Data Driven Multivariate Technique for Fault Detection of Waste Water Treatment Plant

A Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Electronic Instrumentation and Control

Submitted by

Nitesh Gupta
Roll No: 801051010

Under the Guidance of

Dr. Gagandeep Kaur
Assistant Professor

Department of Electrical and Instrumentation Engineering

Thapar University
(Established under the section 3 of UGC act, 1956)

Patiala, 147004, Punjab, India
July 2012

DECLARATION

I hereby certify that the work is being presented in the thesis work entitled "**Data Driven Multivariate Technique for Fault Detection of Waste Water Treatment Plant**" in partial fulfillment of award of degree of **Master of Engineering** in **Electronics Instrumentation and Control** submitted in Electrical and Instrumentation Engineering department, Thapar University, Patiala is an authentic record of my own work carried under the supervision of **Dr. Gagandeep Kaur,** Assistant Professor, Department of Electrical and Instrumentation Engineering, Thapar University, Patiala, Punjab.

Date: 25/06/2012

Nitesh Gupta

Nitesh Gupta
801051010

I certify that the above statement made by the student is correct to the best of my knowledge and belief.

Date:

Dr. Gagandeep Kaur
Assistant Professor,
Department of Electrical
Instrumentation Engineering,
Punjab

S. Ghosh

Dr. Smarajit Ghosh
Head of Department,
Department of Electrical and
Instrumentation Engineering,
Thapar University, Patiala,
Punjab

Countersigned By

Dr. Saroj Kumar Mohapatra
Dean of Academic Affairs,
Thapar University, Patiala,
Punjab

ABSTRACT

Classification of data originating from sensor in to two distinct categories (good and bad) is a challenging job and has a wide spread application. A lot of research is going on in this particular area. Because of the enhanced memory capacity of the present day computers, data logging has reached to a new level. The analyst has to classify the data according to their traits from the offline logged data. The whole task of collection of raw data, classification of data according to their traits involves different statistical as well as soft computational techniques. There are two types of classification algorithms like supervised classification and unsupervised classification. In supervised classification, the classification is done using neural network and in unsupervised classification the classification is done using different clustering algorithm.

This dissertation studies and evaluates the performance of different classification algorithm in a waste water treatment plant. First of all waste water treatment plant is taken in to consideration and data driven classification techniques are implemented to find out the healthy data and faulty data. The healthy and faulty data is classified using supervised and unsupervised classification.

ACKNOWLEDGEMENT

The real spirit of achieving a goal is through the way of excellence and austere discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various personalities.

I am very thankful to **Dr. Abhijit Mukherjee**, Director of Thapar University, Patiala for providing the facilities for the completion of this thesis.

I express my deep sense of gratitude towards **Dr. Smarajit Ghosh**, Professor and Head of the Department of Electrical & Instrumentation Engineering, Thapar University, Patiala who has been a constant source of inspiration for me throughout this work.

With deep sense of gratitude I express my sincere thanks to my esteemed and worthy supervisor **Dr. Gagandeep Kaur**, Department of Electrical and Instrumentation Engineering, Thapar University, Patiala for her valuable guidance in carrying out this work under her effective supervision, encouragement, enlightenment and cooperation. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Her feedback and editorial comments were also invaluable for writing of this thesis.

We express our deep sense of gratitude to Mr. Subhransu Padhee for his valuable and encouraging guidance, which played a major role in the formation of this project work. Their co-oporation and timely suggestions have unparallel stimuli for us to travel eventually towards the completion of this project.

I am also thankful to all the staff members of the Department for their full cooperation and help. This acknowledgement would be incomplete if I do not mention the emotional support and blessings provided by my friends. I had a pleasant enjoyable and fruitful company with them.

Nitesh gupta

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

RELATED PUBLICATION

Nitesh Gupta, Gagandeep Kaur, Subhransu Padhee, "Data Driven Multivariate Technique for Fault Detection of Waste Water Treatment Plant" in International Journal of Engineering and Advanced Technology (ISSN:2249–8958,Volume-1,Issue-4) April 2012.

Chapter 1

Introduction

1.1Overview

In a process control application, identification of process structure and parameters, monitoring the quality of the process outputs and detection of abnormal trends in different variables, is an upcoming research area. Collection of raw data from different sensors, processing the data and extracting valuable information about the condition of the process is a very challenging task. Because of the enhanced memory capacity of the present day computers, data logging has reached to a new level. The analyst has to classify the data according to their traits from the offline logged data. The analyst needs to understand the process dynamics with the help of historical data or logged data. Different statistical and machine learning algorithms are used to determine the faulty trends in process dynamics. To operate the plat in an efficient manner and to obtain quality product in economical way, fault detection in process dynamics is important. The whole new branch of statistical process control came in to existence, because in modern day monitoring and fault diagnosis of process control application are in huge demand because every industry needs to give consistent output with the best of the quality. Looking in to the complexity of the process industry, multivariate statistical process control is used for the above purpose.

1.2 Objective of the dissertation

The objective of the study is to use statistical and soft computational based technique to model the process and monitor of the quality of process variables. The main objectives are

- Identification of process dynamics
- Quality monitoring
- Fault detection

Identification of process dynamics can be obtained from system identification tools. Different multivariate statistical techniques like PCA, ICA, PLS can be used for dimensionality reduction and back-propagation algorithms and variants of BPA, SVM can be used as an classifier which classifies different kind of data.

1.3 Organization of the dissertation

The dissertation is organized as follows:

Chapter 2 describes different soft computing and supervised classification techniques.

Chapter 3 explains unsupervised classification technique where clustering and its types are briefly described.

Chapter 4 explains the Literature Review of dissertation.

Chapter 5 discusses dimensionality reduction in which principle component analysis and its mathematical analysis are explained.

Chapter 6 explains the fault detection in waste water treatment plant in which results of PCA, Supervised and Unsupervised algorithms are mentioned.

Chapter 7 provides the conclusion and future scope of entire work.

Chapter – 2
Soft Computing and Supervised Classification Techniques

2.1 Artificial Intelligence

According to Luger and Stubblefield, artificial intelligence is a branch of science that is concerned with the automation of intelligent behavior. It creates the capability in a device to perform functions that are normally associated with human intelligence, such as reasoning and optimization through experience [1].

2.2 Soft Computing

The idea of soft computing was first introduced in 1981 when Lofti Zadeh introduced his first paper on soft data analysis. According to Lofti Zadeh "Soft Computing is an emerging approach to computing which parallel the remarkable ability of the human mind to reason and learn in a environment of uncertainty and imprecision" [5]. Soft Computing became a formal Computer Science area of study in early 90s Earlier computational approaches could model and precisely analyze only relatively simple systems. More complex systems arising in biology, medicine, the humanities, management sciences, and similar fields often remained intractable to conventional mathematical and analytical methods. That said, it should be pointed out that simplicity and complexity of systems are relative, and many conventional mathematical models have been both challenging and very productive. Soft computing deals with imprecision, uncertainty, partial truth, and approximation to achieve practicability, robustness and low solution cost. Soft computing differs from hard computing in its tolerance to imprecision, uncertainty and partial truth. The Soft Computing consists of several computing paradigms mainly: Fuzzy Systems, Neural Networks, and Genetic Algorithms [2, 3].

• Fuzzy System: For knowledge representation via fuzzy If – Then rules.

• Neural Networks: For learning and adaptation

• Genetic Algorithms: For evolutionary computation

These methodologies form the core of Soft Computing. Hybridization of these three creates a successful synergic effect; that is, hybridization creates a situation where different entities

cooperate advantageously for a final outcome. Soft Computing is still growing and developing. Hence, a clear definite agreement on what comprises Soft Computing has not yet been reached. More new sciences are still merging into Soft Computing.

2.3 Goals of Soft Computing

Soft Computing is a new multidisciplinary field, to construct new generation of Artificial Intelligence, known as Computational Intelligence.

• The main goal of Soft Computing is to develop intelligent machines to provide solutions to real world problems, which are not modeled, or too difficult to model mathematically.

• Its aim is to exploit the tolerance for Approximation, Uncertainty, Imprecision, and Partial Truth in order to achieve close resemblance with human like decision making.

Approximation: Here the model features are similar to the real ones, but not the same.

Uncertainty: here we are not sure that the features of the model are the same as that of the entity.

Imprecision: here the model features (quantities) are not the same as that of the real ones, but close to them.

2.4 Importance of Soft Computing

Soft computing differs from hard (conventional) computing. Unlike hard computing, the soft computing is tolerant of imprecision, uncertainty, partial truth, and approximation. The guiding principle of soft computing is to exploit these tolerances to achieve tractability, robustness and low solution cost [2]. In effect, the role model for soft computing is the human mind. The four fields that constitute Soft Computing (SC) are: Fuzzy Computing (FC), Evolutionary Computing (EC), Neural computing (NC), and Probabilistic Computing (PC), with the latter subsuming belief networks, chaos theory and parts of learning theory. Soft computing is not a concoction, mixture, or combinations rather soft computing is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In principal the constituent methodologies in Soft computing are complementary rather than competitive. Soft computing may be viewed as a foundation component for the emerging field of Conceptual Intelligence.

2.5 Fuzzy Logic System

Fuzzy Logic is extension of Boolean logic [7]. It incorporates partial values of truth. Instead of sentences being completely true or completely false, here in fuzzy logic they are assigned a value which represents their degree of truthiness. In fuzzy systems, values are indicated by a number called as truth value. It lies in the range from 0 to 1. 0.0 represents absolute falseness and 1.0 represents absolute truth. Fuzzification is generalization of theory from discrete to continuous [5]. Fuzzy logic is important to artificial intelligence. Fuzzy logic allows computers to answer to a certain degree unlike Boolean logic which gives one extreme or the other. Computers are allowed to think more human-like. Nothing in our perception is extreme. However, it is true only to a certain degree. In fuzzy logic, machines think in degrees. It can solve problems in the cases where there is no simple mathematical model. Fuzzy logic solves highly nonlinear processes. Fuzzy logic uses expert knowledge to make decisions. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information [4]. In a sense, fuzzy logic resembles human decision making with its ability to work from approximate data and find precise solutions. Unlike classical logic, which requires a deep understanding of a system, exact equations, and precise numeric values, fuzzy logic incorporates an alternative way of thinking, which allows modeling complex systems using a higher level of abstraction originating from our knowledge and experience. Fuzzy logic allows expressing this knowledge with subjective concepts such as very hot, bright red, and a long time, which are mapped into exact numeric ranges.

2.5.1 Fuzzy Set Theory

This section introduces some elements of fuzzy set theory in a more formal way than the previous one. The properties and features of classical set theory are used to introduce their corresponding fuzzy counterparts. Most of the operators and essential definitions are also collected in a glossary in the front of the dissertation.

Let X be a space of objects and x be a generic element of X. A classical set A, $A \in X$, is defined as a collection of elements or objects $x \in X$, such that each element (x) can either belong or not to the set A. By defining a characteristic (or membership) function for each element x in X, we can represent a classical set A by a set of ordered pairs (x, 0) or (x, 1),

which indicates $x \in A$, respectively [6, 8]. Unlike a conventional set, a fuzzy set expresses the degree to which an element belongs to a set. Hence the membership function of a fuzzy set is allowed to have values between 0 and 1 that denote the degree of membership of an element in the given set.

2.5.1.1 Fuzzy sets and membership functions

If X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu A(x)) | x \in X\}$$

where, $\mu A(x)$ is called the membership function (MF) for the fuzzy set A. The MF maps each element of X to a membership degree between 0 and 1. Obviously, the definition of a fuzzy set is a simple extension of the definition of a classical (crisp) set in which the characteristic function is permitted to have any value between 0 and 1 [6, 9]. If the value of the membership function is restricted to either 0 or 1, then A is reduced to a classical set. For clarity, we shall also refer to classical sets as ordinary sets, crisp sets, non fuzzy sets, or just sets. Usually, X is referred to as the universe of discourse**,** or simply the universe**,** and it may consist of discrete (ordered or non-ordered) objects or it can be a continuous space.

2.5.2 Fuzzy Logic Operation

Fuzzy logic requires some numerical parameters in order to operate such as what is considered significant error and significant rate of change of error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them.

2.5.3 Fuzzy Logic Controller

Block diagram of fuzzy logic controller is shown in figure 2.1. The fuzzy logic controller provides an algorithm, which converts the expert knowledge into an automatic control strategy. Fuzzy logic is capable of handling approximate information in a systematic way and therefore it is suited for controlling non linear systems and is used for modelling complex systems, where an inexact model exists or systems where ambiguity or vagueness is common [10,11].

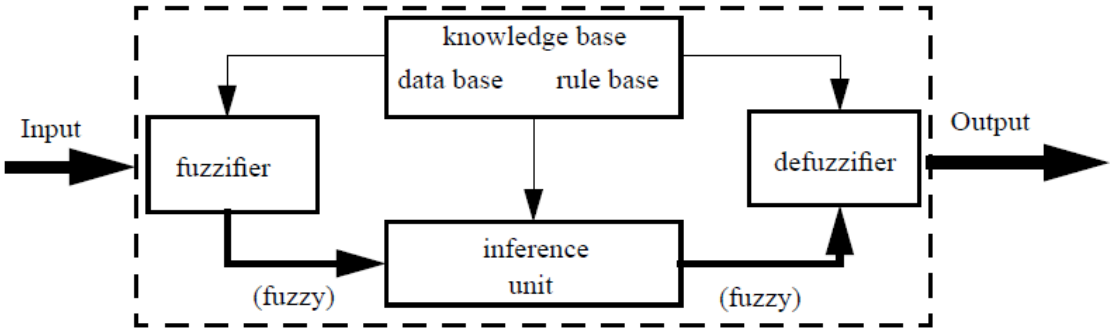


Figure 2.1: Block Diagram of Fuzzy Logic System

The fuzzy control systems are rule-based systems in which a set of fuzzy rules represent a control decision mechanism for adjusting the effects of certain system stimuli. The rule base reflects the human expert knowledge, expressed as linguistic variables, while the membership functions represent expert interpretation of those variables. The above figure 2.1 shows the block diagram of fuzzy logic system. It has different blocks: fuzzifier, knowledge base, inference unit and defuzzifier. The crisp inputs are supplied to the input side Fuzzification unit. The Fuzzification unit converts the crisp input in to fuzzy variable. The fuzzy variables are then passed through the fuzzy rule base. The fuzzy rule base computes the input according to the rules and gives the output [11]. The output is then passed through de-fuzzification unit where the fuzzy output is converted to crisp output (The role of the fuzzifier in a Fuzzy Logic System is to convert a crisp input variable into a fuzzy set that is ready to be processed by the inference engine. The inference engine using the fuzzified inputs and the rules stored in the rule base processes the incoming data and produces an (fuzzy) output. This output needs to be used in the outside world and thus needs to be converted from fuzzy to crisp; the defuzzifier performs this operation.

2.6 Genetic Algorithm

Idea of evolutionary computing was introduced in the year 1960s by I. Rechenberg in his work "Evolution strategies". His idea was then developed by other researchers. Genetic Algorithms (GAs) were invented by John Holland in early 1970's to mimic some of the processes observed in natural evolution. Later in 1992 John Koza used GAs to evolve programs to perform certain tasks. He called his method "Genetic Programming" (GP). GAs simulates natural evolution, a combination of selection, recombination and mutation to evolve a solution to a problem [3]. Gas simulates the survival of the fittest, among

individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome in our DNA (Deoxyribonucleic acid). DNA contains the genetic instructions used in the development and functioning of all known living organisms.

2.6.1 What are Genetic Algorithms?

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. Genetic algorithms (GAs) are a part of evolutionary computing, a rapidly growing area of artificial intelligence. GAs are inspired by Darwin's theory about evolution - "survival of the fittest" [2]. GAs represent an intelligent exploitation of a random search used to solve optimization problems. GAs, although randomized, exploit historical information to direct the search into the region of better performance within the search space. In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones.

2.6.2 Why Genetic Algorithms?

Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime." Salvatore Mangano Computer Design, May 1995. GA is better than conventional AI, in that it is more robust. Unlike older AI systems, GAs do not break easily even if the inputs changed slightly, or in the presence of reasonable noise. In searching a large state-space, multi-modal state-space, or n-dimensional surface, a GA may offer significant benefits over more typical search of optimization techniques, like – linear programming, heuristic, depth-first, breath-first.

2.6.3 Mechanics of Biological Evolution

Genetic Algorithms are a way of solving problems by mimicking processes the nature uses - Selection, Crosses over, Mutation and Accepting to evolve a solution to a problem. Every organism has a set of rules, describing how that organism is built, and encoded in the genes of an organism. The genes are connected together into long strings called chromosomes. Each gene represents a specific trait (feature) of the organism and has several different settings, e.g. setting for a hair color gene may be black or brown. The genes and

their settings are referred as an organism's genotype. When two organisms mate they share their genes. The resultant offspring may end up having half the genes from one parent and half from the other parent. This process is called crossover [12]**.** The newly created offspring can then be mutated. A gene may be mutated and expressed in the organism as a completely new trait. Mutation means, that the elements of DNA are a bit changed. This change is mainly caused by errors in copying genes from parents. The fitness of an organism is measured by success of the organism in its life.

2.6.4 Artificial Evolution and Search Optimization

The problem of finding solutions to problems is itself a problem with no general solution. Solving problems usually mean looking for solutions, which will be the best among others. In engineering and mathematics finding the solution to a problem is often thought as a process of optimization. Here the process is : first formulate the problems as mathematical models expressed in terms of functions; then to find a solution, discover the parameters that optimize the model or the function components that provide optimal system performance. The well-established search / optimization techniques are usually classified in to three broad categories Enumerative, Calculus-based, and Guided random [13] search techniques.

2.6.4.1 Enumerative Method

These are the traditional search and control strategies. They search for a solution in a problem space within the domain of artificial intelligence. There are many control structures for search. The depth-first search and breadth-first search are the two most common search strategies. Here the search goes through every point related to the function's domain space (finite or discredited), one point at a time. They are very simple to implement but usually require significant computation. These techniques are not suitable for applications with large domain spaces.

2.6.4.2 Calculus Based Techniques

Here a set of necessary and sufficient conditions to be satisfied by the solutions of an optimization problem. They subdivide into direct and indirect methods.

a) Direct or Numerical Methods

Methods like Newton or Fibonacci, seek extremes by "hopping" around the search space and assessing the gradient of the new point, which guides the search. This is simply the notion of "hill climbing", which finds the best local point by climbing the steepest permissible gradient. These techniques can be used only on a restricted set of "well behaved" functions.

b) Indirect Methods

These kinds of methods are responsible for search of local extremes by solving the usually non-linear set of equations resulting from setting the gradient of the objective function to zero. The search for possible solutions (function peaks) starts by restricting itself to points with zero slope in all directions.

2.6.4.3 Guided Random Search Techniques

These are based on enumerative techniques but they use additional information to guide the search. Two major subclasses a re simulated annealing and evolutionary algorithms. Both are evolutionary processes.

a) **Simulated annealing**

It uses a thermodynamic evolution process to search minimum energy states.

b) **Evolutionary algorithms**

It uses natural selection principles. This form of search evolves throughout generations, improving the features of potential solutions by means of biological inspired operations. Genetic Algorithms (GAs) are a good example of this technique**.**

2.7 Artificial Neural Network

Artificial neural networks (ANN) are the simplified mathematical model of biological neural network [17]. So before defining the artificial neural network we should have an idea about biological neural network.

2.7.1 Biological Neural Network

Human brain consists of 10^{10} neurons. Each neuron is connected with 10^4 other neurons. A biological neural network describes a population of physically interconnected neurons or a group of disparate neurons whose inputs or signaling targets define a recognizable circuit [18]. Communication between neurons often involves an electrochemical process. The interface through which they interact with surrounding neurons usually consists of several dendrites (input connections), which are connected via synapses to other neurons, and one axon (output connection). If the sum of the input signals surpasses a certain threshold, the neuron sends an action potential (AP) at the axon hillock and transmits this electrical signal along the axon. The connections between neurons are much more complex than those implemented in neural computing architectures. The basic kinds of connections between neurons are chemical synapses and electrical gap junctions [17]. One principle by which neurons work is neural summation, i.e. potentials at the post synaptic membrane will sum up in the cell body. If the depolarization of the neuron at the axon goes above threshold an action potential will occur that travels down the axon to the terminal endings to transmit a signal to other neurons.

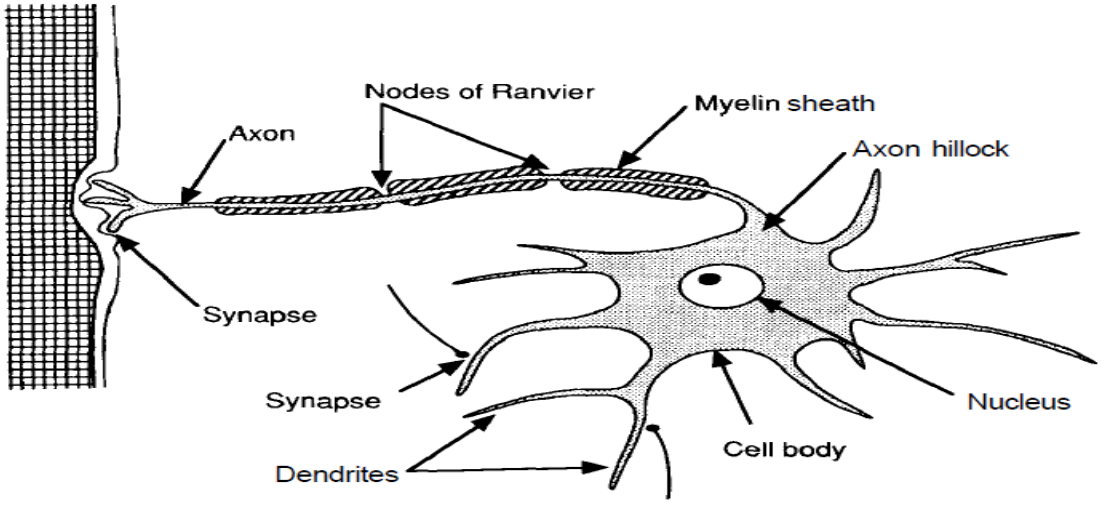


Figure 2.2 Biological Neuron

The Figure 2.2 shows the basic architecture of biological neuron, which consists of synapse, dendrites, soma and axon.

- **Dendrites** are the branching fibers extending from the cell body or soma.

- **Soma or cell body** of a neuron contains the nucleus and other structures, support chemical processing and production of neurotransmitters.
- **Axon** is a singular fiber carries information away from the soma to the synaptic sites of other neurons (dendrites and somas), muscles, or glands.
- **Axon hillock** is the site of summation for incoming information. At any moment, the collective influence of all neurons that conduct as impulses to a given neuron will determine whether or not an action potential will be initiated at the axon hillock and propagated along the axon.
- **Synapse** is the point of connection between two neurons or a neuron and a muscle or a gland. Electrochemical communication between neurons takes place at these junctions.
- **Terminal Buttons** of a neuron are the small knobs at the end of an axon that release chemicals called neurotransmitters.

2.7.1.1 Information Flow in a Neural Cell:

The input/output and propagation of information are shown below.

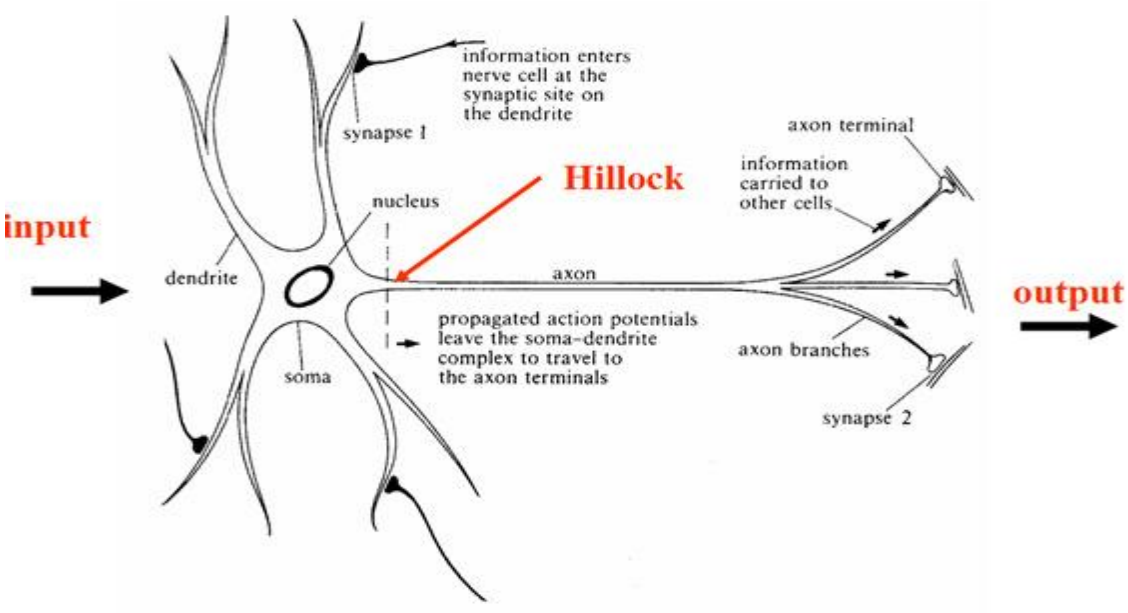


Figure 2.3 Input/output Propagation

Dendrites receive activation from other neurons.

Soma processes the incoming activations and converts them into output activations.

Axons act as transmission lines to send activation to other neurons.

Synapses the junctions allow signal transmission between the axons and dendrites.

The process of transmission is by diffusion of chemicals called neuron-transmitters.

2.7.2 Artificial Neural Network

McCulloch-Pitts introduced a simplified model of real neurons. It is a highly interconnected network of large number of processing elements called neurons. The ANN is capable of performing on nonlinear input and output systems in the workspace due to its large parallel interconnection between different layers and its nonlinear processing characteristics. An artificial neuron basically consists of a computing element that performs the weighted sum of the input signal and the connecting weight. The sum is added with the bias or threshold and the resultant signal is then processed for nonlinear function of sigmoid or hyperbolic tangent type [14, 15]. Every neuron is associated with three parameters whose learning can be adjusted with these three parameters. These three parameters are:

1. The connecting weights.
2. The bias.
3. The slope of non linear function.

The structure of a neural network (NN) may be single layer or it may be multilayer. Each neuron of the one layer is connected to each neuron of the next layer. Figure 2.4 shows the simple perceptron model of neural network.

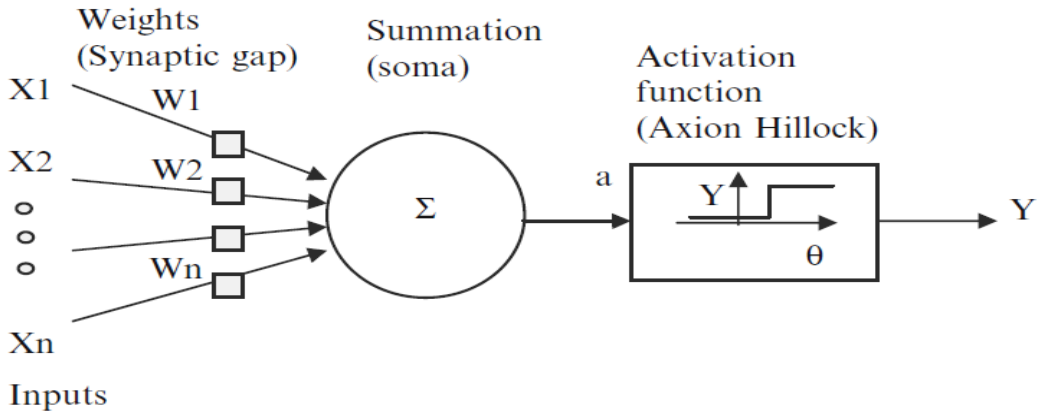


Figure 2.4 Simple model of artificial neural network

Total input I received at the soma is defined as:

$$I = w_1 x_1 + w_2 x_2 + - - + w_n x_n$$

$$I = \sum_{i=1}^{n} w_i x_i$$

The output is a non linear function of input and is defined as $y = \phi(I)$

Threshold function is a commonly used activation function. So output is

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i - \theta \right)$$

2.7.3 Different Neural Network Architectures

The neural network can be single layer or multi layer [19]. The multi layer neural network has one or more than one hidden layers. Hidden layers are used for better computations.

2.7.3.1 Single Layer Feed-forward Network

This kind of neural network consists of two layers, called input layer and output layer. The input layer neurons receive all the input signals and output layer neurons receives all the output signals. The links carrying weight connect each input neuron to a output neuron but not its opposite. This kind of network is known as feed-forward in type or having acyclic nature [22]. Despite these two layers, the network is called single layer because only the output layer alone performs computation. The input layer only transmits signals to the output layer. So it is called single layer feed-forward network. Systematic diagram of SLFN is as shown below:

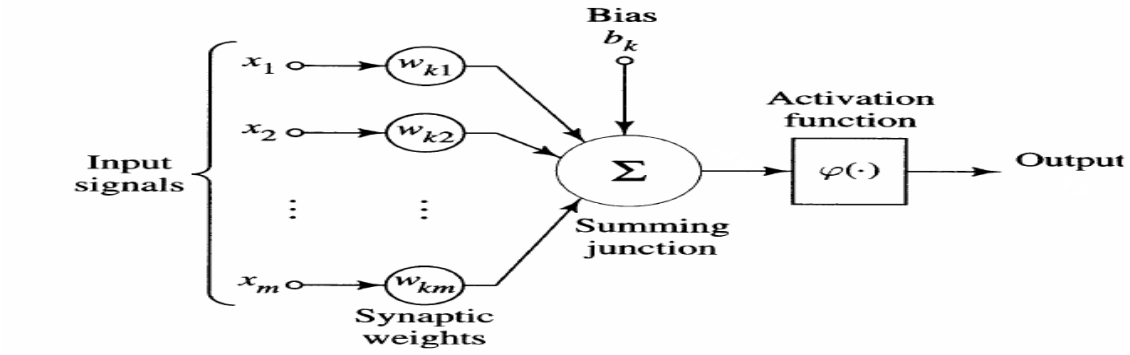


Figure 2.5 Single layer feed-forward network

Activation Functions: There are different activation functions in neural network, namely

Logisitic function is defined as $\frac{1}{1+e^{-\alpha x}}$

Hybperbolic tangent function is defined as $\frac{1-e^{-\alpha x}}{1+e^{-\alpha x}}$

Graphical representation of activation functions us shown in the figure 2.6.

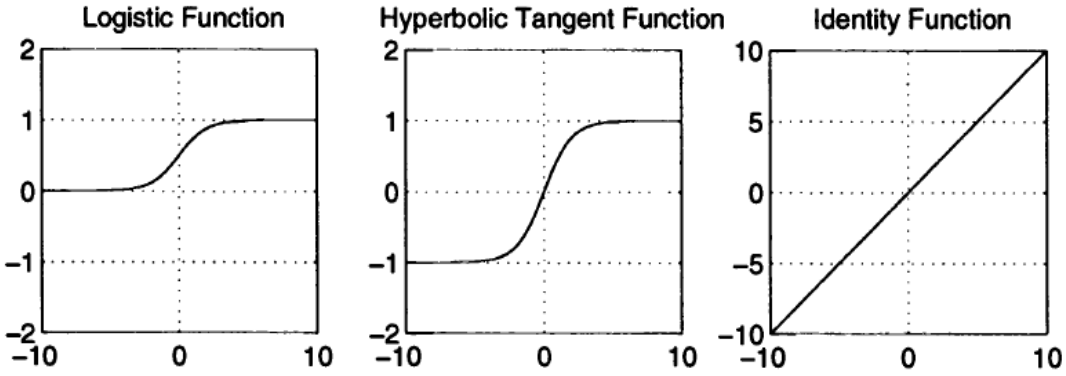


Figure 2.6 Different activation functions.

2.7.3.2 Multilayer Feed-forward Network

This network, as the name suggests consists up of multiple layers. Hence, architectures of this class besides having an input and an output layer it also has one or more intermediate layers known as hidden layers [21]. The most important properties of Multilayer feed-forward neural network (MFN) are: the complex-valued weights, inputs and output coded by the *k*th roots of unity and the activation function, which maps the complex plane into the unit circle. MFN learning is reduced to the movement along the unit circle, it is based on a simple linear error correction rule and it does not require a derivative. By using a traditional architecture of multilayer feed-forward neural network and the high functionality of the MFN, it is possible to obtain a new powerful neural network. Its training does not require a derivative of the activation function and its functionality is higher than the functionality of MFF containing the same number of layers and neurons. Simplified structure of MFF is as shown in figure:

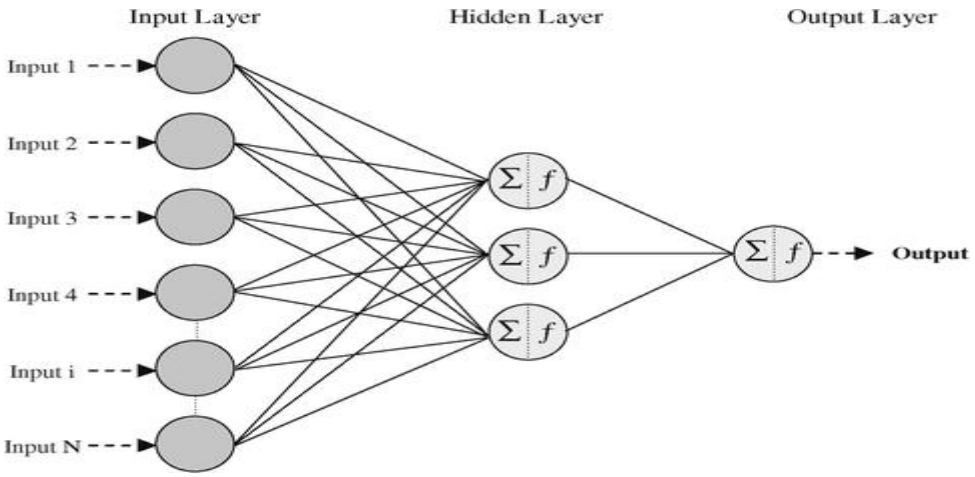


Figure 2.7 Multilayer feed forward network having multiple outputs

The working principle of MFN is shown in the figure 2.8. The computational elements of the hidden layers are called as hidden neurons or units [23]. The hidden layer helps in performing useful inter-mediate computations before sending the input to the output layer. Input layer neurons are linked to hidden layer's neurons and weight on these links is referred to as input hidden layer weight. Also, hidden layer neuron is linked to output layer neurons and its corresponding weight is referred to as hidden output layer weight.

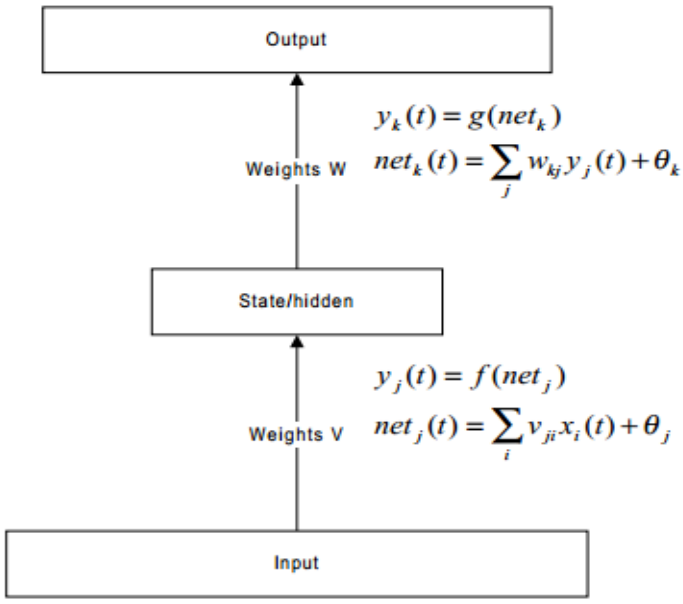


Figure 2.8 Working principle of multi layer feed forward network.

The hidden layers in the in MFN can be more than one. The architecture of such kind of network is as shown:

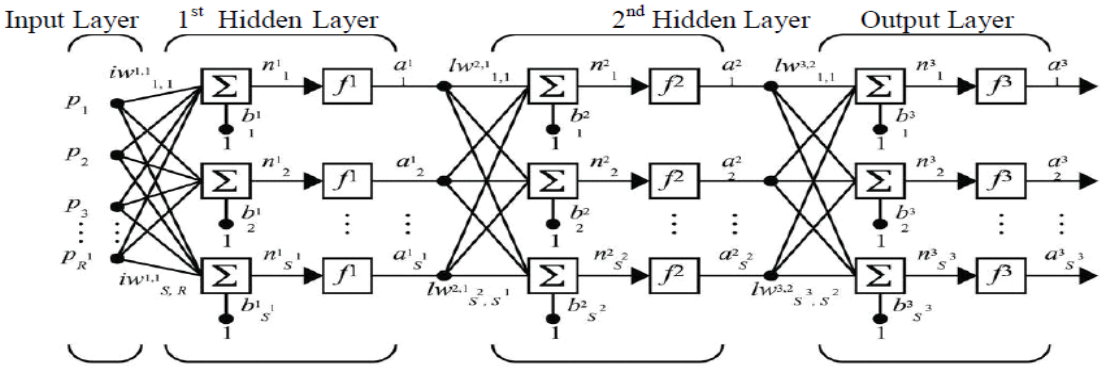


Figure 2.9 MFN with multiple hidden layers and multiple outputs

2.7.3.3 Recurrent Networks

This network differs from the feed-forward architecture in the way that there will be at least one feedback loop. Hence, in this network, for e.g., there could exist one layer for feedback connection. There can also be neurons with self feedback links, that is, the output of the neuron is fed back into itself as an input. Basic architecture of recurrent network is as shown below:

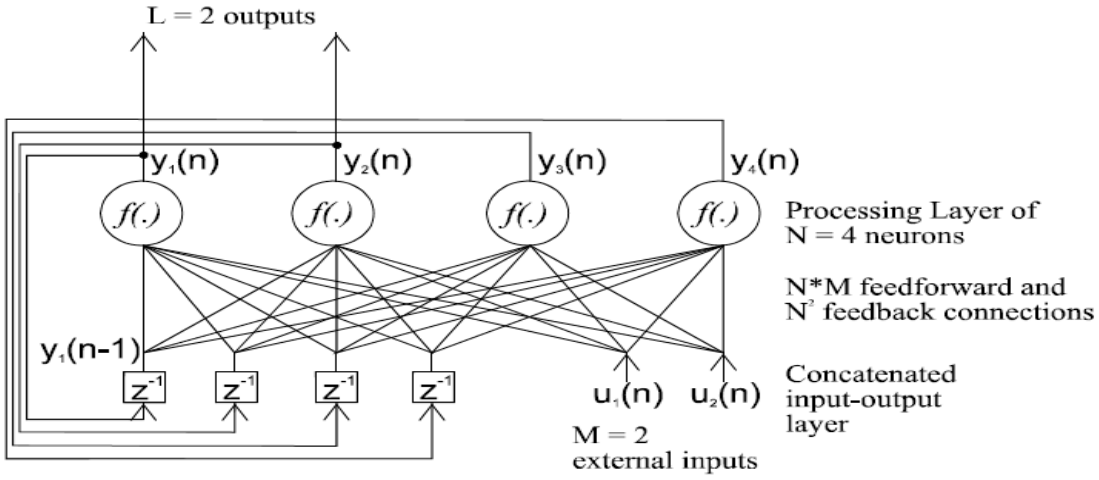


Figure 2.10 Basic Architecture of Recurrent Neural Network.

The architectures of recurrent networks can take many different forms. However, they all share the following common features:

1. They incorporate some form of Multi-Layer Perceptron as a sub-system.

2. They exploit the powerful non-linear mapping capabilities of the Multi-Layer Perceptron, plus some form of memory.

It can learn many behaviors / sequence processing tasks / algorithms / programs that are not learnable by traditional machine learning methods [24]. This explains the rapidly growing interest in artificial RNNs for technical applications: general computers which can learn algorithms to map input sequences to output sequences, with or without a teacher. Architecture of recurrent network is broadly classified into two types:

a) Fully recurrent neural network.
b) Partially recurrent neural network.

a) Fully Recurrent Neural Network

In fully recurrent neural network each neuron is fully connected to all other neurons and itself [26]. Architecture of such kind of network is shown below:

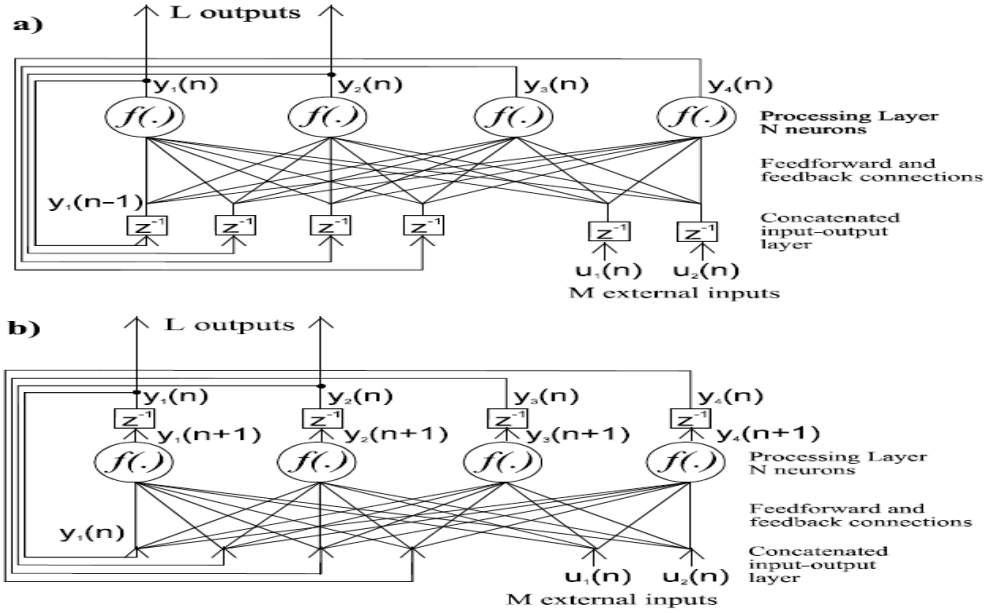


Figure 2.11 FRNN presented in two different ways: a) with delayed external inputs ; b) with delayed neuron outputs.

b) Partially Recurrent Neural Network

As its name says it is kind of recurrent neural network in which partial outputs are feedback to input. Architecture of this kind of network is given as:

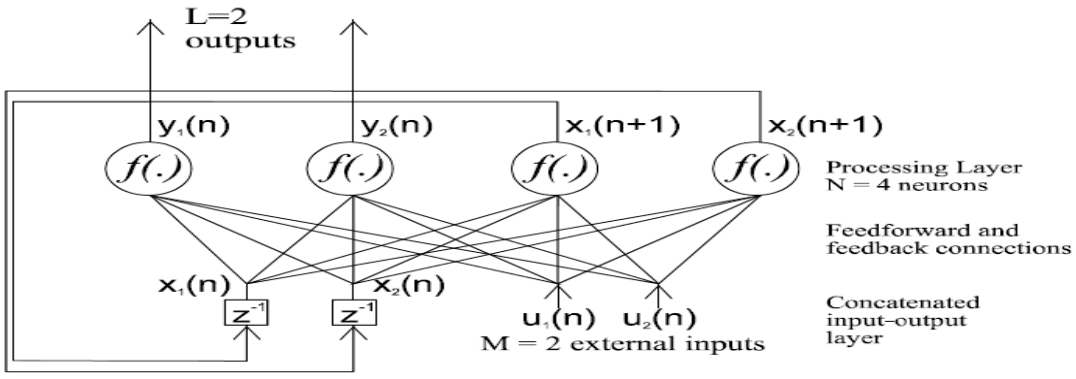


Figure 2.12 Partially recurrent neural network.

2.7.3.4 Working Principle of Recurrent Neural Network

Before we discuss the working principle of recurrent network, we need to introduce a simple delay building block. A simple delay building block is as shown below:

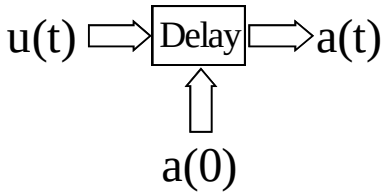


Figure 2.13 Delay Building Block

The delay output a(t) is computed from its input u(t) according to:

$$a(t) = u(t-1)$$

Thus the output is delayed by one step. It requires that the output to be initialized at t=0. This initial condition is indicated in the above figure by an arrow coming into the bottom of delay block. Now the working of recurrent neural network is defined in the figure 2.14. In this kind of network there is a vector let us suppose P which supplies the initial conditions [25] (i.e. a(0) = P)

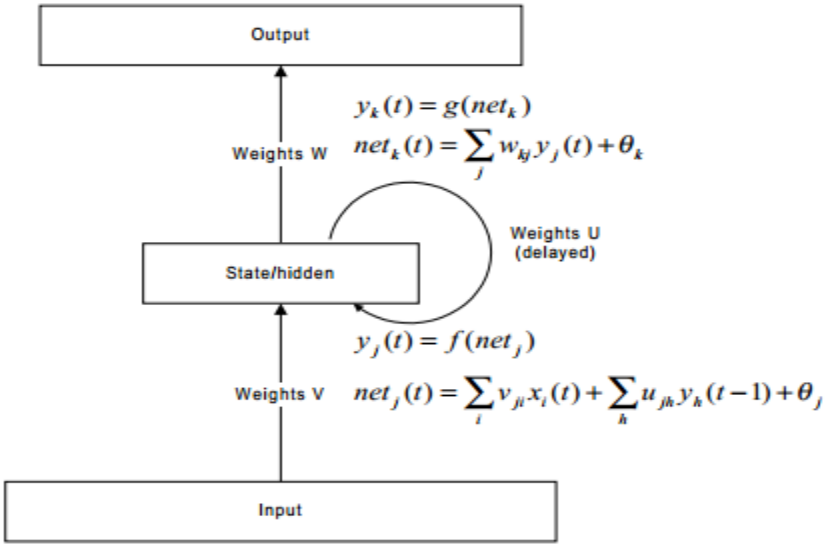


Figure 2.14 Working principle of recurrent neural network.

Then the future outputs are computed from previous outputs. These kind of networks are potentially more powerful than feed-forward network.

2.7.4 Characteristics of Artificial Neural Network

Some characteristics of neural networks are as follows:

1. The Neural Network exhibits mapping capabilities, i.e., they could map input patterns to its associated output pattern.
2. The Neural Network learns by means of examples. Hence, its architectures could be trained using known examples of the problem before they can be tested for its inference capabilities on an un-known instance of that problem. They could therefore identify new objects which were previously untrained.
3. The Neural Network possesses capabilities to generalize. Hence, it can find new outcomes from the past trends.
4. The Neural Network is a robust system and is fault tolerant. It can therefore recall the full patterns from the incomplete or partial, noisy patterns.
5. The Neural Network can compute information in parallel at fast speed and in a very distributed manner.

2.7.5 Learning Methods of Artificial Neural Network

Learning methods of artificial neural network are broadly classified into two types:

a) Supervised Learning.
b) Unsupervised Learning.

2.7.5.1 Supervised Learning

It is the machine learning task of inferring a function from supervised (labeled) training data. The training opportunity consists of a set of training examples [29]. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete, see classification) or a regression function. The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. In order to solve a given problem of supervised learning, one has to perform the following steps [27]:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality, but should contain enough information to accurately predict the output.

4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support of machine or decision trees.
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation
6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

2.7.6 Issues in Supervised Learning

There are four major issues to consider in supervised learning:

1. Bias-variance tradeoff

A first issue is the tradeoff between bias and variance. Imagine that we have available several different, but equally good, training data sets. A learning algorithm is biased for a particular input X if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for X. A learning algorithm has high variance for a particular input X if it predicts different output values when trained on different training sets. The prediction error of a learned classifier is related to the sum of the bias and the variance of the learning algorithm. Generally, there is a tradeoff between bias and variance. A learning algorithm with low bias must be "flexible" so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance. A key aspect of many supervised learning methods is that they are able to adjust this tradeoff between bias and variance.

2. Function complexity and amount of training data

The second issue is the amount of training data available relative to the complexity of the "true" function (classifier or regression function). If the true function is simple, then an "inflexible" learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in

different parts of the input space), then the function will only be learnable from a very large amount of training data and using a "flexible" learning algorithm with low bias and high variance. Good learning algorithms therefore automatically adjust the bias/variance tradeoff based on the amount of data available and the apparent complexity of the function to be learned.

3. Dimensionality of the input space

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimension, the learning problem can be difficult even if the true function only depends on a small number of those features. This is because the many "extra" dimensions can confuse the learning algorithm and cause it to have high variance. Hence, high input dimensionality typically requires tuning the classifier to have low variance and high bias. In practice, if the engineer can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seek to identify the relevant features and discard the irrelevant ones. This is an instance of the more general strategy of dimensionality reduction which seeks to map the input data into a lower dimensional space prior to running the supervised learning algorithm.

4. Noise in the output values

A fourth issue is the degree of noise in the desired output values (the supervisory targets). If the desired output values are often incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples. Attempting to fit the data too carefully leads to over fitting. You can over fit even when there are no measurement errors (stochastic noise) if the function you are trying to learn are too complex for your learning model. In such a situation that part of the target function that cannot be modeled "corrupts" your training data - this phenomenon has been called deterministic noise. When either type of noise is present, it is better to go with a higher bias, lower variance estimator. In practice, there are several approaches to alleviate noise in the output values such as early stopping to prevent over fitting as well as detecting and removing the noisy training examples prior to training the supervised learning algorithm. There are several algorithms that identify noisy training examples and removing the

suspected noisy training examples prior to training has decreased generalization error with statistical significance.

2.7.7 Basic Architecture of Supervised Learning

The basic structure of supervised learning is shown in figure 2.15. It is also called error based learning [28, 30]. Supervised learning is the machine learning task of inferring a function from supervised (labeled) training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete, see classification) or a regression function (if the output is continuous, see regression). The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data.

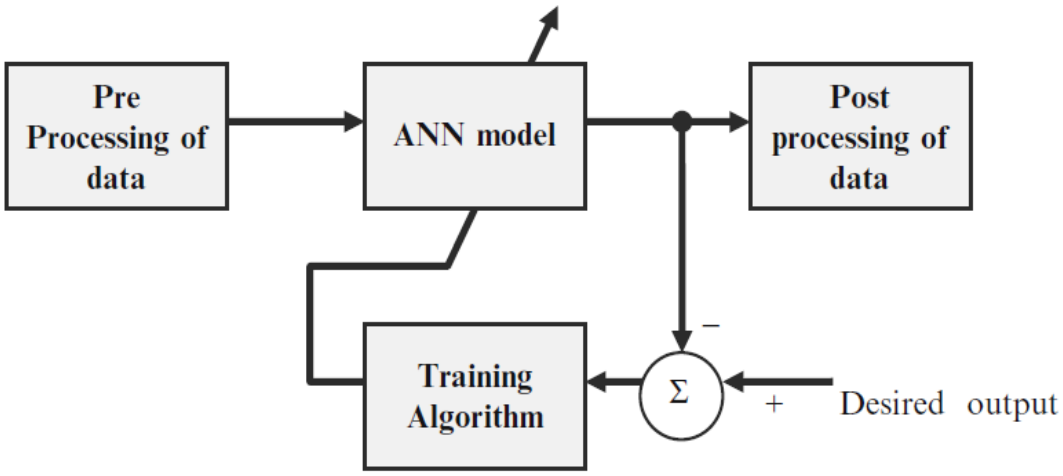


Figure 2.15: Basic structure of supervised learning

2.7.8 Back-propagation Learning Algorithm

Back-propagation is a systematic method of supervised learning of multi layer neural network. Most of the multi layer feed forward neural network is trained using back-propagation network. Back-propagation Algorithm is a common method for training artificial neural networks to perform a given task. It was first proposed by Arthur E. Bryson and Yu-

Chi Ho in 1969. This method is easy to understand and easy to program. It is usually used for decision making and pattern recognition. In this approach, the error is propagated backwards according to the delta rule. One way to adjust weights in back-propagation algorithm is to use gradient descent learning rule, which is called Delta Rule. In the backward mode, the update of the weight is accomplished by passing error backward from the output layer to the input layer. In this rule, the partial derivative is calculated to adjust the weights to decrease the error [31-34].

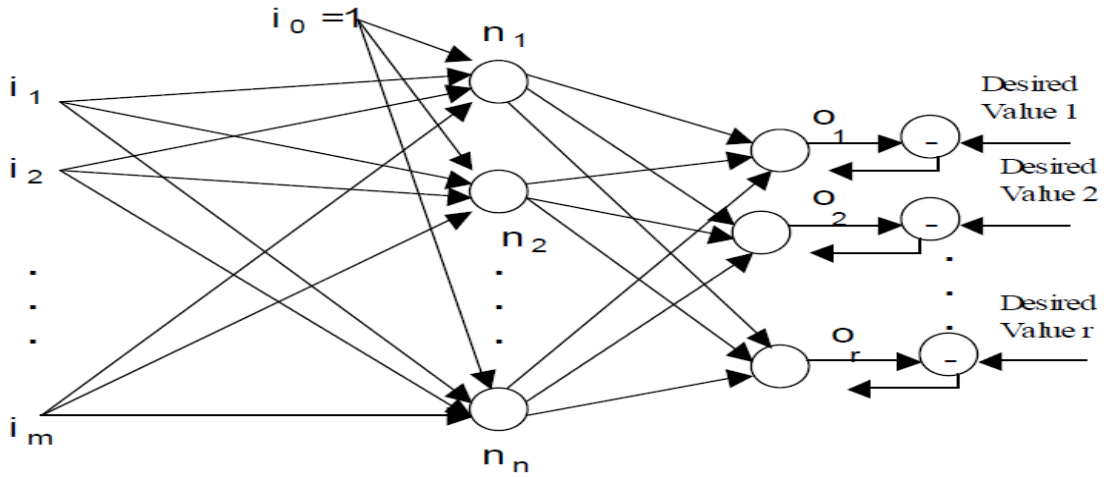


Figure 2.16 Back-propagation algorithm for multilayer feed forward network

2.7.8.1 Algorithm of Back-propagation Algorithm

1. Normalize the inputs and outputs with respect to their maximum value
2. For each training pair, assume there are *l* inputs and *n* outputs
3. Number of neurons in the hidden layers are $l < m < 2l$
4. Initialize small random weights for hidden layers (V) and weights for output layers (W) ranging from -1 to 1.
5. The output of input layer is the linear function of the inputs, so $\{O\}_I = \{I\}_I$
6. Input to hidden layer is calculated as $\{I\}_H = V^T \{O\}_I$
7. Output of hidden layer is computed as $\{O\}_H = \frac{1}{1+e^{-I_{Hi}}}$
8. Input to output layer is calculated as $\{I\}_o = W^T \{O\}_H$
9. Output of output layer is computed as $\{O\}_o = \frac{1}{1+e^{-I_{oj}}}$

10. Error is calculated as $E = \frac{\sqrt{\sum (T-O)^2}}{N}$

Here T is the target output, O is the actual output computed by the network and N is the number of training set.

11. $d = (T - O_o) O_o (1 - O_o)$

12. $Y = \{O\}_H \, d$

13. Change in weight (Output-Hidden) is $\Delta W^{t+1} = \alpha \Delta W^t + \eta Y$

14. $e = Wd$

15. $d^* = eO_H (1 - O_H)$

16. $X = \{O\}_I \, d^* = \{I\}_I \, d^*$

17. Change in weights (Hidden-Inputs) is $\Delta V^{t+1} = \alpha \Delta V^t + \eta Y$

18. Weights for next iterations are $[V]^{t+1} = [V]^t + [\Delta V]^{t+1}$ and $[W]^{t+1} = [W]^t + [\Delta W]^{t+1}$

19. Repeat the weight updating steps till the error is in the tolerance limit

2.7.8.2 Performance Parameters of Back Propagation

To calculate the overall training accuracy and testing accuracy of the network, the following formulas are used.

Training Accuracy = $\frac{\textbf{number of correct actual outputs}}{\textbf{total number of train samples}}$

Test Accuracy = $\frac{\textbf{number of correct actual outputs}}{\textbf{total number of test cases}}$

Chapter 3

Unsupervised Classification: Clustering

3.1 Clustering

Clustering is an unsupervised classification technique used to classify data into groups of similar objects [35]. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. In supervised classification, we are provided with a collection of labeled (pre classified) patterns; the problem is to label a newly encountered, yet unlabeled, pattern. Typically, the given labeled (training) patterns are used to learn the descriptions of classes which in turn are used to label a new pattern. In the case of clustering, the problem is to group a given collection of unlabeled patterns into meaningful clusters. In a sense, labels are associated with clusters also, but these category labels are data driven; that is, they are obtained solely from the data [36].

3.2 Application of Clustering

Clustering plays an outstanding role in data mining applications such as:

1. Scientific data exploration,
2. Information retrieval and text mining,
3. Spatial database applications,
4. Web analysis
5. Marketing,
6. Medical diagnostics

3.3 Cluster Analysis

Cluster analysis is the organization of a collection of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into clusters based on similarity. It may reveal associations and structure in data which, though not previously evident, nevertheless are sensible and useful once found [36]. The results of cluster analysis may contribute to the definition of a formal classification scheme, such as a taxonomy for related process variables, parameters or objects; or suggest statistical models with which to describe control; or indicate rules for assigning new cases to classes for control and analysis

purposes; or provide measures of definition, variations and change in what previously were only broad concepts.

3.4 Types of Clustering

The taxonomy shown in Figure 3.3 must be supplemented by a discussion of cross-cutting issues that may (in principle) affect all of the different approaches regardless of their placement in the taxonomy. Different types of Clustering are as follows [38].

3.4.1 Monothetic vs. Polythetic

This aspect relates to the sequential or simultaneous use of features in the clustering process. Most algorithms are polythetic; that is, all features enter into the computation of distances between patterns, and decisions are based on those distances. A simple monothetic algorithm considers features sequentially to divide the given collection of patterns. Here, the collection is divided into two groups using feature *x*1; the vertical broken line V is the separating line. Each of these clusters is further divided independently using feature *x*2, as depicted by the broken lines *H*1 and *H*2. The major problem with this algorithm is that it generates 2*d* clusters where *d* is the dimensionality of the patterns. For large values of *d* (*d* . 100 is typical in information retrieval applications), the number of clusters generated by this algorithm is so large that the data set is divided into uninterestingly small and fragmented clusters.

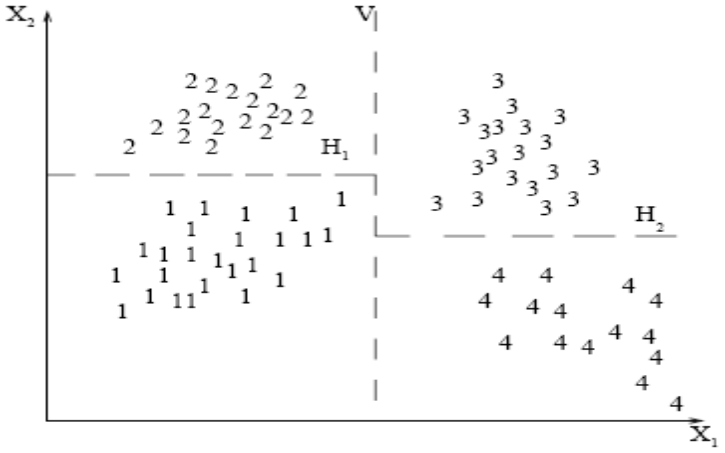


Figure 3.1 Monothetic partitional clustering.

3.4.2 Hard vs. Fuzzy

A hard clustering algorithm allocates each pattern to a single cluster during its operation and in ts output. A fuzzy clustering method assigns degrees of membership in several clusters to each input pattern. A fuzzy clustering can be converted to a hard clustering by assigning each pattern to the cluster with the largest measure of membership

3.4.3 Deterministic vs. Stochastic

This issue is most relevant to partitional approaches designed to optimize a squared error function. This optimization can be accomplished using traditional techniques or through a random search of the state space consisting of all possible labeling.

3.4.4 Incremental vs. Non-incremental

This issue arises when the pattern set to be clustered is large, and constraints on execution time or memory space affect the architecture of the algorithm. The early history of clustering methodology does not contain many examples of clustering algorithms designed to work with large data sets, but the advent of data mining has fostered the development of clustering algorithms that minimize the number of scans through the pattern set, reduce the number of patterns examined during execution, or reduce the size of data structures used in the algorithm's operations.

3.5 Issues of Clustering

The main requirements that a clustering algorithm should satisfy are the following

1. **Dealing with different types of attributes**

   There are various attributes of data that any clustering algorithm need to satisfy, the most general taxonomy being in common use distinguishes among numeric (continuous), ordinal, and nominal variables. A numeric variable can assume any value in R. An ordinal variable assumes a small number of discrete states, and these states can be compared.

2. **Scalability to large datasets**

   The data sets could be in any possible range, varying between large extremes and they need to be normalized by the clustering algorithm.

3. **Ability to work with high dimensional data**

   The data could be multidimensional varying from 1, 2……n.; depending on the application data on which clustering is being applied.

4. **Ability to find clusters of irregular or arbitrary shape**

   The shape of clusters could be any arbitrary shapes. We prefer using Euclidean distance to get a circular shape of the clusters, but still shape of clusters can not be accurately defined

5. **Handling outliers**

   The data points on the boundary of clusters need to be handled; this is done in hierarchical methods by associating the boundary points to one of the clusters. While in fuzzy clustering, we associate membership functions to the points lying on the boundary of clusters.

6. **Time complexity**

   Complexity of the data points in terms of time has to be taken care of while clustering.

7. **Data order dependency**

   Dependency o data points on other variable can affect the clustering of data and there by the cluster centers too, so it has to be taken care before hand.

3.6 Distance Measures

Some of the distance measures are shown below:

1. Euclidean distance: $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

2. Hamming distance: $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$

3. Minkowski distance: $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^p}, p > 0$

3.7 Components of a Clustering Task

Typical pattern clustering activity involves the following steps:

1. Pattern representation (optionally including feature extraction and/or selection),
2. Definition of a pattern proximity measure appropriate to the data domain,
3. Clustering or grouping,
4. Data abstraction (if needed), and
5. Assessment of output (if needed).

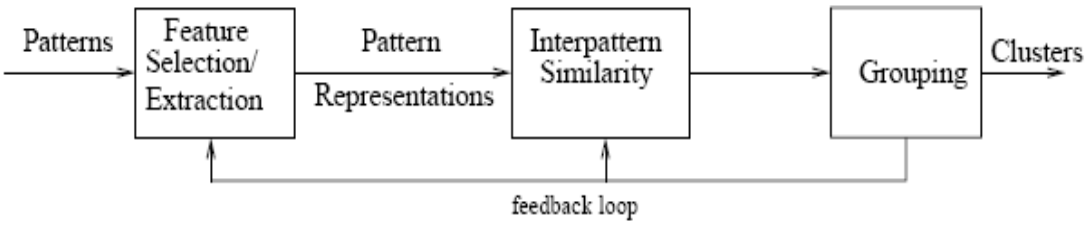


Figure 3.2 Stages of Clustering

Figure 3.2 above depicts a typical sequencing of the first three of these steps, including a feedback path where the grouping process output could affect subsequent feature extraction and similarity computations [39, 40]. Pattern representation refers to the number of classes, the number of available patterns, data, and the number, type, and scale of the features available to the clustering algorithm. Some of this information may not be controllable by the practitioner.

3.7.1 Feature Selection

Feature selection is the process of identifying the most effective subset of the original features to use in clustering. Feature extraction is the use of one or more transformations of the input features to produce new salient features. Either or both of these techniques can be used to obtain an appropriate set of features to use in clustering.

3.7.2 Pattern Proximity

Pattern proximity is usually measured by a distance function defined on pairs of patterns. A variety of distance measures are in use in the various communities. A simple

distance measure like Euclidean distance can often be used to reflect dissimilarity between two patterns, whereas other similarity measures can be used to characterize the conceptual similarity between. The grouping step can be performed in a number of ways. The output clustering (or clustering) can be hard (a partition of the data into groups) or fuzzy (where each pattern has a variable degree of membership in each of the output clusters). Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity. Partitional clustering algorithms identify the partition that optimizes (usually locally) a clustering criterion. Additional techniques for the grouping operation include probabilistic and graph-theoretic clustering methods.

3.7.3 Data Abstraction

Data abstraction is the process of extracting a simple and compact representation of a data set. Here, simplicity is either from the perspective of automatic analysis (so that a machine can perform further processing efficiently) or it is human-oriented (so that the representation obtained is easy to comprehend and intuitively appealing). In the clustering context, a typical data abstraction is a compact description of each cluster, usually in terms of cluster prototypes or representative patterns such as the centroid. How is the output of a clustering algorithm evaluated? What characterizes a ‘good’ clustering result and a ‘poor’ one? All clustering algorithms will, when presented with data, produce clusters — regardless of whether the data contain clusters or not. If the data does contain clusters, some clustering algorithms may obtain ‘better’ clusters than others. The assessment of a clustering procedure’s output, then, has several facets. One is actually an assessment of the data domain rather than the clustering algorithm itself— data which do not contain clusters should not be processed by a clustering algorithm. The study of cluster tendency, wherein the input data are examined to see if there is any merit to a cluster analysis prior to one being performed, is a relatively inactive research area, and will not be considered further in this survey.

3.7.4 Cluster Validity

Cluster validity analysis, by contrast, is the assessment of a clustering procedure’s output [40]. Often this analysis uses a specific criterion of optimality; however, these criteria are usually arrived at subjectively. Hence, little in the way of ‘gold standards’ exist in clustering except in well-prescribed sub domains. Validity assessments are objective and are

performed to determine whether the output is meaningful. A clustering structure is valid if it cannot reasonably have occurred by chance or as an artifact of a clustering algorithm. When statistical approaches to clustering are used, validation is accomplished by carefully applying statistical methods and testing hypotheses. There are three types of validation studies. An external assessment of validity compares the recovered structure to an apriori structure. An internal examination of validity tries to determine if the structure is intrinsically appropriate for the data. A relative test compares two structures and measures their relative merit.

3.7.5 Cluster Validity Assessment

The process of evaluating the results of a clustering algorithm is called cluster validity assessment. Two measurement criteria have been proposed for evaluating and selecting an optimal clustering scheme.

3.7.5.1 Compactness

The member of each cluster should be as close to each other as possible. A common measure of compactness is the variance.

3.7.5.2 Separation

The clusters themselves should be widely separated. There are three common approaches measuring the distance between two different clusters: distance between the closest member of the clusters, distance between the most distant members and distance between the centers of the clusters.

There are three different techniques for evaluating the result of the clustering algorithms

- External Criteria
- Internal Criteria
- Relative Criteria

3.8 Indexes of clustering

The various indexes that are linked with the clustering are as given below:

3.8.1 Partition Coefficient (PC)

It measures the amount of "overlapping" between clusters. It is described below:

$$PC(c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}\left(\mu_{ij}\right)^2$$

Where μ_{ij} is the membership of data point *j* in cluster *i*. The disadvantage of PC is lack of direct connection to some property of the data themselves. The optimal number of cluster is at the maximum value.

3.8.2 Classification Entropy (CE)

It measures the fuzziness of the cluster partition only, which is similar to the Partition Coefficient

$$CE(c) = -\frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}\left(\mu_{ij}\right)\log\left(\mu_{ij}\right)$$

3.8.3 Partition Index (SC)

It is the ratio of the sum of compactness and separation of the clusters. It is a sum of individual cluster validity measures normalized through division by the fuzzy cardinality of each cluster.

$$SC(c) = \sum_{i=1}^{c}\frac{\sum_{j=1}^{N}\left(\mu_{ij}\right)^m \left\|x_j - v_i\right\|}{N_i \sum_{k=1}^{c}\left\|v_k - v_i\right\|^2}$$

SC is useful when comparing different partitions having equal number of clusters. A lower value of SC indicates a better partition.

3.8.4 Separation Index (S)

On the contrary of partition index (SC), the separation index uses a minimum-distance separation for partition validity.

$$S(c) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{N}\left(\mu_{ij}\right)^2 \left\|x_j - v_i\right\|^2}{N \min_{i,k}\left\|v_k - v_i\right\|^2}$$

3.8.5 Xie and Beni's Index (XB)

It aims to quantify the ratio of the total variation within clusters and the separation of clusters.

$$XB(c) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{N}(\mu_{ij})^{m}\|x_j - v_i\|^2}{N \min_{i,k}\|v_k - v_i\|^2}$$

The optimal number of clusters should minimize the value of the index.

3.8.6 Dunn's Index (DI)

This index is originally proposed to use at the identification of "compact and well separated clusters". So the result of the clustering has to be recalculated as it was a hard partition algorithm.

$$DI(c) = \min_{i\in c}\left\{\min_{j\in c, i\neq j}\left\{\frac{\min_{x\in C_i, y\in C_j} d(x,y)}{\max_{k\in c}\left\{\max_{x,y\in C} d(x,y)\right\}}\right\}\right\}\backslash$$

The main drawback of Dunn's index is computational since calculating becomes computationally very expansive as c and N increase.

3.9 Taxonomy of clustering

At the very high end of the overall taxonomy we envision two main categories of clustering, known as hierarchical and objective function-based clustering. At the top level, there is a distinction between hierarchical and partition approaches. These two clustering methods are further sub divided as shown in figure below. Different approaches to clustering data can be described with the help of the hierarchy shown in figure.

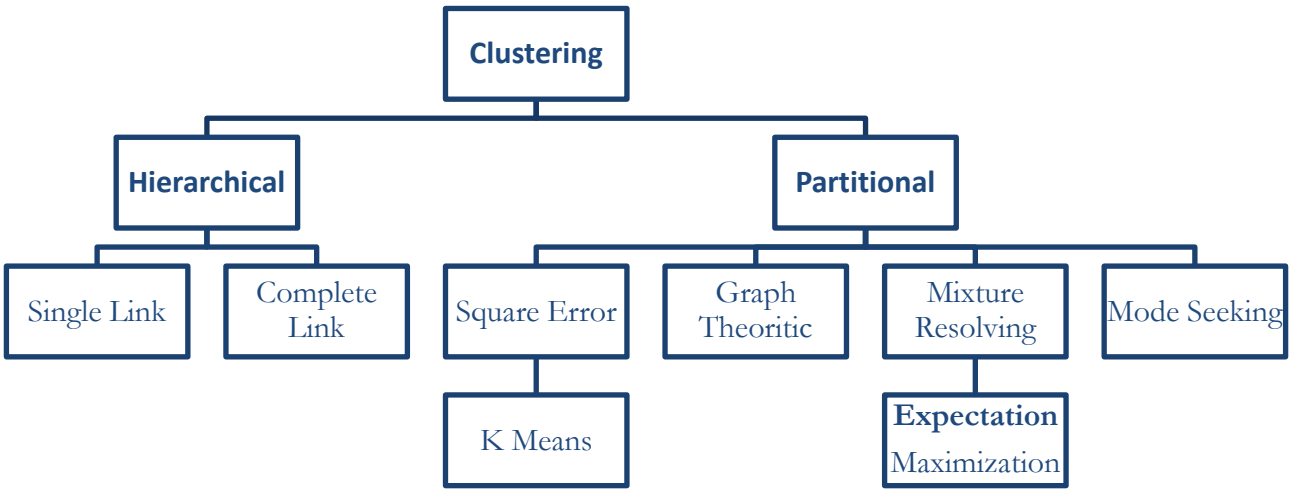


Figure 3.3 Taxonomy of Clustering Approaches

3.10 Nearest Neighbor Clustering

Since proximity plays a key role in our intuitive notion of a cluster, nearest neighbor distances can serve as the basis of clustering procedures. An iterative procedure; it assigns each unlabeled pattern to the cluster of its nearest labeled neighbor pattern, provided the distance to that labeled neighbor is below a threshold. The process continues until all patterns are labeled or no additional labeling occurs. The mutual neighborhood value (described earlier in the context of distance computation) can also be used to grow clusters from near neighbors.

3.11 K-Means Clustering

Let $X = \{x_1, x_2, x_3.....x_n\}$ is a set of n dimensional objects to be clustered in to k clusters represented by $C = \{c_1, c_2, c_3.....c_k\}$. K Means clustering algorithm finds a partition such that the squared error between the empirical mean of the cluster and points in the cluster is minimized [41]. Let μ_k is the mean of cluster c_k. The squared error between μ_k and c_k is defined as:

$$J(c_k) = \sum_{x_i \in c_k} \|x_i - \mu_k\|^2$$

The objective of K means algorithm is to find the centroid [42] and minimize the sum of squared error over all K clusters. So the minimization function is written as

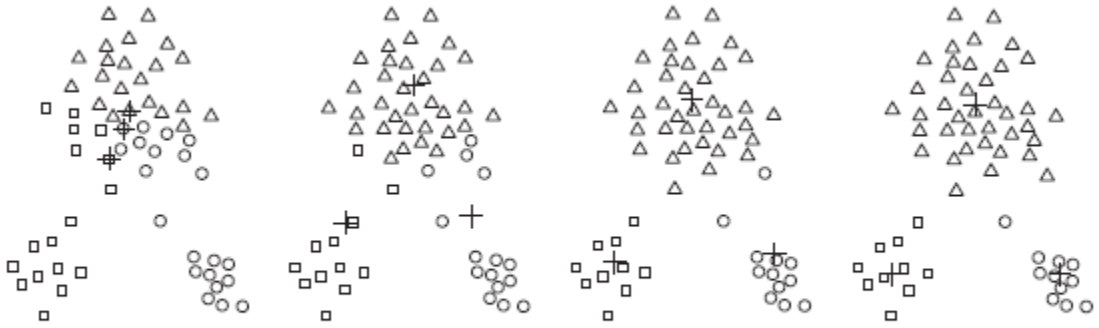
$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \|x_i - \mu_k\|^2$$

Minimizing this objective function is known to be an NP-hard problem (even for K = 2). Thus K-means, which is a greedy algorithm, can only converge to a local minimum, even though recent study has shown with a large probability K-means could converge to the global optimum when clusters are well separated [44, 45]. K-means starts with an initial partition with K clusters and assign patterns to clusters so as to reduce the squared error. Since the

squared error always decrease with an increase in the number of clusters K (with J(C) = 0 when K = n), it can be minimized only for a fixed number of clusters. The main steps of K-means algorithm are as [43]:

3.11.1 Algorithm of K means algorithm

1. Select K points as initial centroids.
2. Repeat.
3. Form K clusters by assigning each point to its closest centroid
4. Recompute the centroid of each cluster
5. Until centroid do not change



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Figure 3.4 Different iteration to find out 3 clusters of a sample data

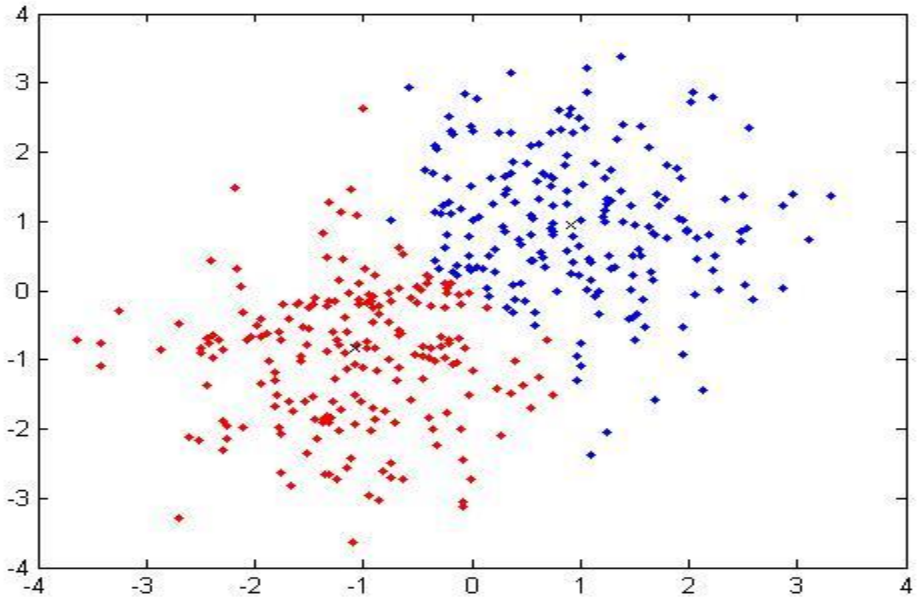


Figure 3.5 K-Means algorithm implemented in sample data

The simple K-means partition clustering algorithm described above is computationally efficient and gives surprisingly good results if the clusters are compact, hyper spherical in shape and well-separated in the feature space. If the Mahalanobis distance is used in defining the squared error, then the algorithm is even able to detect hyper ellipsoidal shaped clusters. Numerous attempts have been made to improve the performance of the basic K-means algorithm by

1. Incorporating a fuzzy criterion function, resulting in a fuzzy K-means (or c-means) algorithm,
2. Using genetic algorithms, simulated annealing, deterministic annealing, and tabu search to clustering Algorithms optimize the resulting partition
3. Mapping it onto a neural network for possibly efficient implementation.

3.12 Fuzzy Clustering

In fuzzy clustering, one object can be clustered in more than one cluster according to the degree of membership function.

Let a set of objects $X = \{x_1, x_2, x_3 ..... x_n\}$ has to be clustered in to $C = \{c_1, c_2, c_3 ..... c_k\}$. $\delta(x, C_i)$ Denote the similarity between object x and cluster C_i. The membership function for object x and cluster C_i is represented by the following equation

$$fc_i(x) = \frac{P_i \delta(x, C_i)}{\sum_{k=1}^{K} P_k \delta(x, C_k)}$$

$P_k = \frac{n_k}{n}$ is the relative size of cluster C_k. This membership function is non negative [46]. Membership function can also be expressed in terms of Euclidian distance. This is represented in following equation

$$fc_k(x) = \frac{1 - \left(\frac{1}{\beta}\right) d\left(x, m^k\right)}{K - \left(\frac{1}{\beta}\right) \sum_j d\left(x, m^j\right)}$$

$d\left(x, m^k\right)$ Represent the Fuzzy distance [48] between vector x and centroid m^k of cluster C_k. β denotes the belongingness. Traditional clustering approaches generate partitions; in a partition, each pattern belongs to one and only one cluster. Hence, the clusters in a hard clustering are disjoint. Fuzzy clustering extends this notion to associate each pattern with every cluster using a membership function [47]. The output of such algorithms is a clustering, but not a partition. Contrary to other methods of clustering, the fuzzy clustering methods provide a number of membership values that indicate the degree of membership of the different samples to the different groups.

3.13 Fuzzy C Means

Fuzzy C Means (FCM) is a feature clustering technique wherein each feature point belongs to a cluster by some degree that is specified by a membership grade. These kind of clustering algorithms are known as objective function based clustering [49]. Given M dimensional database of size N where N is the total number of feature vectors and M is the dimension of each feature vector. FCM assigns every feature vector a membership grade for each cluster. The problem is to partition the database based on some fuzziness criteria using

membership values. To find membership values, the partition matrix U of size N x C is calculated that defines membership degrees of each feature vector. The values 0 and 1 in U indicate no membership and full membership respectively. Grades between 0 and 1 indicate that the feature point has partial membership in a cluster. Looking at the picture, we may identify two clusters in proximity of the two data concentrations. We will refer to them using "A" and "B". In the first approach shown in this tutorial - the k-means algorithm - we associated each datum to a specific centroid; therefore, this membership function [51] looked like this:

3.13.1 Initialization of the partition matrix

Initially a fuzzy partition matrix U is generated that is of size N x c, where c is number of clusters and N is total number of feature vectors. Subject to the constraint that

$$\sum_{j=1}^{c} U_{ij} = 1$$

i = {1, 2, 3 -------- N}

3.13.2 Calculation of fuzzy centers

The fuzzy centers are calculated using the partition matrix generated by:

$$C_j = \frac{\sum_{i=1}^{N} U_{ij}^m x_i}{\sum_{i=1}^{N} U_{ij}^m}$$

where $m \geq 1$ is a fuzzification exponent. The larger the value of m the fuzzier the solution will be. This indicates the number of iterations that is required for clustering. xi is ith feature vector. The value of i ranges from 1 to N (total number of templates in the database).

3.13.3 Updating membership and cluster centers

FCM is an iteration loop. The method of clustering is based on minimization of the objective function defined by

$$J = \sum_{i=1}^{N} \sum_{j=1}^{C} U_{ij}^m \left\| x_i - c_j \right\|^2$$

Uij describes the degree of member of feature set (*xi*) with cluster *cj*. |*| represents norm between *xi* and cluster center *cj* given by $\|x_i - c_j\|^2 = (x_i - c_j)^T A(x_i - c_j)$

where *A* is identity matrix for Euclidean distance used here [50]. At every iteration the membership matrix is updated using

$$U_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_j\|} \right)^{\frac{2}{m-1}}}$$

The revised membership matrix is used for updating the cluster centers.

The iteration will stop when $\max_{ij} \left\{ \left| U_{ij}^{m+1} - U_{ij}^{m} \right| \right\} < \varepsilon$

where ε is a termination criteria. The value of ε ranges between 0 and 1.

3.13.4 Algorithm

1. Fix $1 < m < \infty$, initial partition matrix U (N x c) and termination criteria
2. Calculate fuzzy cluster centers
3. Update membership matrix
4. Calculate change in membership function $\Delta = \|U^{m+1} - U^m\| = \max_{ij} |U_{ij}^{m+1} - U_{ij}^m|$

   If $\Delta \leq \varepsilon$ then set m = m+1 and go to step 2 Or else stop

Chapter 4
Literature Review

Yasuhiko Dote gave brief introduction to fuzzy logic control and its design procedure. He implemented fuzzy logic controller on an automobile system to control the speed of system and made a comparative study between fuzzy logic system and neuro system [4].

Joachim Stender began his work on genetic algorithm at the beginning of 90s. And he gets his first success when he published paper on introduction to genetic algorithm and implementation of genetic algorithm in natural and artificial system. The basic idea of his research was that the genetic pool of given population potentially contains the solution or a better solution to a given adaptive problem [12].

J.C. Eccles was an Australian neurophysiologist who won a noble prize for his work on synapse. In his research paper he gave the complete description on working of human brain [18].

G.X. Ritter and G. Urcid in their research change a simplified model of a single neuron with a more realistic one that incorporates the dendritic processes; a novel paradigm in artificial neural networks is being established. In their work, they introduce and develop a mathematical model of dendrite computation in a morphological neuron based on lattice algebra. [22].

Yanjun Pang in his paper introduce a special supervised learning algorithm by assuming that if there is only one normal sample in every class, then the learning stage is omitted. He gave a new algorithm by using membership as similarity measure between index value and classes [27].

El Adawy introduced a new hybrid Soft backpropagation algorithm where the Soft algorithm is applied first to obtain an initially good weight vector. This vector will be introduced to the back propagation algorithm, which improves the precession of the weight vector to reach an acceptable error limit. The results show an acceptable improvement in the training speed for the hybrid technique as compared with the individual back propagation [31].

Jiamthapthaksin addresses two main challenges for clustering which require extensive human effort: selecting appropriate parameters for an arbitrary clustering algorithm and identifying

alternative clusters. He proposes an architecture and a concrete system MR-CLEVER for multi-run clustering that integrates active learning with clustering algorithms [36].

Raza Ali Data dissusses Clustering along with its two traditional approaches and their analysis. Some applications of Data Clustering like Data Mining using Data Clustering and Similarity Searching in Medial Image Databases are also discussed along with a case study of Microsoft Windows NT Operating system. The implementation of clustering in NT is also discussed [38].

Sun Jigui, Liu Jie, Zhao Liany proposes an improved k-means algorithm which require a simple data structure to store some information in every iteration, which is to be used in the next interation.The improved method avoids computing the distance of each data object to the cluster centers repeatly, saving the running time [43].

Miyamoto gave introduction to recently algorithms for fuzzy clustering and related methods. After a short introduction to fuzzy clustering he described three topics of kernel functions, sequential algorithms, and cluster validity measures [46].

R Dunia et.al, has proposed the use of PCA for sensor fault identification. The principal component model captures measurement correlation and reconstructs the variables using optimization techniques. The status of the sensor is determined by sensor validity index [53].

Y M Sebzalli et.al, has proposed two techniques like principal component analysis (PCA) and fuzzy C means clustering to identify and develop operational strategy for manufacture of desired product in process industry. This research paper takes a case study of fluid catalytic cracking process used in refinery industry. The authors analyzed the problem by collecting three hundred data from the process site and applying principal component analysis and fuzzy c means clustering algorithm in the datasets [54].

N Bendwell et.al performed a comparative study of multidimensional visualization techniques and multivariate statistical process control for process historical data analysis. The study was carried out for a waste water treatment plant [55].

David L. Hall provides a tutorial on data fusion, introducing data fusion applications, process models, and identification of applicable techniques. He also discussed the applications of data fusion in Department of Defense (DoD) such as battlefield surveillance and automatic

target recognition for smart weapons to non-DoD applications it has applicability in condition-based maintenance and improved medical diagnosis [59].

V.Venkatasubramaniam et.al in his pioneer work reviews different quantitative model based methods [66], qualitative methods and search strategies [67] and process history based methods [68] to efficiently detect process faults.

Vasil Simeonov et.al, has proposed a novel method of water quality assessment of high mountain lakes in Pirin Mountain in Bulgaria by application of cluster analysis and principal component analysis. The authors have also studied the classification of dataset by using self organizing map[69]

Chapter 5

Dimensionality Reduction: Principal Component Analysis

5.1 Introduction

Principle Component Analysis (PCA) was invented in 1901 by Karl Pearson. It is mostly used as a tool to analyzing data sets to summarize their main characteristics in easy-to-understand form. Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible [52]. We can also define it as way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analyzing data. The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, i.e. by reducing the number of dimensions, without much loss of information.

There are many cases arises in our system when the dimension of input data is very large and it became very difficult for us to deal with such kind of data because data is originated from large number of sensors [59]. Data fusion techniques combine data from multiple sensors [60, 61], and related information from associated databases, to achieve improved accuracies and more specific inferences than could be achieved by the use of a single sensor alone [63, 64]. Primary thing which we have to do is that we have to find that whether there is any kind of redundancy in our data or not. We can discard this redundant data and keep the rest of data. Let us suppose that m-dimensional input data is available to us:

$$\vec{x} = x_1, x_2, x_3 \ldots\ldots x_n, x_{n+1}, \ldots\ldots x_{m-1}, x_m$$

We suppose that m is a very large number and we do not want to deal with such a large number of dimensions. We want that the data of n-dimension should be there and rest all truncated. We can not do this just by keeping data up to n-dimension in our system and

discard rest of data. Because it may be possible that the data after n-dimension contains lot of information. So we have to find the technique which we can use to convert this m-dimensional data into n-dimensional without much loss of information and that technique is PCA. In this case Mean Square Error is equal to sum of variances of the elements that are eliminated.

$$\text{MSE} = \text{Sum of Variances of the Elements that are Eliminated.}$$

So we truncate elements in such a way that it keeps the limit of MSE in tolerable range [57]. This is achieved by transforming the $\vec{x}$ of m-dimension into the $\vec{X}$ of n-dimension in such a way that information loss is minimum. Before applying this transformation one should have knowledge about the following things:

5.2 Standard Deviation

The Standard Deviation (SD) of a data set is a measure of how spread out the data is. It is defined as "The average distance from the mean of the data set to a point". The way to calculate it is to compute the squares of the distance from each data point to the mean of the set, add them all up divide by n-1and take the positive square root. As a formula:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}}$$

Where s is the usual symbol for standard deviation of a sample. This will be clearer to you by following example:

Let us suppose that we have two data sets

A= [0 8 12 20], B= [8 9 11 12]

It may be notice here that both the two data sets have same mean but their standard deviation is as follows:

Table 5.1 Calculation of Standard Deviation for SET 1

X	$(X - \bar{X})$	$(X - \bar{X})^2$
0	-10	100
8	-2	4
12	2	4
20	10	100
Total		208
Divided by (n-1)		69.333
Square Root		8.3266

Table 5.2 Calculation of Standard Deviation for SET 2

X_i	$(X_i - \bar{X})$	$(X_i - \bar{x})^2$
8	-2	4
9	-1	1
11	1	1
12	2	4
Total		10
Divided by (n-1)		3.333
Square Root		1.8257

And so, as expected, the first set has a much larger standard deviation due to the fact that the data is much more spread out from the mean.

5.3 Variance

Variance is another measure of the spread of data in a data set. In fact it is almost identical to the standard deviation. The formula is this:

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

You will notice that this is simply the standard deviation squared, in both the symbol (s^2) and the formula (there is no square root in the formula for variance). s^2 Is the usual symbol for variance of a sample? Both these measurements are measures of the spread of the data. Standard deviation is the most common measure, but variance is also used.

5.4 Covariance

The last two measures we have looked at are purely 1-dimensional. Data sets like this could be: heights of all the people in the room, marks for the last exam etc. However many data sets have more than one dimension, and the aim of the statistical analysis of these data sets is usually to see if there is any relationship between the dimensions. For example, we might have as our data set both the height of all the students in a class, and the mark they received for that paper. We could then perform statistical analysis to see if the height of a student has any effect on their mark. Standard deviation and variance only operate on 1 dimension, so that you could only calculate the standard deviation for each dimension of the

data set independently of the other dimensions. However, it is useful to have a similar measure to find out how much the dimensions vary from the mean with respect to each other. Covariance is such a measure. Covariance is always measured between 2 dimensions. If you calculate the covariance between one dimension and itself, you get the variance. So, if you had a 3-dimensional data set (x, y, z), then you could measure the covariance between the x and z dimension, x and y dimension, y and z dimension. The formula for covariance is very similar to the formula for variance. The formula for variance could also be written like this:

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

Where I have simply expanded the square term to show both parts. So given that knowledge, here is the formula for covariance:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

The concept of covariance will be clearer to you by following example.

A plot of the covariance data showing positive relationship between the numbers of hours studied against the mark received is as shown below:

Table 5.3 Relation between number of Hours and Marks received

	$Hours(H)$	$Mark(M)$
DATA	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42

Table 5.4 Calculation of Variance of data

H	M	$(H_i - H)$	$(M_i - M)$	$(H_i - H)(M_i - M)$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

5.5 The covariance Matrix

Recall that covariance is always measured between 2 dimensions. If we have a data set with more than 2 dimensions, there is more than one covariance measurement that can be calculated. For example, from a 3 dimensional data set(x, y, and z) you could calculate cove(x, y), cove(x, z), cove(y, z). In fact, for an n-dimensional data set, you can calculate $\frac{n!}{(n-2)!*2}$ different covariance values. A useful way to get all the possible covariance values between all the different dimensions is to calculate them all and put them in a matrix. So, the definition for the covariance matrix for a set of data with n dimensions is:

$$C^{m\times n} = \left(c_{i,j}, c_{i,j} = cov\left(D_i m_i, D_i m_j\right)\right)$$

An example. We'll make up the covariance matrix for an imaginary 3 dimensional data set, using the usual dimensions x, y and z. Then, the covariance matrix has 3 rows and 3 columns, and the values are this:

$$C = \begin{pmatrix} cov(x,x)\ cov(x,y)\ cov(x,z) \\ cov(y,x)\ cov(y,y)\ cov(y,z) \\ cov(z,x)\ cov(z,y)\ cov(z,z) \end{pmatrix}$$

5.6 Matrix Algebra

This section serves to provide a background for the matrix algebra required in PCA. Specifically I will be looking at eigenvectors and Eigen values of a given matrix.

5.6.1 Eigenvectors

As you know, you can multiply two matrices together, provided they are compatible sizes. Eigenvectors are a special case of this. Consider the two multiplications between a matrix and a vector as shown below:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

In this example the resulting vector is not an integer multiple of the original Vector, whereas in the following example:

$$2 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

The example is exactly 4 times the vector we began with. Why is this? Well, the vector is a vector in 2 dimensional spaces. The vector ($^{3}/_{2}$) represents an arrow pointing from the origin, (0, 0) to the point (3, 2). The other matrix, the square one, can be thought of as a transformation matrix. If you multiply this matrix on the left of a vector, the answer is another vector that is transformed from its original position. It is the nature of the transformation that the eigenvectors arise from. Important thing to know is that when mathematicians find eigenvectors.

They like to find the eigenvectors whose length is exactly one. This is because, as you know, the length of a vector doesn't affect whether it's an eigenvector or not, whereas the direction does. So, in order to keep eigenvectors standard, whenever we find an eigenvector we usually

scale it to make it have a length of 1, so that all eigenvectors have the same length. Here's a demonstration from our example above.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Is an eigenvector, and the length of that vector is:

$$\sqrt{(3^2 + 2^2)} = \sqrt{13}$$

So we divide the original vector by this much to make it have a length of 1.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3/\sqrt{13} \\ 2/\sqrt{13} \end{pmatrix}$$

5.6.2 Eigen values

Eigen values are closely related to eigenvectors, in fact, we saw an eigenvalue in our example. Notice how, in both those examples, the amount by which the original vector was scaled after multiplication by the square matrix was the same? In that example, the value was 4. 4 is the eigenvalue associated with that eigenvector. No matter what multiple of the eigenvector we took before we multiplied it by the square matrix, we would always get 4 times the scaled vector as our result. So you can see that eigenvectors and eigenvalues always come in pairs. When you get a fancy programming library to calculate your eigenvectors for you, you usually get the eigenvalues as well.

5.7 Mathematical analysis of Principle Component Analysis

Let us suppose that our input vector is $\vec{x}$ of m dimension. And we want to truncate some elements from the m dimension vector so that the final vector will be of n dimension. This is achieved by considering the transformation matrix T [53 54] and we multiply the given input vector with this transformation matrix.

$$\vec{T} * \vec{\mathrm{X}} = \vec{\mathrm{X}}$$

Now it may be notice here that the elements in the final $\vec{\mathrm{X}}$ will be in such a way that they are in descending order according to their variances i.e. element with highest variance comes first and elements with low variance comes at last. By this we can easily truncate the elements of low variances. So in PCA main task is to design this transformation matrix T [56, 57]. For the designing of T we assume that:

$E[\vec{x}] = 0$

$\vec{x}$ = input vector

$E[\vec{x}]$ = expectation of input vector.

Now it my be possible that the expectation of $\vec{x}$ is not always equal to zero because not every time the summation of positive and negative term will be equal to zero, to make it zero we calculate the mean of vector and subtract it from every value. We suppose that a new vector:

$\vec{q}$ = m dimensional unit vector.

And we have to do is we have to project $\vec{x}$ onto this $\vec{q}$. Meaning of projection is that we require 3-dimensional space or co-ordinate system to represent or project a point of 3-dimensions similarly we require m-dimensional system to project our input vector. As we are assuming that $\vec{q}$ is a unit vector that means Euclidian of $\vec{q}$ is always 1 [58]. Let us suppose that A is projection of $\vec{x}$ onto $\vec{q}$. That means

$A = \vec{q}\,\vec{x}^T$ or $\vec{q}^T\vec{x}$

Now $E[A] = 0$

$\because E[A] = E[\vec{q}^T\vec{x}] = 0 \qquad [\because E[\vec{x}] = 0]$

The variance of this quantity σ^2 =

$E[A^2] = E[(\vec{q}^T\vec{x})(\vec{x}^T\vec{q})]$

$= \vec{q}^T E(\vec{x}\vec{x}^T)\vec{q}$

$= \vec{q}^T\vec{R}\vec{q}$

Where in the above equation $\vec{R}$ is the co-relation matrix symmetric matrix. We know that for symmetric matrix:

$$\vec{a}^T\vec{R}\vec{b} = \vec{b}^T\vec{R}\vec{a}$$

But here we are more interested in variance expression because in this our main objective is to maximize the rate of decrease of variance. We do not have any control on $\vec{R}$, so with a proper choice of $\vec{q}$ we can go for minimum variance. That means $\vec{q}$ is a tool in our hand that we can vary to minimize the variance. Now:

$$\sigma^2 = f\{\vec{q}\} = \vec{q}^T\vec{R}\vec{q} \qquad (1)$$

for minimizing the variance we have to minimize the function $f\{\vec{q}\}$. For minimum value of $f\{\vec{q}\}$ we know that:

$$f\{\vec{q}\} = f\{\vec{q}+\Delta\vec{q}\} \qquad (2)$$

put the value of equation (2) in equation (1).

$$f\{\vec{q}+\Delta\vec{q}\} = \vec{q}^T\vec{R}\vec{q} + 2(\Delta\vec{q})^T\vec{R}\vec{q} + (\Delta\vec{q})^T\Delta\vec{q}\,\vec{R}$$

$$f\{\vec{q}+\Delta\vec{q}\} = \vec{q}^T\vec{R}\vec{q} + 2(\Delta\vec{q})^T\vec{R}\vec{q} \qquad \{(\Delta\vec{q})^T\Delta\vec{q}\,\vec{R} \text{ can be neglected} \qquad (3)$$

if we compare the eq.(1), eq.(2) and eq.(3) we get:

$$2(\Delta\vec{q})^T\vec{R}\vec{q} = 0$$

$$\text{i.e} \quad (\Delta\vec{q})^T\vec{R}\vec{q} = 0 \qquad (4)$$

Now we have changed the $\vec{q}$ with a very small amount $\Delta\vec{q}$. As we know that $\vec{q}$ is a unit vector so:

$$\|\vec{q}+\Delta\vec{q}\| = 1$$

Or equivalently $(\vec{q} + \Delta\vec{q})^T(\vec{q}+\Delta\vec{q}) = 1$

$$\vec{q}^T\vec{q} + 2(\Delta\vec{q})^T\vec{q} + \Delta\vec{q}^T\Delta\vec{q} = 1$$

$$1 + 2(\Delta\vec{q})^T\vec{q} = 1 \qquad \{\Delta\vec{q}^T\Delta\vec{q} \text{ can be neglected}\}$$

$$2(\Delta\vec{q})^T\vec{q} = 0$$

$$(\Delta\vec{q})^T\vec{q} = 0 \qquad (5)$$

The above equation implies that $\vec{q}$ and $(\Delta\vec{q})^T$ are orthogonal which permits the change in direction of $\vec{q}$ only. The difference between the eq.(4) and eq.(5) is that there is a an additional term $\vec{R}$ in the eq.(4) and no such term in eq.(5).

We know that $\vec{q}$ and $\Delta\vec{q}$ are dimensionless and let us suppose that we introduce a scalar term λ whose dimension is same as of $\vec{R}$. Now we can re-write both the equations

$$(\Delta\vec{q})^T\vec{R}\vec{q} - \lambda(\Delta\vec{q})^T\vec{q} = 0$$

$$(\Delta\vec{q})^T(\vec{R}\vec{q} - \lambda\vec{q}) = 0$$

But $(\Delta\vec{q})^T$ can never be equal to 0 so:

$$\vec{R}\vec{q} - \lambda\vec{q} = 0$$

$$\vec{R}\vec{q} = \lambda\vec{q} \qquad (6)$$

Now λ is the eigenvalue of $\vec{R}$ matrix and $\vec{R}$ is of m*m dimensions, so there are m such solutions possible.

$\lambda_1, \lambda_2, \lambda_3 \ldots\ldots\ldots\ldots\ldots\lambda_m$ Represents the eigenvalues of $\vec{R}$ and

$\vec{q}_1, \vec{q}_2, \vec{q}_3 \ldots\ldots\ldots\ldots\ldots\vec{q}_m$ Represents the eigenvector of co-relation matrix $\vec{R}$.

As there is m such solution possible so:

$$\vec{R}\vec{q}_{\,j} = \lambda_j \vec{q}_j \qquad \text{where j = 1, 2, 3 ,……. m}$$

And also

$$\lambda_1 > \lambda_2 > \lambda_3 \ldots\ldots\ldots\ldots\ldots\ldots > \lambda_m \tag{7}$$

Now

$$\vec{R}\vec{q} = \vec{q}\vec{\Delta} \tag{8}$$

Where $\vec{\Delta}$ is a diagonal matrix whose diagonal elements are $[\lambda_1 , \lambda_2 , \lambda_3 \ldots..\lambda_m]$. Now $\vec{q}$ is an orthogonal matrix which satisfies

$$\vec{q}_i^{\,T}\vec{q}_j = \begin{matrix} 1 & j = i \\ 0 & j \neq i \end{matrix}$$

From the orthogonal property:

$$\vec{q}^{\,T}\vec{q} = \vec{I}$$

From here we conclude that:

$$\vec{q}^{\,T} = \vec{q}^{\,-1}$$

Now by using eq.(8), we find that:

$$\vec{q}^{\,T}\,\vec{R}\vec{q} = \vec{\Delta} \tag{9}$$

In expanded form this can be written as:

$$\vec{q}_j^{\,T}\vec{R}\vec{q}_{\,k} = \begin{matrix} \lambda_j & k = j \\ 0 & k \neq j \end{matrix}$$

$$\vec{R} = \sum_{i=1}^{m} \vec{q}_i^{\,T}\vec{q}_j\,\lambda_i \tag{10}$$

Also we find that:

$$f\{\vec{q}_i\} = \lambda_j \tag{11}$$

Now there are m possible projection for the unit vector ($\vec{q}$). The j^{th} projected value of $\vec{x}$ onto $\vec{q_j}$ is given as:

$$a_j = \vec{q}_j^{\,T}\vec{x} = \vec{x}_j^{\,T}\vec{q} \quad [\,j = 1,2,3,\ldots\ldots\ldots\ldots m] \tag{12}$$

Where a_j = projection of $\vec{x}$ onto the principle directions represented by q_j and a_j is called as principal components, we will analyze this principal component that's why it is called as principle component analysis. Since j is varying from 1 to m.

$$\vec{a} = \{\, a_1 , a_2 , a_3 \ldots\ldots\ldots\ldots\ldots a_m\}$$

$$\{\vec{x}^T\overrightarrow{q_1}, \vec{x}^T\overrightarrow{q_2}, \ldots\ldots\ldots\ldots \vec{x}^T\overrightarrow{q_m},\}$$

We can also write eq.(12) as:

$$\vec{a} = \vec{q}^T\vec{x} \tag{13}$$

Multiply eq.(13)by $\vec{q}$ on both sides:

$$\vec{q}\,\vec{a} = \vec{x}$$

$$\vec{x} = \vec{q}\,\vec{a}$$

It can also be written as:

$$\vec{x} = \sum_{j=1}^{m} a_j \overrightarrow{q_j} \tag{14}$$

The above equation represents the synthesis and $\overrightarrow{q_j}$ are the basic vector of synthesis. From this we realize that we are not using anything just by knowing co-relation matrix we can reduce the dimensions and co-relation matrix will be known to us just by knowing the eigenvalues and corresponding eigenvectors.

5.8 Method

Step 1: Get some data

In my simple example, I am going to use my own made-up data set. It's only got 2 dimensions, and the reason why I have chosen this is so that I can provide plots of the data to show what the PCA analysis is doing at each step. The data I have used is given :as :

Table 5.5 Randomly selected Data set

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

Step 2: Subtract the mean

For PCA to work properly, you have to subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. So, all the x values have $\vec{x}$ (the mean of the x values of all the data points) subtracted, and all the y values $\vec{y}$ have subtracted from them. This produces a data set whose mean is zero.

Table 5.6 Data Adjusted

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

Step 3: Calculate the covariance matrix

This is done in exactly the same way as was discussed in the initial section. Since the data is 2 dimensional, the covariance matrix will be 2*2.

$$\text{Cov} = \begin{matrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{matrix}$$

So, since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalue for this matrix. These are rather important, as they tell us useful information about our data. The eigenvectors and eigenvalues:

$$\text{Eigenvalues} = \begin{matrix} .0490833989 \\ 1.28402771 \end{matrix}$$

$$\text{Eigenvectors} = \begin{matrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{matrix}$$

It is important to notice that these eigenvectors are both unit eigenvectors ie. Their lengths are both 1. This is very important for PCA.

Step 5: Choosing components and forming a feature vector

Here is where the notion of data compression and reduced dimensionality comes into it. If you look at the eigenvectors and eigenvalues from the previous section, you

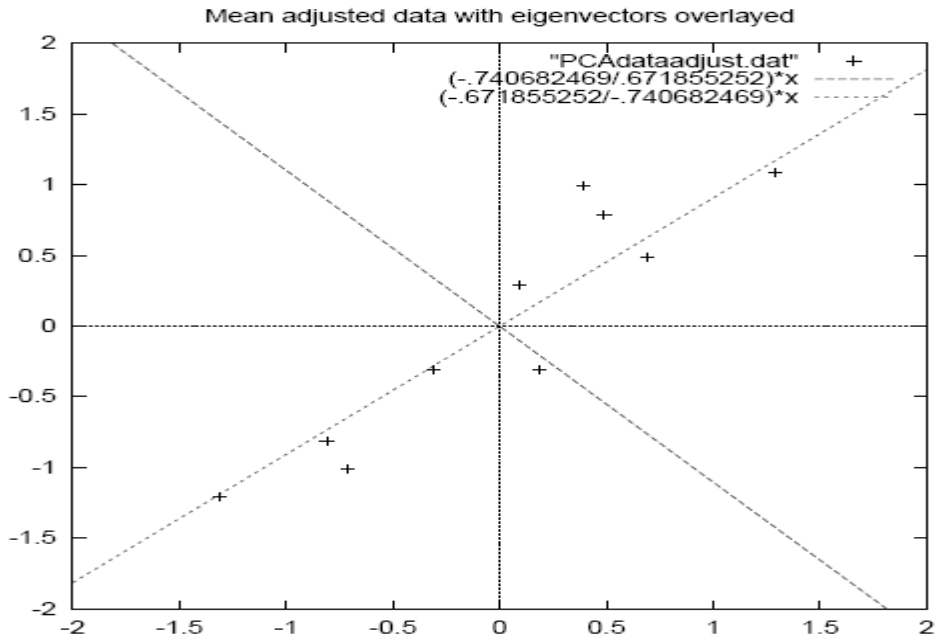


Figure 5.1: Mean adjusted data.

Normalized data with the eigenvectors of the covariance matrix overlaid on top will notice that the eigenvalues are quite different values. In fact, it turns out that the eigenvector with the highest eigenvalue is the principle component of the data set. In our example, the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data. It is the most significant relationship between the data dimensions.

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance. Now, if you like, you can decide to ignore the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don't lose much. If you leave out some components, the final data set will have less dimensions than the original. To be precise, if you originally have n dimensions in your data, and so you calculate n eigenvectors and eigenvalues, and then you choose only the first p eigenvectors, then the

final data set has only p dimensions. What needs to be done now is you need to form a feature vector, which is just a fancy name for a matrix of vectors. This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

Feature vector = (eig1, eig2, eig3…… eign n)

Given our example set of data, and the fact that we have 2 eigenvectors, we have two choices. We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & -.677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Step 5: Deriving the new data set

This the final step in PCA, and is also the easiest. Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.

Final Data = Row Feature Vector * Row Data Adjust

where row feature vector is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and row data adjust is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension. The equations from here on are easier if we take the transpose of the feature vector and the data first, rather that having a little T symbol above their names from now on final data is the final data set, with data items in columns, and dimensions along rows. a little T symbol above their names from now on. Final data is the final data set, with data items in columns, and dimensions along rows. It will give us the original data solely in terms of the vectors we chose. Our original data set had two axes, x and y, so our data was in terms of them. It is possible to express data in terms of any two axes that you like. If these axes are perpendicular, then the expression is the most efficient. This was why it was important that eigenvectors are always perpendicular to each

other. We have changed our data from being in terms of the axes x and y , and now they are in terms of our 2 eigenvectors. In the case of when the new data set has reduced dimensionality, ie. We have left some of the eigenvectors out, the new data is only in terms of the vectors that we decided to keep. To show this on our data, I have done the final transformation with each of the possible feature vectors. I have taken the transpose of the result in each case to bring the data back to the nice table-like format. I have also plotted the final points to show how they relate to the components. In the case of keeping both eigenvectors for the transformation, we get the data and the plot found in Figure 1.

Table 5.7 Transformed Data

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

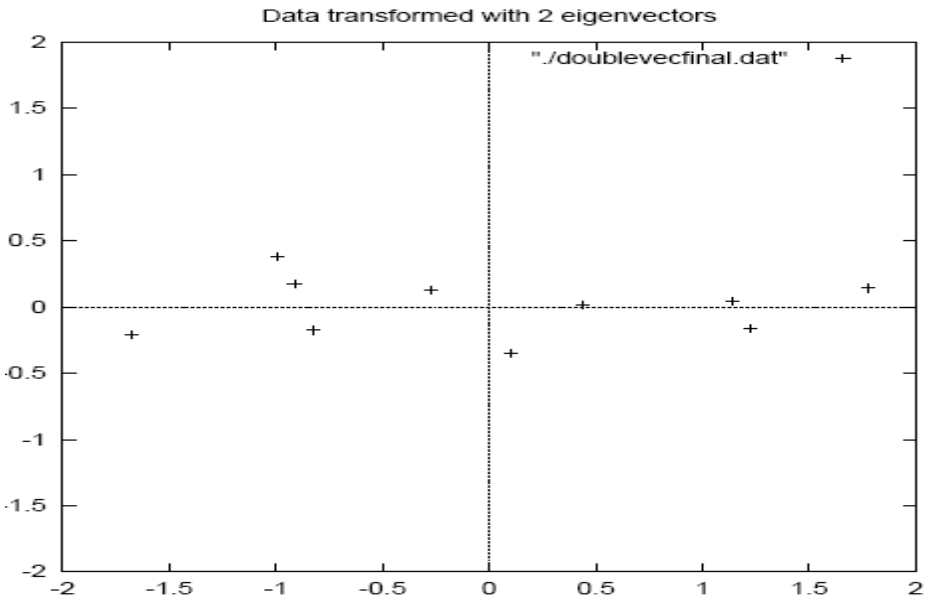


Figure 5.2: Transformed Data

This plot is basically the original data, rotated so that the eigenvectors are the axes. This is understandable since we have lost no information in this decomposition. The other transformation we can make is by taking only the eigenvector with the largest eigenvalue. The table of data resulting is as given as:

Table 5.8 Transformed Data (Single eigenvector)

-.827970186
1.77758033
-.992197499
-.274210416
-1.67580142
-.912949103
.099109437
1.14457216
.438046137
1.22382056

As expected, it only has a single dimension. If you compare this data set with the one resulting from using both eigenvectors, you will notice that this data set is exactly the first column of the other. So, if you were to plot this data, it would be 1 dimensional, and would be points on a line in exactly the x positions of the points in the plot in Figure 1. We have effectively thrown away the whole other axis, which is the other eigenvector. So what have we done here? Basically we have transformed our data so that is expressed in terms of the patterns between them, where the patterns are the lines that most closely describe the relationships between the data. This is helpful because we have now classified our data point as a combination of the contributions from each of those from each of those lines. Initially we had the simple x and y axes. This is fine, but the x and y values for each data point do not really tell us how points are related to rest of data. Now, the values of the data points tell us exactly where (i.e. above/below) the trend lines the data point sits. In the case of the transformation using *both* eigenvectors, we have simply altered the data so that it is in terms of those eigenvectors instead of the usual axes. But the single-eigenvector decomposition has removed the contribution due to the smaller eigenvector and left us with data that is only in terms of the other.

Chapter 6

Fault Detection in Waste Water Treatment Plant

6.1 Waste water treatment plant

Wastewater treatment can encompass a number of steps, which filter, clarify and clean wastewater from start to finish. Currently, the CRD region employs a variety of wastewater treatments, some of which filter and some of which provide secondary treatment [65].

6.1.1 CRD

The CRD is exploring treatment technologies as a way of recovering wastewater components for reuse and for energy purposes. Some of the most advanced wastewater technologies emerging today occur in tertiary treatment.

6.1.2 Preliminary Treatment

Preliminary treatment screens out coarse solids (rocks, rags, plastics, etc.) and grit (sand and gravel) which are normally sent to landfill. Wastewater is screened down to 6mm sized particles at the CRD's Clover Point and Macaulay Point Facilities, where preliminary treatment is currently in effect. The screened wastewater then continues unimpeded to outfalls, where it is discharged through two deep ocean outfalls into the marine waters of Juan de Fuca Strait.

6.1.3 Primary Treatment

Primary treatment screens wastewater, and performs some rudimentary treatment to remove crude solids and skim off grease, oil and fat. Wastewater sits in settling tanks, which are designed to hold the wastewater for several hours. During that time, most of the heavy solids fall to the bottom of the tank, where they become a thick slurry known as primary sludge. The material that floats is also skimmed from the surface of the tanks. Both the primary sludge and skimmed material are typically pumped or trucked to a solids treatment processing plant.

6.1.4 Secondary Treatment

Secondary (or biological) treatment removes dissolved oxygen-demanding organic substances by using bacteria to convert degradable organic matter into bacterial cells. The wastewater is then clarified by separating treated liquid from grown bacterial cells using gravity. Bacteria and sludge is then either processed onsite or sent to a separate solids treatment facility.

6.1.5 Tertiary Treatment

Tertiary treatment further treats effluent to remove nitrogen, phosphorus, fine suspended particles and microbes, and to kill or disable disease-causing organisms and viruses. It is possible to treat effluent in this phase, resulting in a non-potable reclaimed water source, which can be reused in a variety of ways.

6.2 Objective:

This report considers a waste water treatment plant and ascertains the condition of the plant (Healthy or Faulty) by using multivariate statistical techniques and artificial neural network [66-68]. The dataset for waste water treatment plant is taken from UCI machine learning respiratory; and is available online:

http://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant (last accessed 4th March 2012).

This is a multivariate dataset created by Manel Poah, Unitat d'Enginyeria Quimica Universitat Autonoma de Barcelona. Bellaterra. Barcelona; Spain and donated by Javier Bejar and Ulises Cortes, Dept. Llenguatges i Sistemes Informatics; Universitat olitecnica de Catalunya. Barcelona; Spain. The dataset shows the recorded reading of the variables for 527 days. This is an activated sludge process located and treats a daily flow of 35000 m^3 of domestic and industrial wastewater. This plant consists of 3 parts:

1. Pre treatment
2. Primary Treatment
3. Secondary treatment

Different attributes, minimum maximum and mean values of data set are mentioned in table 1 and table 2 and table 3 illustrate the number of missing values in our data.

Table 6.1: Different attributes of waste water treatment plant

Attributes	Description
Q-E	Input flow to plant
ZN-E	Input zinc to plant
PH-E	Input ph to plant
DBO-E	Input biological Demand of O_2 to Plant
DQO-E	Input chemical demand of oxygen to plant
SS-E	Input suspended solids to plant
SSV-E	Input volatile suspended solids to plant
SED-E	Input sediments to plant
COND-E	Input conductivity to plant
PH-P	Input pH to primary settler
DBO-P	Input Biological demand of oxygen to primary settler
SS-P	Input suspended solids to primary settler
SSV-P	Input volatile suspended solids to primary settler
SED-P	Input sediments to primary settler
COND-P	Input conductivity to primary settler
PH-D	Input pH to secondary settler
DBO-D	Input Biological demand of oxygen to secondary settler
DQO-D	Input chemical demand of oxygen to secondary settler
SS-D	Input suspended solids to secondary settler
SSV-D	Input volatile suspended solids to secondary settler
SED-D	Input sediments to secondary settler
COND-D	Input conductivity to secondary settler
PH-S	Output pH
DBO-S	Output Biological demand of oxygen
DQO-S	Output chemical demand of oxygen
SS-S	Output suspended solids
SSV-S	Output volatile suspended solids
SED-S	Output sediments

COND-S	Output conductivity
RD-DBO-P	Performance input Biological demand of oxygen in primary settler
RD-SS-P	Performance input suspended solids to primary settler
RD-SED-P	Performance input sediments to primary settler
RD-DBO-S	Performance input Biological demand of oxygen to secondary settler
RD-DQO-S	Performance input chemical demand of oxygen to secondary settler
RD-DBO-G	Global performance input Biological demand of O_2
RD-DQO-G	Global performance input chemical demand of O_2
RD-SS-G	Global performance input suspended solids
RD-SED-G	Global performance input sediments

Table 6.2: Maximum, Minimum and Mean values of variables:

Variables	Max	Min	Mean
Q-E	10000	60081	37226.56
ZN-E	0.1	33.5	2.36
PH-E	6.9	8.7	7.81
DBO-E	31	438	188.71
DQO-E	81	941	406.89
SS-E	98	2008	227.44
SSV-E	13.2	85	61.39
SED-E	0.4	36	4.59
COND-E	651	3230	1478.62
PH-P	7.3	8.5	7.83
DBO-P	32	517	206.2
SS-P	104	1692	253.95
SSV-P	7.1	93.5	60.37
SED-P	1	46	5.03
COND-P	646	3170	1496.03
PH-D	7.1	8.4	7.81
DBO-D	26	285	122.34

DQO-D	80	511	274.04
SS-D	49	244	94.22
SSV-D	20.2	100	72.96
SED-D	0	3.5	0.41
COND-D	85	3690	1490.56
PH-S	7	9.7	7.7
DBO-S	3	320	19.98
DQO-S	9	350	87.29
SS-S	6	238	22.23
SSV-S	29.2	100	80.15
SED-S	0	3.5	0.03
COND-S	683	3950	1494.81
RD-DBO-P	0.6	79.1	39.08
RD-SS-P	5.3	96.1	58.51
RD-SED-P	7.7	100	90.55
RD-DBO-S	8.2	94.7	83.44
RD-DQO-S	1.4	96.8	67.67
RD-DBO-G	19.6	97	89.01
RD-DQO-G	19.2	98.1	77.85
RD-SS-G	10.3	99.4	88.96
RD-SED-G	36.4	100	99.08

The dataset provided in UCI machine learning respiratory has some missing data. The number of missing data is illustrated in table 3.

Table 6.3: Number of missing variables in each parameter of the dataset

Variables	Number of Missing Variables
Q-E	18
ZN-E	3
PH-E	0

DBO-E	23
DQO-E	6
SS-E	1
SSV-E	11
SED-E	25
COND-E	0
PH-P	0
DBO-P	40
SS-P	0
SSV-P	11
SED-P	24
COND-P	0
PH-D	0
DBO-D	28
DQO-D	9
SS-D	2
SSV-D	13
SED-D	25
COND-D	0
PH-S	1
DBO-S	23
DQO-S	18
SS-S	5
SSV-S	17
SED-S	28
COND-S	1
RD-DBO-P	62
RD-SS-P	4
RD-SED-P	27
RD-DBO-S	40

RD-DQO-S	26
RD-DBO-G	36
RD-DQO-G	25
RD-SS-G	8
RD-SED-G	31

The missing values are interpolated with the help of bicubic interpolation technique. As we have seen that a large amount of data is available to us so it is very difficult for us to deal with such kind of high dimensioned data. To avoid this problem we use a statistical tool Principle Component already described to reduce the dimension of data. On applying PCA we get 23 principle components. But only two principle components are sufficient for us because they are representing more that 99% of data. The systematic flow chart for the fault detection in waste water plant is as shown:

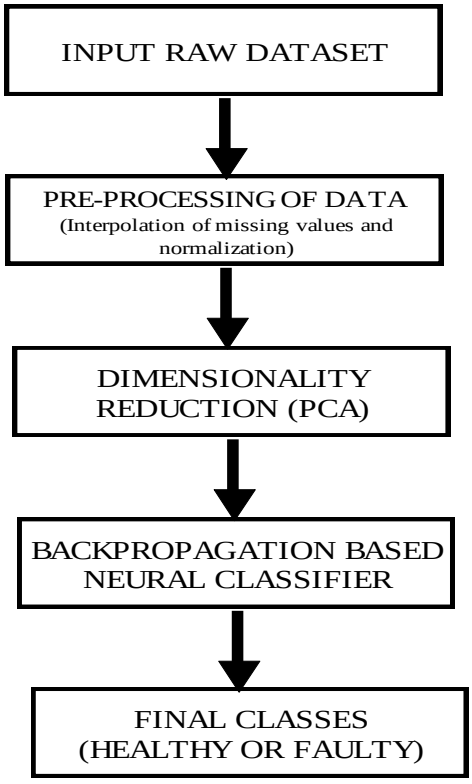


Figure 6.1 Flow Chart of Error Detection in Waste Water Treatment Plant

Figure 6.1 represents the flow chart to classify the dataset in to two categories, healthy or faulty. The first step is pre processing of data, then dimensionality reduction of data and then neural network based classification to classify the dataset. After dimensionality reduction, we apply two types of algorithms for fault detection:

1.) Supervised algorithm (Back-propagation).
2.) Unsupervised algorithm (Clustering).

For supervised algorithm neural network based classification technique is used to differentiate between healthy and faulty data. In neural network based classification technique, back-propagation algorithm is used. And we use clustering technique for unsupervised algorithm to differentiate the healthy and faulty data.

After pre-processing of data, PCA is implemented in the dataset to reduce the dimension. PCA gives two variables, PC1 and PC2 which imitates 98.6% and 1.4% of the dataset.\ Classification of variable using different principle components is as shown below:

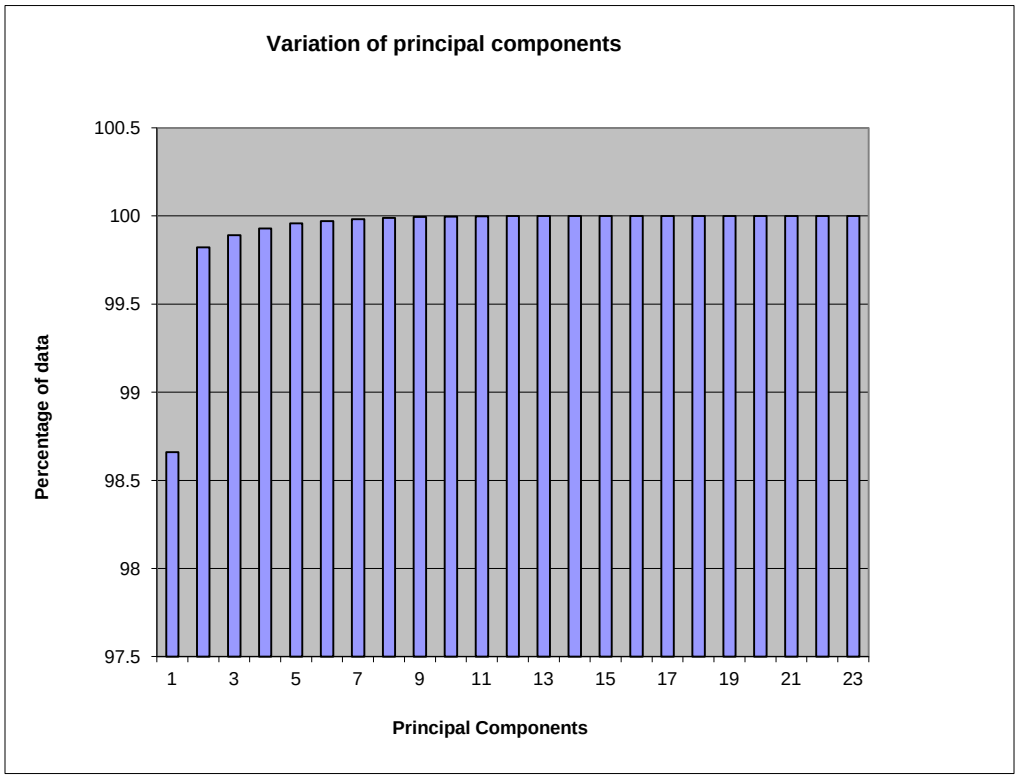


Figure 6.2 Classification of variables using PCA

Figure 6.2 classifies the variable using PCA. As we have already mentioned that we are dealing with 23 variables so with respect to that 23 variables we get 23 Principle Components for 100% representation of data. But only 2 principle components are enough for us because they are representing more that 99% of data which is clear from the figure 6.2. The scatter plot of PC1 and PC2 as shown in below:

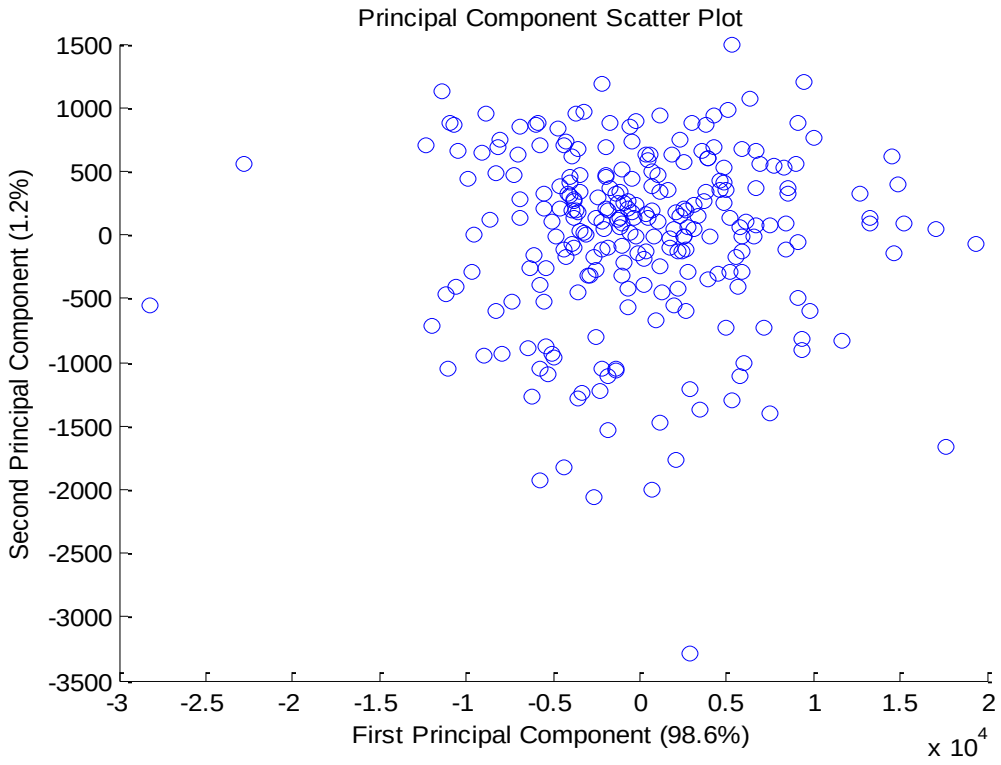


Figure 6.3 Scatter plot for PC1 and PC2

In PC1, 98.6% of variance of the dataset is represented where as in PC2, 1.4% of the variance of dataset is represented. By the help of PCA, very large dataset is reduced to only two variables which replicate 100% variance of the dataset. The two principle components are not distinguishable in the above plot. These can be easily distinguished in the colored plot as shown in the next figure:

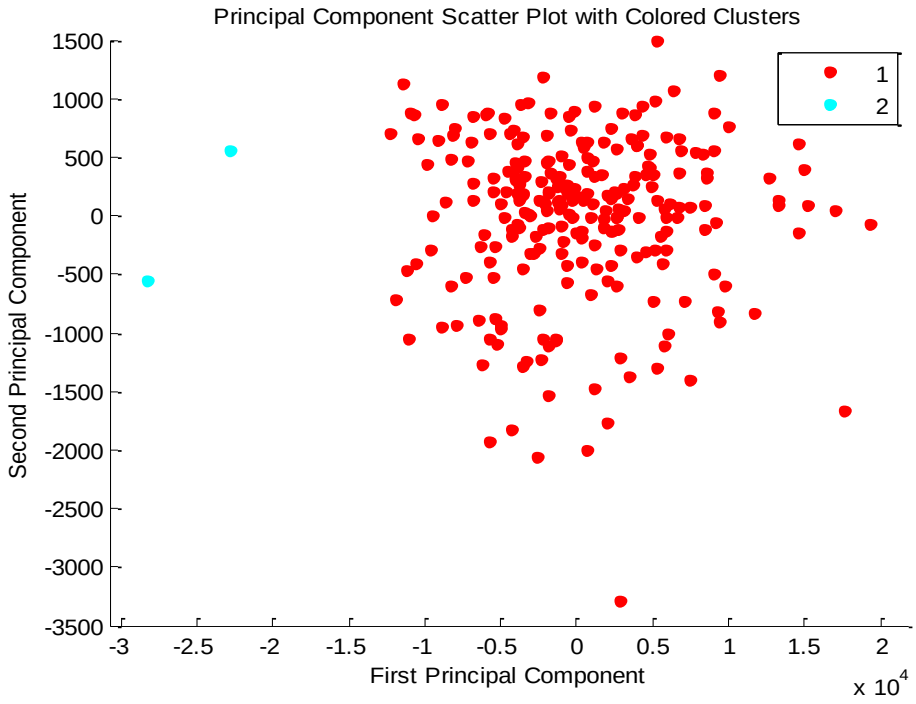


Figure 6.4 Different principal components and the variation of principal components.

After the implementation of Principle Component Analysis on the available data set we apply supervised learning algorithm for the implementation of supervised learning algorithm we use back-propagation algorithm. The results obtained from back-propagation are as shown below:

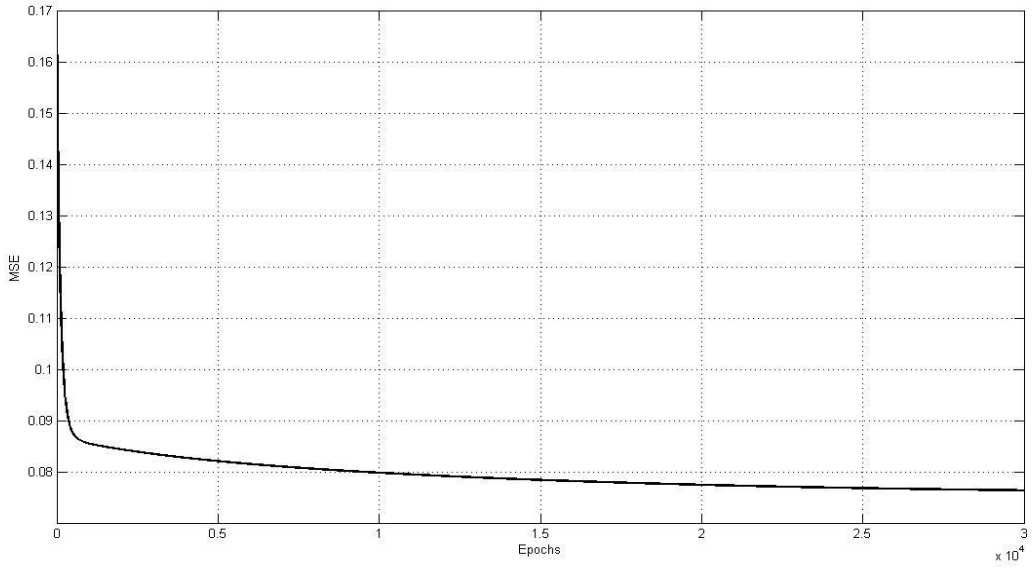


Figure 6.4 Epoch curve of Backpropagation

The above result signifies that the MSE(mean square error) is decreasing with respect to every iteration. Accuracies that we obtained is:

Mean training accuracy is 99.207%.

Mean test accuracy is 93.1661%.

After the implementation of supervised algorithm we switch our self towards unsupervised algorithm. For unsupervised algorithm implementation we use clustering technique that we have already mentioned. For this we use two types of clustering algorithms:

1) K-means Clustering.
2) Fuzzy C means clustering.

Cluster of healthy and faulty data after the implementation of K-means clustering is as shown below:

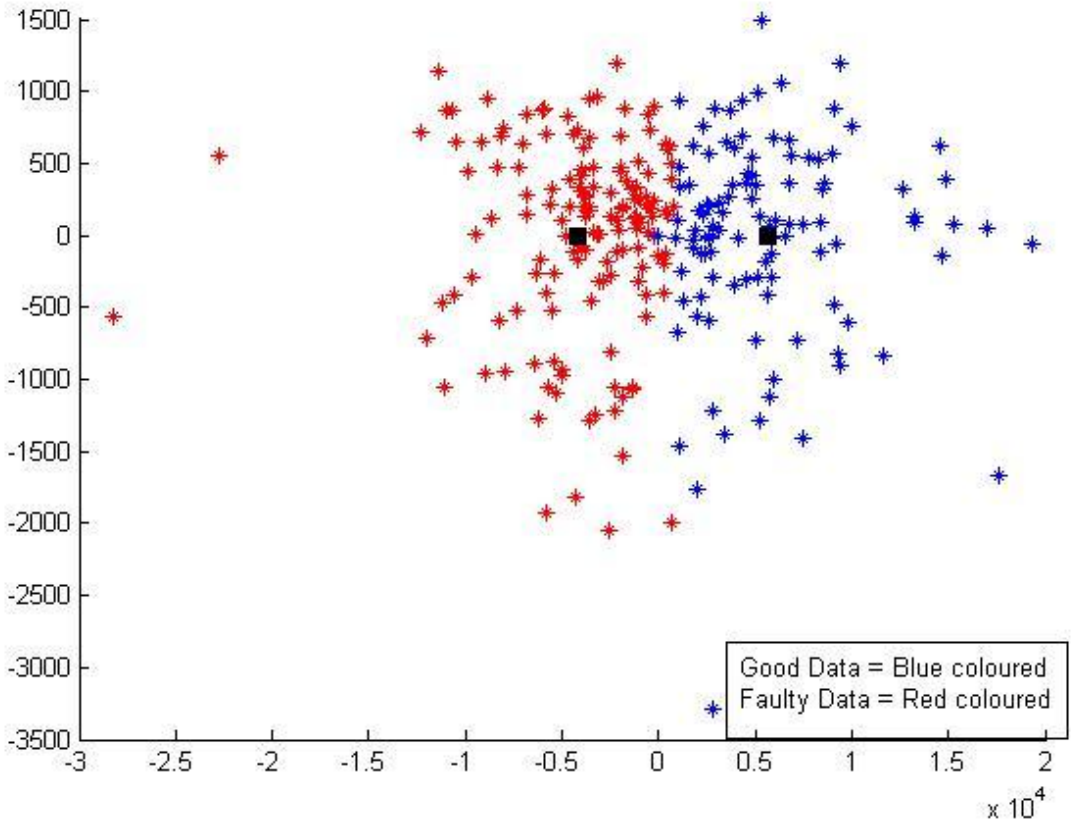


Figure 6.5 Clusters of Good and Faulty Data with K-means

After K-means clustering implementation we use Fuzzy C-means as second technique for clustering of good and faulty data. Result that we obtained after the implementation of Fuzzy C-means is as shown in the Figure 6.6:

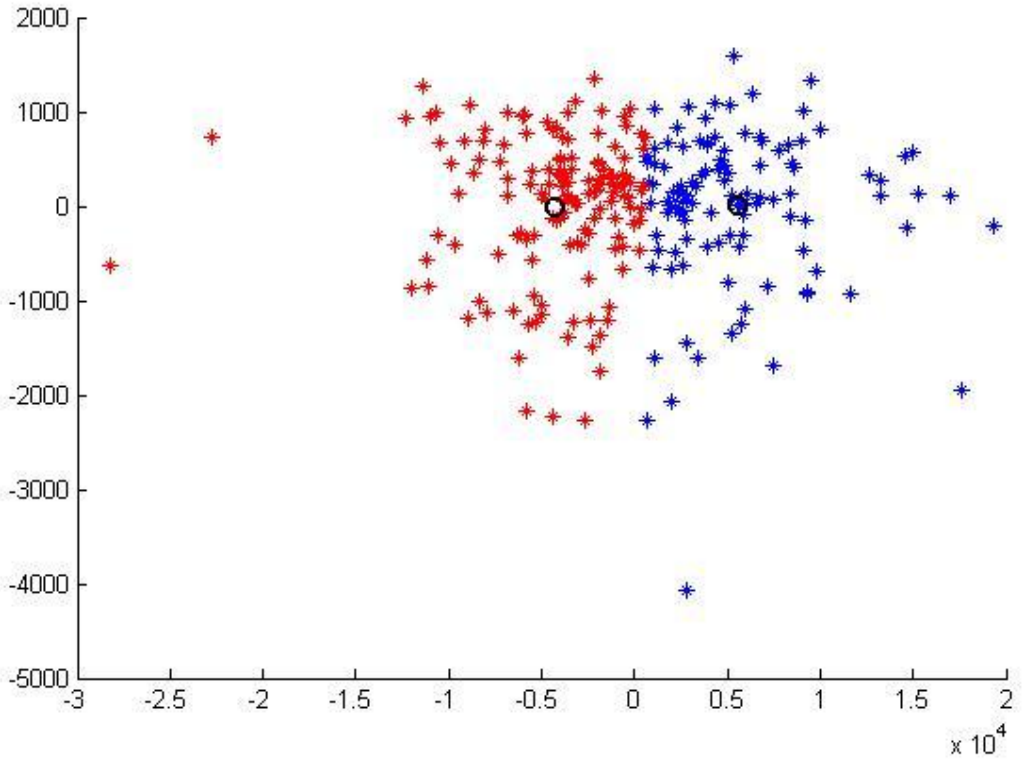


Figure 6.6 Clusters of Good and Faulty data with Fuzzy C-means

In the figure 6.6 blue colored cluster represents the good data and red colored cluster represents the cluster of bad data. From figure 6.5 and figure 6.6 we have easily concluded that:

K-means and Fuzzy C-means are very similar in approaches. But in Fuzzy-C Means clustering, each point has a weighting associated with a particular cluster, so a point doesn't sit "in a cluster" as much as has a weak or strong association to the cluster, which is determined by the inverse distance to the center of the cluster. Fuzzy-C means will tend to run slower than K means, since it's actually doing more work. Each point is evaluated with each cluster, and more operations are involved in each evaluation. K-Means just needs to do a distance calculation, whereas fuzzy c means needs to do a full inverse-distance weighting.

Chapter 7

Conclusion and Future Scope

Classification of data is a challenging task. This thesis takes a case study of waste water treatment plant and classifies the data in to healthy data or faulty data. Different supervised and unsupervised classification techniques are used for classification. In unsupervised classification, different clustering algorithm is used and the performances of all the algorithms are evaluated. In supervised classification, the data is reduced using PCA then trained using neural network using error-backpropagation algorithm and after that it is tested to find out the accuracy.

In future scope, different statistical classification techniques like (naive-bayes classification technique) and different hybrid unsupervised segmentation technique (genetic C Means, Fuzzy C Means+PSO) can be implemented and a comparative study can be performed.

REFERENCES

[1] Shapiro, Stuart C, "Handbook of Artificial Intelligence", John Wiley & Sons Ltd, 1992.

[2] Naresh K. Sinha and Madan M. Gupta, "Soft Computing and Intelligent Systems Theory and Application," Academic Press, 2000.

[3] L.X.Wang, "Fuzzy Systems are Universal Approximations," Proc. 7th Int. Sci. & Tech. Conf.- Process Control, vol. 2, pp. 1-7, 2006.

[4] Christer Carlsson, "Introduction to the Minitrack on Fuzzy Logic and Soft Computing in Service and Management Support," Proc. 45th Hawaii International Conf, pp. 3849-3853, 2009.

[5] L.Zadeh, "Handbook of Fuzzy Logic and Soft Computing", Muroran & Sons Ltd, 1993.

[6] M.Mukaidono, "Introduction to Fuzzy Set Theory," Magazine Trigger, 1990.

[7] Fakhreddinekarray and Clarence de Silva, "Handbook of Soft Computing and Intelligent Systems Design", Addison Wesley & Company Ltd, 2004.

[8] Yasuhiko Dote, " Textbook of Introduction To Fuzzy Logic", S. Chand & Company Ltd, 1995.

[9] P. Elmer-DeWitt, "A textbook of Time for some fuzzy thinking", Khanna Publishers, 1989.

[10] Joseph Bih, "Article on Introduction to fuzzy logic controller", Soft computing society 1991.

[11] Lee, C, "Fuzzy Logic in Control Systems :F'uzzy Logic Controller- Part I and Part 11", Proc. IEEE Conf. Decis. Cont, pp. 404-435, 1990.

[12] Joachim Stender, "Introduction to Genetic Algorithms and its adaptation in natural and artificial system", Brainware Ltd, 1999.

[13] XiaoFeng Li, Li Li Yang, "Study of classification of Genetic algorithm," Proc. Of 3rd International Conference, ICACTE, , pp. 3182-3187, 2010.

[14] Beale, R. and Jackson, "Handbook on Neural Computing", Adam Hilger Ltd, 1990.

[15] Diederich, J, "Article on Artificial neural networks: concept learning", IEEE Computer Society press, 1990.

[16] Vemuri, "Article on Artificial neural networks: theoretical concepts", IEEE Computer Society press, 1991.

[17] Uhrig, R.E., "Introduction to artificial neural networks," Proc. Of IEEE IECON 21st International Conference, pp. 1390-1395, 1995.

[18] J.C. Eccles, "Handbook on The understanding of the Brain", McGraw-Hill, 1997.

[19] B.M. Wilamowski, "Neural network architectures and learning algorithms", Proc. 7th World Congress Intelligent Control and Automation, pp. 3380-3383, 2008.

[20] B. M. Wilamowski, ''Neural networks for nonlinear application," Proc. Of 11th Int. Conf. Intelligent Engineering Systems, Budapest pp. 13–19, 2007.

[21] B. Vries and J. C. Principe, "The gamma model-Neural Networks", Proc. American Control Conference, pp. 3182-3187, 2006.

[22] G.X. Ritter and G. Urcid, "Lattice algebra approach to single neuron computation", IEEE Tranr. Neural Network, pp. 1791-1796, 2008, 2004.

[23] M. Gori. F. Scarrelli, "Multilayer Perceptrons Adequate for Pattern Recognition and Verification" IEEE Trans. on Pattern Analysis and Machine Inrelligmce, vol. 20(1 I), pp. 3123-3130, 2000.

[24] C.L. Giles and C.W. Omlin, "Inserting rules into recurrent neural networks," Proc. of IEEE Workshop, pp. 13-22, 1992.

[25] C.L Giles, "Remembering the past: the role of embedded memory in different recurrent neural network architectures," Proc. of the IEEE Workshop, pp. 171-180, 1997.

[26] D. P. Mandie and J. A. Chambers, “Handbook on Recurrent Neural Networks for Prediction”, Chichester &Wiley Ltd, 2001.

[27] Yanjun Pang, “A Special Supervised Learning Algorithm and Its Applications,” Proc. of 9th Int. Conf. on Hybrid Intelligent Systems, pp. 4767-4771, 2009.

[28] YuanyuanGuo, “An Extensive Empirical Study on Semi-supervised Learning,” Proc. Of IEEE Int. Conf. on Data Mining, pp. 792 – 812, 2010.

[29] X. Zhu,“Semi-supervised and supervised learning literature survey,” Journal of Artificial Intelligence, 2008.

[30] Yanjun Pang, Jiqiang Chen, Nianpeng Wang, “A Special Supervised Learning Algorithm and Its Applications,” Proc. of 9th Int. Conf. on Hybrid Intelligent Systems, pp. 1325-1331, 2009.

[31] M.I El Adawy, “A soft backpropagation algorithm for training neural networks,” Proc. of 19th National Conf. on Radio Science, pp. 696-701, 2002.

[32] Q. Song, J. Xiao, and Y.C. Soh,"Robust Back-propagation Training Algorithm for Multi-layered Neural Tracking Controller", IEEE Trans. On Neural Networks, Vol.10, No.5, pp.1133-1141, 1999.

[33] P. J. Werbos, "Generalisation of backpropagation with application to a recurrent gas market model", Neural Networks, Vol. 1, pp. 339356, 1998.

[34] Z Maalej, “Reduced complexity for back-propagation method algorithm,” Proc. of IEEE Photonics Conference, 2011.

[35] G.Vachkov, “Unsupervised learning algorithm for comparison an analysis of Multisensory data,” Proc. of Int. Conf. Mechatronics and Automation, 2008

[36] Jiamthapthaksin, “An architecture and algorithms for multi types clustering,” proc. of Symposium on Computational Intelligence and Data Mining, 2009.

[37] R. T. Ng, J. Sander, and M. C. Sleumer, “Cluster analysis of SAGE data for cancer profiling,” Proceedings American Control Conference, San Diego, pp.3843-3847, 1999.

[38] Raza Ali, “Different types of clustering for data mining”, article available at members.tripod.com/asim_saeed/paper.htm

[39] D. H. Hutchens and V. Basili, "System Structure analysis: clustering with databindings", IEEE Transaction, vol. SE-ll(8), 1985.

[40] R.Ibba and D.Natale, “Structure-based Clustering of Components for Software Reuse and Assessment criteria,” Proc. of IEEE Conf. on Software Maintenance, 2000.

[41] Shi Na, “Research on k-means Clustering Algorithm: An Improved k-meansClustering Algorithm,” Proc. of 3rd Int. Symposium on Intelligent Information Technology and Security Informatics, 2010.

[42] Yuan F, Meng Z. H, Zhang H. X and Dong C. R, “A New Algorithm to Get the Initial Centroids,” Proc. of 3rd Int. Conf. on Machine Learning and Cybernetics, pp. 26–29, 2004.

[43] Sun Jigui, Liu Jie, Zhao Liany , “K-Means Clustering algorithms Research,” Journal of Software Vol 19,No 1, pp.48-61, 2008.

[44] Sun Shibao, Qin Keyun, “Research on Modified k-means Data Cluster Algorithm,” Journal of Zhejiang University Science A, Vol.10, pp.1626-1633, 2007.

[45] Z Huang, “Extensions to the k-means algorithm for clustering large data sets with categorical values”, Data Mining and Knowledge Discovery, Vol.2, pp:283–304, 1998

[46] Miyamoto, “Recent studies on algorithms for fuzzy clustering,” Proc. of IEEE Int. Conf. on Granular Computing, 2008.

[47] Yaqiong, L., L. K. Man,"A literature review and classification of Fuzzy clustering theory”, Engineering Applications of Artificial Intelligence, 2000.

[48] S.H Chen, C. C. Wang, “Fuzzy distance of trapezoidal fuzzy numbers,” Proc. of the 9th Joint Conference on Information Sciences, 2006.

[49] Weina Wang; Yunjie Zhang; Yi Li; Xiaona Zhang, "The Global Fuzzy C-Means Clustering Algorithm," Proc. of 6th World Conf. on Intelligent Control and Automation, 2006.

[50] T. Kwok, R Smith, S. Lozano, and D. Taniar," Handbook on Parallel fuzzy c", vol. 2400 of LNCS, pp. 365-374, 2002.

[51] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler,"Article on Fuzzy cluster analysis," Wiley Press, 1999.

[52] Chen Yu, Zhang Jian ,Yi Bo, "A Novel on Principal Component Analysis NeuralNetwork Algorithm," Proc. of Asia-Pacific Conference on Information Processing, Vol. 2 pp. 2312-2318, 2009.

[53] R Dunia, S J Qin, T F Edgar and T J McAvoy, "Identification of faulty sensors using principal component analysis," AICHE Journal, vol. 42, no. 10, pp. 2797-2812, 1996.

[54] Y M Sebzalli and X Z Wang, "Knowledge Discovery from Process Operational Data using PCA and Fuzzy Clustering", Engineering Applications of Artificial Intelligence, vol. 14, pp. 607-616, 2001.

[55] N Bendwell, "Monitoring of a waste water treatment plant with a multi variate model", Canada: Pulp and Paper, vol. 103, no. 7, pp.43-46, 2002.

[56] D Wang and J.A. Romagnoli, "Robust multi scale principal components analysis with applications to process monitoring," Journal of process control, vol. 15, no. 8,pp. 869-882, 2005.

[57] Z.Q. Ge and Z.H. Song, "Process monitoring based on independent component Analysis principal component analysis and similarity factors", Industrial & Engineering Chemistry Research, vol. 46, no. 7, pp. 2054 – 2063, 2007.

[59] David L. Hall,"An Introduction to Multisensor Data Fusion", IEEE, vol. 85 no. 1, 2004.

[60] D. Hall, "Mathematical Techniques in Multisensor Data Fusion", Boston, 1990.

[61] R. J. Linn, and J. Llinas, "A survey of data fusion systems," Proc. of SPIE Conf. on Data Structure and Target Classification, vol. 1470, 1991.

[62] J. Llinas and E.Waltz, "Multisensor Data Fusion", Boston:1990.

[63] L A. Klein, "Sensor and Data Fusion Concepts and Applications" SPIE Engineering Press, Tutorial Texts, vol. 14,1993.

[64] J. Llinasl, "A challenge for the data fusion community II: Infrastructure imperatives," Proc. of 7th National Symposium on Sensor Fusion, vol.9 ,pp. 99-108, 1994.

[65] Article on "Waste Water Treatment Plant" available at

http://www.gocolumbiamo.com/PublicWorks/Sewer/wwtppg_4.php

[66] V. Venkatasubramaniam, R Rengaswamy, K Yin and S N Kavuri, "A review of process fault detection and diagnosis Part-I: Qulitative model based methods", Computers and Chemical Engineering, vol. 27, no. 3, pp. 293 – 311, 2003.

[67] V. Venkatasubramaniam, R Rengaswamy and S N Kavuri, "A review of process fault detection and diagnosis Part-II: Quantitative model and search strategies", Computers and Chemical Engineering, vol. 27, no. 3, pp. 313 – 326, 2003.

[68] V. Venkatasubramaniam, R Rengaswamy, S N Kavuri and K Yin, "A review of process fault detection and diagnosis Part- III: Process history based methods", Computers and Chemical Engineering, vol. 27, no. 3, pp. 327 – 346, 2003.

[69] Vasil Simeonov, "Lake Water Monitoring Data Assessment by Multivariate Statistics," Journal of Water Resource and Protection, vol. 2, pp. 353-361, 2010.