room				Four lectures each room				Four lectures each room				Four lectures each room			
	Original data															
Prof ID	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
Course Level	3	4	4	3	3	4	5	5	3	4	6	4	3	4	5	5
Course (Theory/ Lab)	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
course name	48	29	30	11	6	33	40	42	10	50	15	17	11	29	31	30
Time (AM/PM)	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1
Co-course	0	8.7	9	0	0	0	0	0	2	10	0	0	0	0	0	0
Course Name (letters)	Fluid Mechanic lab	Thermo-fluid System	Thermal Eng. lab	Applied Thermo.	Manuf. process	Intro. to CAM	Selected topics in Manufact.	Intro to Com. Design	Dynamic	System Dynamic lab	Adv. Dynamic and Control	Kinematic	Applied Thermo	Thermo. System	Introd. to Robotics	SEL Topics in Mechatronic

Table 34: Raw data with constraints (2)

Four lectures each room				Four lectures each room				Four lectures each room				Four lectures each room			
Original data															
5	5	5	5	6	6	6	6	7	7	7	7	8	8	8	8
4	5	3	3	4	3	4	4	4	4	3	4	4	3	4	3
1	1	1	390	1	1	1	1	1	0	1	1	1	1	0	0
33	42	6	90	7	6	12	40	17	50	10	7	29	11	30	48
0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	10	0	0	8.7	0	9	0
Intro to CAM	Intro to Composites Design	Manuf. process	Engineering Materials	Machine Design I	Manuf. process	Machine Design II	Design and Manufacturing Lab	Kinematics	System Dynamics lab	Dynamics	Machine Design I	Thermo. System	Applied Thermo	Thermal Engineering lab	Fluid Mechanics lab

Table 35: Raw data with constraints (3)

Four lectures each room				Four lectures each room				Four lectures each room				Four lectures each room			
Original data															
9	9	9	9	10	10	10	10	11	11	11	11	12	12	12	12
5	5	5	3	5	3	3	3	3	3	3	3	3	3	3	3
1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
20	23	42	6	23	5	10	20	25	30	35	20	45	36	35	70
0	0	1	0	0	1	0	1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Selected Topic in Bio.	Biomaterials	Intro to Composites Design	Manuf. process	Biomaterials	Electric Circuits I	Electric Circuits I lab	Electric Circuits II	Engineering Electromagnetics	Computer Programming	Digital Logic Design	Electric Circuits II	Digital Logic Design Lab	Signals & Systems	Digital Logic Design	Electronic Circuits

Table 36: Raw data with constraints (4)

Four lectures each room				Four lectures each room				Four lectures each room				Four lectures each room			
Original data															
13	13	13	13	14	14	14	14	15	15	15	15	16	16	16	16
3	3	4	4	4	4	4	4	5	5	5	6	5	5	6	6
1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1
72	75	51	61	51	61	62	72	62	82	85	0	82	85	18	25
1	1	1	0	1	0	1	1	0	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Electro-Mechanical Devices	Electro. Circuits Lab	Microproces	Microproces Lab	Microproces	Microproces Lab	Comp. Arch. & Organizat	Power System	Embed System Design	Ang. Integ. Cir.	Grad. Project I	Numerical Methods in Eng.	Ang. Integ. Cir.	Grad. Project I	Microwave Eng.	Power System Quality

Table 37: Raw data with constraints (5)

Four lectures each room				Four lectures each room				Four lectures each room				Four lectures each room			
Original data															
17	17	17	17	18	18	18	18	19	19	19	19	20	20	20	20
6	6	6	6	5	6	5	5	6	6	5	6	6	4	4	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
37	41	18	25	85	0	82	85	18	25	85	18	94	62	72	62
1	1	0	0	1	0	1	1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sensors Design and App.	Contemp. Digital Systems	Microwave Eng.	Power Sys. Qu.	Grad. Project I	Numerical Methods in Eng.	Analog Integ. Circuit Dg.	Grad. Project I	Microwave Eng.	Power System Quality	Grad. Project I	Microwave Eng.	Research/Design paper	Comp. Arch. & Organizat	Power System	Embed System Dg.

Table 38: Input data after clustering by SOM - normalized

Neuron Number	Professor ID	Course level	Course Type	Course name	Timing
1	0.473684	0	1	0.224719	1
1	0.526316	0	1	0.292135	1
1	0.526316	0	1	0.337079	1
2	0	0.333333	1	0.235955	1
2	0.052632	0.333333	1	0.314607	1
2	0.263158	0.333333	1	0.078652	1
3	0.105263	0	1	0.05618	1
3	0.157895	0	1	0.067416	1
3	0.157895	0	1	0.393258	1
3	0.210526	0	1	0.011236	1
3	0.210526	0	1	0	1
4	0.105263	0.333333	1	0.044944	0
4	0.105263	0.333333	1	0.134831	0
4	0.210526	0.333333	1	0.314607	0
4	0.263158	0.333333	1	0.022472	0
4	0.315789	0.333333	1	0.134831	0
4	0.368421	0.333333	1	0.235955	0
5	0.315789	0	1	0.044944	0
5	0.368421	0	1	0.067416	0
5	0.473684	0	1	0.168539	0
5	0.473684	0	1	0.168539	0
5	0.526316	0	1	0.157303	0
5	0.526316	0	1	0.348315	0
6	0	0.333333	0	0.280899	0
6	0.157895	0	0	0.483146	0
6	0.157895	0	0	0.483146	0
6	0.263158	0.333333	0	0.393258	0
6	0.315789	0.333333	0	0.101124	0
7	0.368421	0	1	0.88764	1
7	0.578947	0	1	0.730337	1
8	0	0.666667	1	0.101124	1
8	0	0.666667	1	0.078652	1
8	0.052632	0.666667	1	0.404494	1
8	0.052632	0.666667	1	0.41573	1
8	0.315789	0.666667	1	0.505618	1
8	0.421053	0.666667	1	0.168539	1
8	0.421053	0.666667	1	0.179775	1
9	0.052632	0.666667	1	0.393258	0
9	0.105263	0.666667	1	0.011236	0
9	0.210526	0.666667	1	0.41573	0

Table 38: Input data after clustering by SOM – normalized (Continued)

Neuron Number	Professor ID	Course level	Course Type	Course name	Timing
9	0.263158	0.666667	1	0.449438	0
9	0.421053	0.666667	1	0.202247	0
10	0.631579	0	1	0.752809	0
10	0.631579	0.333333	1	0.516854	0
10	0.684211	0.333333	1	0.516854	0
12	0.473684	0	0	0.05618	0
12	0.578947	0	0	0.449438	0
12	0.578947	0.333333	0	0.314607	0
12	0.631579	0	0	0.786517	0
12	0.684211	0.333333	0	0.629213	0
13	0.842105	1	1	0.359551	1
13	0.842105	1	1	0.404494	1
13	0.894737	1	1	0.224719	1
13	0.894737	0.666667	1	0.292135	1
13	0.947368	0.666667	1	0.191011	1
13	0.947368	0.666667	1	0.078652	1
14	0.578947	0.333333	1	0.640449	1
14	0.684211	0.333333	1	0.640449	1
14	0.684211	0.333333	1	0.752809	1
14	1	0.333333	1	0.640449	1
15	0.736842	0.666667	1	0.865169	1
15	0.736842	0.666667	1	0.898876	1
15	0.789474	0.666667	1	0.865169	1
15	0.789474	0.666667	1	0.898876	1
15	1	0.666667	1	0.640449	1
16	0.736842	0.666667	1	0.640449	0
16	0.736842	1	1	0.876404	0
16	0.894737	1	1	1	0
16	0.894737	0.666667	1	0.898876	0
16	1	0.666667	1	0.898876	0
16	1	0.666667	1	0.52809	0
17	0.789474	1	1	0.146067	0
17	0.789474	1	1	0.224719	0
17	0.842105	1	1	0.146067	0
17	0.842105	1	1	0.224719	0
17	0.947368	0.666667	1	0.179775	0
17	0.947368	0.666667	1	0.179775	0
18	0.421053	0.666667	0	0.898876	0

Table 39: Input data after clustering by SOM - de-normalized

Neuron Number	Professor ID	Course level	Course Type	Course name	Timing
1	10	3	1	25	1
1	11	3	1	31	1
1	11	3	1	35	1
2	1	4	1	26	1
2	2	4	1	33	1
2	6	4	1	12	1
3	3	3	1	10	1
3	4	3	1	11	1
3	4	3	1	40	1
3	5	3	1	6	1
3	5	3	1	5	1
4	3	4	1	9	0
4	3	4	1	17	0
4	5	4	1	33	0
4	6	4	1	7	0
4	7	4	1	17	0
4	8	4	1	26	0
5	7	3	1	9	0
5	8	3	1	11	0
5	10	3	1	20	0
5	10	3	1	20	0
5	11	3	1	19	0
5	11	3	1	36	0
6	1	4	0	30	0
6	4	3	0	48	0
6	4	3	0	48	0
6	6	4	0	40	0
6	7	4	0	14	0
7	8	3	1	84	1
7	12	3	1	70	1
8	1	5	1	14	1
8	1	5	1	12	1
8	2	5	1	41	1
8	2	5	1	42	1
8	7	5	1	50	1
8	9	5	1	20	1
8	9	5	1	21	1
9	2	5	1	40	0
9	3	5	1	6	0
9	5	5	1	42	0

Table 39: Input data after clustering by SOM - de-normalized (Continued)

Neuron Number	Professor ID	Course level	Course Type	Course name	Timing
9	6	5	1	45	0
9	9	5	1	23	0
10	13	3	1	72	0
10	13	4	1	51	0
10	14	4	1	51	0
12	10	3	0	10	0
12	12	3	0	45	0
12	12	4	0	33	0
12	13	3	0	75	0
12	14	4	0	61	0
13	17	6	1	37	1
13	17	6	1	41	1
13	18	6	1	25	1
13	18	5	1	31	1
13	19	5	1	22	1
13	19	5	1	12	1
14	12	4	1	62	1
14	14	4	1	62	1
14	14	4	1	72	1
14	20	4	1	62	1
15	15	5	1	82	1
15	15	5	1	85	1
15	16	5	1	82	1
15	16	5	1	85	1
15	20	5	1	62	1
16	15	5	1	62	0
16	15	6	1	83	0
16	18	6	1	94	0
16	18	5	1	85	0
16	20	5	1	85	0
16	20	5	1	52	0
17	16	6	1	18	0
17	16	6	1	25	0
17	17	6	1	18	0
17	17	6	1	25	0
17	19	5	1	21	0
17	19	5	1	21	0
18	9	5	0	85	0

Matlab codes:

Self-organizing feature map neural network code in Matlab:

```
clear;  % delete all memory
clc;    % clear windows screen
clf;    % clear figure screen

net = selforgmap([2,1],100,4,'topologyFcn','hextop','distanceFcn',
'linkdist');


% Input data "P"
P = xlsread('Book1.xlsx');
% Configure inputs & outputs
net = configure (net,P);

%net = setwb(net,k);

% Figure network before training
plotsompos(net,P);

% Set the SOM Traning parameters stage
net.trainParam.epochs =10000;
%lp = learnsomb('pdefaults');
%lp.order_lr = 0.1;

%net.trainParam.LP = [];
%w = rand(6,2);
%[dW,ls] = learnsom(w,p,[],[],a,[],[],[],[],d,lp,ls)

% Traning stage
net= train(net,P);

% Figure network after training
plotsompos (net,P);

% Results in matrix
outputs = net(P);

%plotsompos (net,outputs);

%for each input in inputs, op_som will have a numbering between 1 to
n based on which cluster it belongs to.
 op_som=vec2ind(sim (net,(P)))';


 % convert a sparse matrix to full
outputs = full(outputs);

 output this to a file (excel)
xlswrite('test0.csv',outputs);
xlswrite('test1.csv',op_som);
```

```
%view(net)
centers = net.IW;

%nntraintool close

color coding Code = som_colorcode(outputs);
```

Feedforward Back-propagation neural network code in Matlab:

```
clear;                                  %Cleaning previous DATA
clc;
data= xlsread('Book1.xlsx');           %Training data Input
Input= ((data([1:64], [1:5])));   %Defining the input range
out= (data([1:64], [6]));         %Defining the output range
In=[Input ones(size (Input,1),1)];      %Adding Bias column to the
input data
NN=30;                                  %neurons for the first
hidden layer
NN2=20;                                 %neurons for the second
hidden layer
[m,n]=size(In);
Nout=size(out,2);
A=-0.5; B=0.5;                          %initial weights range
W1= A+(B-A)*rand(n,NN-1);               %First weights layer
definition
[e,r]=size(W1);
W2= A+(B-A)*rand(r+1,NN2-1);            %second weights layer
definition
W3= A+(B-A)*rad(NN2,Nout);             %thired weights layer
definition
eta=0.1; alfa=0.1;                     %Learning rate definition
W1n=zeros(size(W1));
E=10;
epoch=0;
EW1=zeros(m,r);
EW2=zeros(m,NN2);
DW1f=zeros (size(W1));
DW2f=zeros (size(W2));
DW3f=zeros (size(W3));

H1= In*W1;                          %first layer calculation
    H1f= 1./(1+exp( - H1));
    H1f= [H1f ones(size(H1f,1),1)];
while E>0.01                         %definig loop condition
    H2= H1f*W2;                     %second layer calculation
    H2f= 1./(1+exp( - H2));
    H2f= [H2f ones(size(H2f,1),1)];
    O1= H2f*W3;                     %third layer calculation
    Of= 1/(1+exp( - O1));
    error= out - O1f;
    D=error.*O1f .* (1-O1f);        %error calculation
    DW3=eta*(D')*H2f;
    Iw3=W3';
    %
                               %
    for i=1:size(D,1)          %
```

```
        EW2(i,:)=D(i,1)*Iw3;   %  APPlying Wight changes to the
        i=i+1;                 %    second Hidden Layer
    end                        %
 s=EW2(:,(1:size(EW2,2)-1));
 o=H2f.*(1-H2f);                %last layer wights changes
 o=o(:,(1:size(o,2)-1));
 s=s.*o;%
 DW2=eta*H1f*s;            %

     W2=W2+DW2;                 %new weights
    W3=W3+DW3';
    DW1f=W1;
    DW2f=W2;
    DW3f=W3;
    E=round(0.5*(sum(((error.*error)))),3);
    epoch = epoch + 1;
        if rem(epoch,50)==0     % Every 50 epochs, show how training 
is doing
                 disp([' Epoch ' num2str(epoch) '  SSE ' 
num2str(E)]);
        end
end

check= xlsread('Book1test.xlsx'); %Testing Results inputs and 
calculations
x= (check([1:15], [1:5]));
x=[x ones(size (x,1),1)];
x1=x*W1;
x1f=1./(1+exp( - x1));
x1f= [x1f ones(size(x1f,1),1)];
x2=x1f*W2;
x2f=1./(1+exp( - x2));
x2f= [x2f ones(size(x2f,1),1)];
y1= x2f*W3;
y1f=( 1./(1+exp( - y1)))

cout=O1f;
 beep, pause(0.5), beep,pause(0.5), beep,pause(0.5), 
beep,pause(0.5), beep
```

Centers (weight of the neurons after cluster) in matrix size (18X5)

[0.508771929666667	0	1	0.284644194666667	1
0.122807017666667	0.166666666500000	1	0.191011235833333	1
0.151315789625000	0.500000000000	1	0.2380617978125	0.375
0.157894737000000	0.333333333000000	1	0.224719101500000	0
0.463157894800000	0	1	0.177528090000000	0

```
0.144736842250000      0.166666666500000      0      0.410112359250000      0
0.473684210500000      0                      1      0.808988764000000      1
0.201754386166667      0.666666667000000      1.     0.241573033833333      1
0.210526316000000      0.666666667000000      1      0.294382022400000      0
0.649122806666667      0.222222222000000      1      0.595505618333333      0
0.589473684000000      0.133333333200000      0      0.447191011200000      0
0.894736842000000      0.833333333500000      1      0.258426966166667      1
0.736842105000000      0.333333333000000      1      0.668539325750000      1
0.810526315600000      0.666666667000000      1      0.833707864800000      1
0.873684210400000      0.800000000200000      1      0.788764044800000      0
0.881578947250000      0.833333333500000      1      0.202247191000000      0
0.421052632000000      0.666666667000000      0      0.898876404000000      0];
```