Pattern Analysis Techniques for Identification of Individual Gases Using Response of a Poorly Selective Solid-state Sensor Array

Thesis submitted towards the partial fulfilment of requirement

for the award of degree of

Master of Engineering

In

Electronics and Communication Engineering

Submitted by:

Akash Agarwal

Roll No: 801161002

Under the guidance of:

Dr. Ravi Kumar

Assistant Professor, ECED

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

JUNE 2013

DECLARATION

I hereby declare that the thesis entitled **"Pattern Analysis Techniques for Identification of Individual Gases Using Response of a Poorly Selective Solid-state Sensor Array"** is an authentic record of my work carried out as requirement for the award of degree of Master of Engineering (Electronics and Communication Engineering) at Thapar University, Patiala, under the supervision of Dr. Ravi Kumar, Assistant Professor, Department of Electronics and Communication Engineering, Thapar University, Patiala.

I have not submitted the matter presented in the thesis for the award of any other degree to any other university.

Date: 25/06/13

Akash Agarwal

801161002

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

Date: 25/06/13

Dr. Ravi Kumar

Assistant Professor

Dr. Rajesh Khanna

Professor & Head

ECED, Thapar Uuniversity

Patiala

Dr. S.K. Mohapatra

Dean, Academic Affairs

Thapar University

Patiala

ACKNOWLEDGEMENT

At first, my sincere thanks to the almighty for his blessing throughout this journey to complete this work of mine. I am thankful to my parents for their great support throughout my life. It is because of them that I am here today.

I am thankful to my guide Dr. Ravi Kumar, Assistant Professor, Department of Electronics & Communication Engineering, Thapar University**,** for his excellent guidance, valuable discussions, encouragement and constructive criticism.

I am also thankful to Dr. Rajesh Khanna, Professor & Head and Dr. Kulbir Singh, Associate Professor, Electronics & Communication Engineering Department, Thapar University, Patiala for their valuable advice

I would like to thank all my colleagues for their support and encouragement in carrying out my research smoothly.

INDEX

List of Figures

List of Tables

Chapter 1 - Introduction and Literature Review

1.1 Introduction

Perception is one of the most complex process found in nature. It is a continuously evolving dynamic process involving organization, identification and interpretation of sensory information which gives form to one's personality and gives him an interpretation of environment. It is a process which shapes an individual's behaviour, attitude, analytical skills and gives him an ability to visualize and process thoughts.

Out of these sensory processes, smell is the least understood phenomenon. Smell triggers human memory faster than any other sensory process. It is unarguably the most complex and unexplored of all the sensory mechanisms with unmatched potential and abilities.

Our zest to characterize the odors around us persuaded us to manually smell and check them. As a result we used our collective expertise to characterize and sometimes even quantify these odours. But the inherent glitches in the above method like individual variability, adaptation, mental state, fatigue, subjectivity, infections and exposure to hazardous compounds etc. persuaded researchers to somehow design a system that could replace these traditional methods so as to achieve our target of odor characterization.

Gas chromatography and mass spectroscopy seemed to be the solutions at that time but the inherent deficiencies in the method like bulkiness, slow rate of process and cost of the apparatus didn't allow it to face the test of time. With the help of modern day technology we were able to replace these purely chemical science based methods with material science and microelectronics. As a result sensor arrays fabricated with technology known at that time came into being. These devices popularly known as electronic nose systems emerged as a viable and cost effective alternative for gas chromatography and mass spectroscopy based odor classifiers. Being appreciably sensitive to a large number of odor/gases, e-noses are a versatile genre of solid-state sensing devices. Although these systems cannot match the human olfactory system in any way, they are still considered to be the artificial counterpart of the human olfactory system.

An electronic nose is one of the first practical application of multiple sensors aligned in the form of an array so as to make the entire assembly work as a complete system and give an

integrated response. In the array individual sensors respond broadly to a range of gases rather to specific ones.

In an electronic nose system adsorption is the key process behind the whole working. The odorant molecules after striking the surface of the sensing array produce change in the properties of the sensors. This conversion may be seen as a conversion from chemical energy to electrical one. The electrical signals produced are fed to a circuit whose job is to amplify, pre-process and digitize the output. However the job is not yet complete. To overcome the partial sensitivities and many other inherent factors involved, the above system needs to be followed by an efficient signal processing and pattern classification system in order to have any meaningful results from the system.

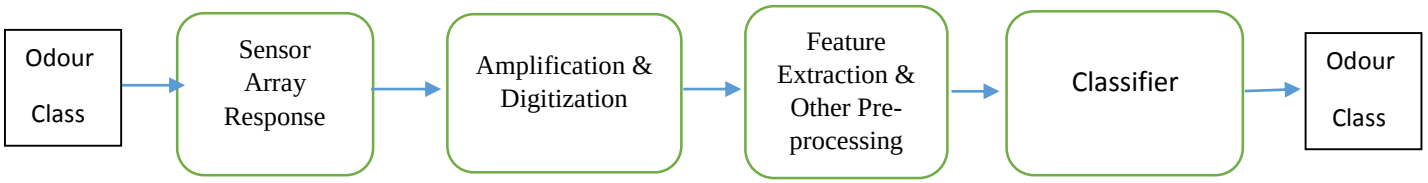


Fig. 1.1 Block diagram of an artificial olfactory system

Initially single sensor and single gas approach was popular. In order to increase the working domain of these devices the integrated approach came into existence. In this approach the sensors are integrated with same substrate but with different dopants rather than just connecting different sensors in parallel. Thus the concept of sensor array came into existence.

When any odorant molecule strikes the surface of the sensor it chemically binds with the sensor atom and creates a film of adsorbate on the surface of the sensor (adsorbent). As a result there is a change in conductivity/resistivity of the sensor. It takes certain time for the resistivity/conductivity to get back to its normal value. This process is represented using reverse response recovery curves and is shown as the first block in the above figure.

Next stage is the amplification, sampling and quantization phase. The curves are first amplified in order to increase the resolution so that minute changes could be observed. This is followed by sampling and quantization. These values obtained in total constitute the input data space.

Next comes the feature extraction and other pre-processing tasks stage. Generally, the response of each sensor is assumed to be independent of the response of other sensors in that array. As a result these multi-sensor responses are encoded into high dimensional numerical values. Each dimension in a given value is considered as an electronic marker or identifier generated at any instant from a particular sensor in the array for one of the test gas. Now it is the task of feature extraction techniques to appropriately extract meaningful information so as to reduce the computational burden for the next stage.

The next stage is the classifier which does the task of taking a decision and classifying an input based upon the markers presented to it in a suitable form. A classifier does its task with the help of a cost function defined in terms of the statistical properties of the input data. What a classifier does is making decisions in order to minimize the total cost incurred. Various pattern classifiers are there for pattern classification problems with each having some merits and demerits of their own thus, making a choice among them is quite a situation dependent task.

1.2 Literature Review

Persaud and Dodd [1] along with Ikegami & Kaneyasu [2] were the first to report their thoughts regarding the mechanisms in the olfactory system found in mammals using an artificially designed nose model system. Zaromb & Steller [3] proposed the use of an array of sensors with partially overlapping sensitivities such that, each of the sensors respond broadly to a range or class of gases rather than to a specific one. Such an array could increase the working domain of such a system many times as compared to a single sensor approach.

It was the pioneering work of Gardener [4] who combined the concept of sensor array response with pattern analysis techniques and laid the foundation stone of a concept known as an electronic nose and opened a vast area of study for researchers. Although some thoughts regarding these systems were there before his work, it was him who moulded those ideas into a more concrete form.

Dably & Shurmer [5] and later Shurmer & Gardner [6] along with Osuna [7] showed a glimpse of the possible areas of application these systems could possess. In their papers they asserted that many complex tasks like identification, comparison etc. are possible

through appropriate pattern analysis techniques. Hoffheins & Lauf [8] on similar lines demonstrated that these sensor responses can be considered as fingerprints of a particular gas. Shurmer et al. [9] in their paper demonstrated the advantage of an integrated sensor array approach over individualistic array approach further consolidating this sensor array approach.

At the end of 80's and the decade of 90's saw the application of many statistical pattern recognition techniques. In another work by Shurmer et al. [10] the authors demonstrated the effectiveness of correlation and cluster analysis methods to discriminate alcohols and tobaccos. In an independent works Gardner et al. [11-12] along with Shurmer & Gardner [6] too used correlation for discrimination purpose and validated these techniques.

PCA found one of its first application in this field in the paper by Gardner [13]. In this he applied PCA and cluster analysis in tandem to detect vapours and odours from a multisensor array. M. Pardo et al. [14] in their classical work used hierarchical modelling in which they subdivided the total problem into subclasses i.e. the individual classes were divided into subclasses followed by classification using PCA and again clubbing the sublclasses. PCA has always shown its efficacy as a classification technique. However, application of PCA as a sensory information redundancy removal technique and as a pre-processor only has been more recent. First example being work reported by Ciosek et al. [15]. In this paper authors have used PCA to select dominant sensors from the total sensor array thus redundancy removal or pre-processing. Similar application of PCA has been done by Jesús Brezmes et al. [16] and Jesús Lozano et al. [17] for various type of identification tasks. Latest application of PCA in electronic nose data processing has been done by Jha et al. [18]. In this work PCA has been used to extract features based upon SVD matrix.

Many other pattern recognition techniques have been used to perform the specified task. G. Horner and Hierod [19] have used partial model building as a pattern recognition tool. In various works by various authors varied techniques have been applied [20-23] but all lack in some sense due to their inability to handle non-linearity situations.

On the other hand a lesser known ANN technique the Independent Component Analysis has shown good results in multivariate signal processing arena. In the work done by Natale et al. [24] the authors have demonstrated the noise counteraction capability of ICA. In another work by kermit [25] the author has shown the drift counteraction capability of ICA. In this work they have reported ICA better in comparison to PCA.

Fuzzy logic has emerged of late as a very promising tool for biological information processing owing to its proximity to real world ambiguous situations. Wide et al. [26] has shown the use of fuzzy sets for air quality monitoring applications. In another work by Bargagna et al. [27] characterization of olive oils has been done using the concepts of fuzzy theory. Application of fuzzy logic has also been demonstrated in a classical work by Sundic et al. [28] where the authors have used fuzzy logic for triggering of domestic gas alarms.

In an extraordinary work done by Keller et al. [29] the authors have integrated the concepts of fuzzy logic and k-nearest neighbour algorithm together. Thus a new type of classifier came into being with large application domain.

The use of popular supervised techniques have shown varied amount of efficacy however unsupervised classification techniques still have ample scope of application. Self-organizing-maps developed by Kohonen [30] in pattern classification problems is a step in that direction. In general these are cartographic representations of statistical features of a data set. Marco et al. [31] have successfully applied Self-organizing maps for gas identification. In a different work by Natale et al. [32] the authors have implemented SOM for performance comparison of a sensor array with a trained human panel. However application of SOM is still limited to small number of cases. SOM faces convergence problems in the case of complex data inputs.

1.3 Gaps In Study

Though ICA algorithm is better equipped in solving classification problems as compared to PCA due to the higher order statistics involved in it, no such significant work has been reported on the performance comparison of PCA and ICA as pre-processors when clubbed with a KNN classifier i.e. PCA-KNN versus ICA-KNN using the response of a poorly selective sensor array.

There is always a scope for exploration and exploitation of fuzzy nature in the data whenever high correlation along with unclear class boundaries are the characteristic features of a data set. No such significant work has been reported on the application of Fuzzy-KNN algorithm for the classification of individual gases using highly complex data with high inter class correlation and intra class sparseness, which is the data of application

in this work. There is a scope of solving classification problem using kernel approach i.e. Support Vector Machine and a unsupervised pattern classification approach namely Self-Organizing Maps using the response of the given poorly selective solid state sensor array data used Implementation of PCA using neural network approach i.e. PCA using Generalized-Hebbian Algorithm is an another aspect that has not been analysed to the optimum till now. This algorithm is important because of its hardware implementation capabilities.

1.4 Data Acquisition

An array of sensors was prepared and exposed to four different gases/odours. The sensors were fabricated by thick film screen printing technique at [Centre for Research in Microelectronics (CRME) IIT-BHU, Varanasi India]. A gas sensitive tin oxide paste was prepared and printed on to an alumina substrate. The paste was doped with 1 % Pd, Pt, Cuo, Zno and Cd. Thus, five of six sensors were doped with the above mentioned material, while the sixth sensor was left undoped. Gold electrodes (ESL 8080) were also printed along with gas sensitive layer of tin oxide and were subsequently fired onto alumina substrate. SAMCO plasma deposition system (model no. 10) was employed to generate oxygen plasma at low pressure by applying r.f. power at 13.56 MHz. The test gases (LPG, CCL_4, CO and C_3H_7OH) were injected into a test chamber of volume 2894.7 ml which contained the array. The resistance variations of all the sensors of array were measured simultaneously with time through a counter decoder circuit. Different concentrations (viz. 25 ppm, 50 ppm, 75 ppm, 100 ppm) of the test gases were injected into the chamber. Fig. 1.2 (a)-(d) depict the dynamic response of the sensor array to four test gases at different values of concentration .It was observed after experimentation that Cd doped sensor did not show any significant sensitivity towards any of the test gases. Therefore, the dynamic response curves have been sampled only for the response of other five sensors.

The dynamic response curves of Fig. 1.2(a)-(d) are actually response recovery plots. When the test gas is adsorbed on the sensor surface, the conductivity of the sensor first increase and then reduces back to its original value. The time required for the test gas to reduce the conductivity of the sensor to its background conductance is known as recovery time. It has been observed that as the concentration of the test gas is increased, the recovery time increases. The experimental results thus obtained have already been reported [37] and in

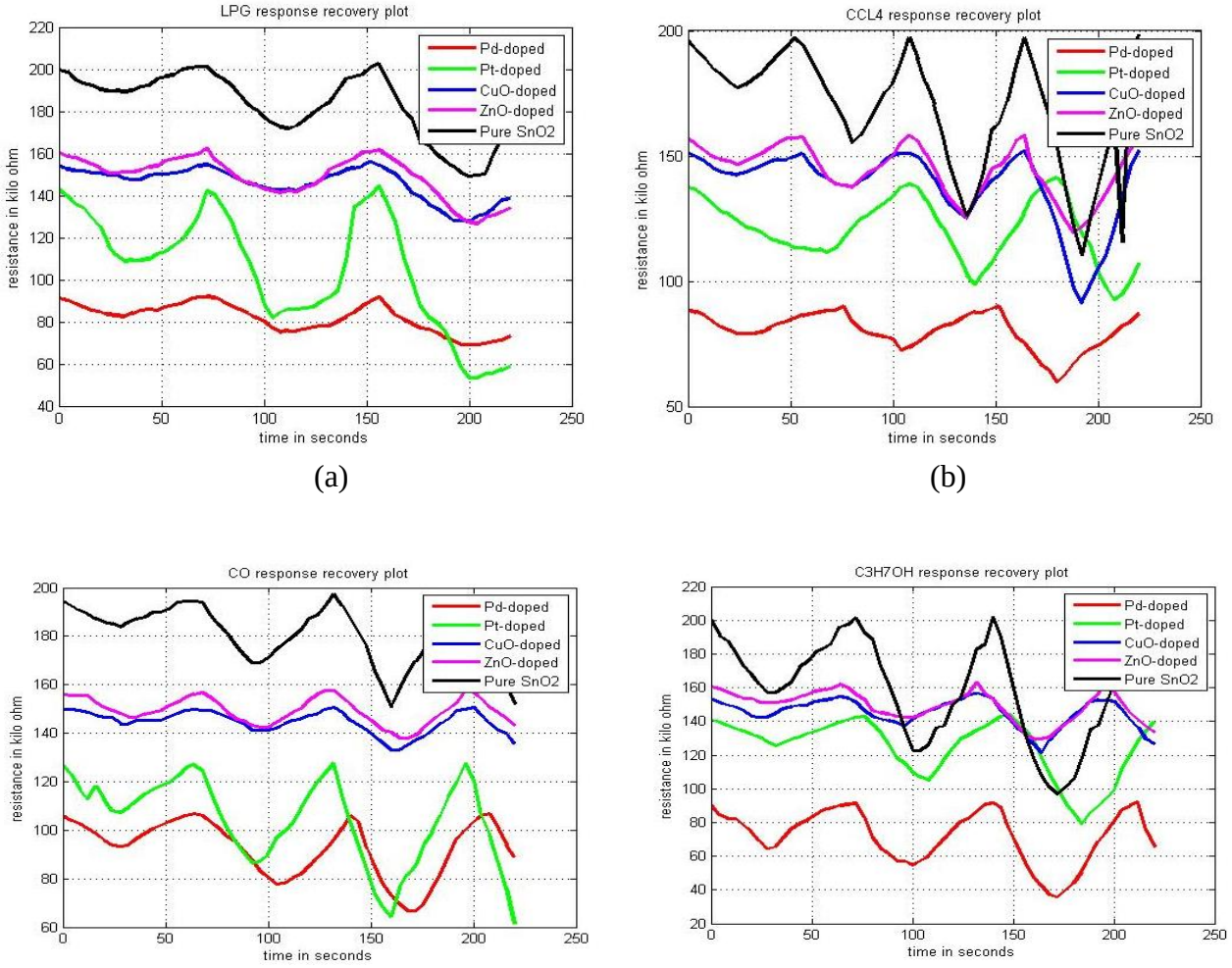


Fig. 1.2 Response-recovery plot of the array for different concentrations of the four gases: (a) LPG, (b) CCL_4, (c) CO, and (d) C_3H_7OH.

this work the same response has been used to extract data for gas/odour identification task. Due to their room temperature operation, oxygen plasma treated thick film sensors have a clear edge over conventional thick film sensors. However, this advantage is negated by their poor selectivity. Application of an appropriate computational technique can overcome the limitation of poor selectivity. This has served as a motivation to employ various pattern analysis techniques like PCA, ICA, KNN, Fuzzy KNN, SVM, GHA and Self-Organizing Map etc. for identification of individual gases as described in the subsequent chapters

1.5 Problem Formulation

Metal oxide based sensor arrays form an integral part of a typical electronic nose system, reason being cost effectiveness and broad sensitivity of sensors [33-35].Metal oxide films are promising materials for implementation of solid-state gas sensors [36]. However, they suffer from some inherent limitations like cross-sensitivity leading to poor selectivity, saturating tendency at higher concentrations and high drift. Overcoming poor selectivity of the sensor array remains the most challenging task if an E-nose system is to be realized. Thus, there arises a strong need for an efficient computational model which could take into picture the high correlation in data due to the cross sensitivities of sensors while performing classification task. Thus drawing sharp decision boundaries between odour classes requires a highly efficient and problem independent computational paradigm.

Emergence of the field of soft computing has paved the way for appropriate modelling and optimization of these complex systems. Soft computing techniques primarily consist of artificial neural networks (ANN), fuzzy logic and evolutionary computing. With their ability to adapt, learn, massive parallelism and their conceptual resemblance to biological neural networks, ANNs are the optimal choice in our attempt to electronically model the human sensory perceptions.

Chapter2: It gives a brief description and comparison of the popular pattern classifiers

Chapter3: Detailed performance comparison of pre-processing techniques like Principal Component Analysis and Independent Component Analysis has been done in terms of their classification efficiency when used prior to a k-nearest neighbour classifier

Chapter4: Fuzzy properties of the data have been explored and exploited by using Fuzzy k-nearest neighbour algorithm. The fuzziness in the data has been used in such manner that the results are highly consistent throughout

Chapter5: In this chapter the classification problem has been tackled using a Support Vector Machine. Training data after pre-processing has been fed to the machine to create the decision hyperplane for each two gas combination. Finally, through majority voting technique the test samples have been classified to a particular class.

Chapter6: In this chapter titled Eigen filtering using GHA, the MATLAB simulation of another PCA extraction technique known as GHA. It is a neural network approach for

extraction of principal components. This technique is more suitable than its counterpart from hardware implementation point of view.

Chapter7: Includes development of a Self-Organizing Map for the sensor data

Chapter8: Conclusions and future scope of this work

1.6 Novel Aspects of this Thesis

To the best of author's knowledge the following aspects of this thesis can be considered novel:

1. Most of the work in this field has been based upon conventional thick film sensors. These sensors don't perform well at room temperature. In our case the input data belongs to oxygen plasma treated thick film oxide sensors which perform very well at room temperature. However the cost that has to be paid in return is poor selectivity of the sensors. As a result data obtained is highly correlated and classification problem becomes tough.

2. It is a first step towards blind separation of odor samples. It paves the way for the realization of a real time odor monitoring system.

3. Most of the work involving application of ICA algorithm has been performed using conventional ICA packages available for example the "Fast ICA package" developed by Helsinki University. But in this work no such pre-written package has been used and a separate simple code has been written for simulation purpose.

4. Authors till now have primarily used an oft-repeated methodology i.e. training on ANN with BP network. Going beyond Back-Propagation trained Artificial Neural Network is another novel aspect incorporated in this work.

5. Application of both supervised and unsupervised classification techniques on poorly separable sensor data. In this work both types of techniques are used for classification purpose. PCA and ICA being the unsupervised ones while k-NN and SVM representing the supervised ones. Thus the combining these techniques we can call overall learning as semisupervised

6. Incorporating the principle of self-organization to chemical sensor signal processing.

Chapter 2 - Theory of Classifiers

Pattern Classification is the process of assigning a given object an identity by assigning it to a particular class having members more similar or alike than in any other class. Thus it is the task of identifying and quantifying all the costs involved and taking decisions in order to minimize the total cost and dissimilarity. A classifier does the above job by evaluating the evidences present and by comparing the cost of each and every possible decision.

The coming sections give a critical comparison between five most popular patter classifiers.

2.1 Bayesian Decision Theory/ Parametric Estimation [38]

Bayesian decision theory is one of the most fundamental parametric analytical tool in understanding the statistical aspects of pattern classification problem. It is based upon a model which is probabilistic in nature and casted out of quantified space such that numerical values denote a trade-off between various decisions and the cost associated with each of them.

Bayesian decision theory provides a framework for classification of features into a particular class in terms of their likelihood or belongingness to that class and the prior knowledge of their occurrence.

Let **x** be a feature vector in a d-dimensional feature space with each of the individual d dimensions denoting an independent feature of the input space.

Let $\{\omega_1, \omega_2 \dots \dots \dots \omega_c\}$ be the set of c classes and let $\{\alpha_1, \alpha_2 \dots \dots \dots \alpha_a\}$ be the set of a possible actions with inequality $a \geq c$ in order to include state of no decision. The posterior probability can be computed as:

$$P(\omega_j/\mathbf{x}) = \frac{p(\mathbf{x}/\omega_j)P(\omega_j)}{p(\mathbf{x})} \tag{2.1}$$

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}/\omega_j)\, P(\omega_j) \tag{2.2}$$

$P(\omega_j/\mathbf{x})$ denotes the posterior probability or the probability that the class is ω_j given where feature vector **x** is already known

$P(\omega_j)$ denotes the prior probability of occurrence of class ω_j

$p(\mathbf{x}/\omega_j)$ denotes the state-conditional probability distribution/mass function

$p(\mathbf{x})$ denotes the distribution of **x**

$$R(\alpha_i/\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i/\omega_j)\, P(\omega_j/\mathbf{x}) \tag{2.3}$$

$\lambda(\alpha_i/\omega_j)$ is the loss incurred by taking action α_i when the true class is ω_j

$R(\alpha_i/\mathbf{x})$ is the expected loss incurred when action α_i is taken after observing feature vector **x**

In Bayesian classification the basic motive is to frame a decision function α(**x**) such that the overall risk given by

$$R = \int R(\alpha(\mathbf{x})/\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{2.4}$$

is minimized

The Bayes decision rule finally assigns any feature vector x to a particular class such that

$$R(\alpha_i/\mathbf{x}) < R(\alpha_j/\mathbf{x}) \qquad \forall\, j \neq i \tag{2.5}$$

Although the Bayesian classifier gives least error the applicability of such a tool is very limited. This vulnerability arises due to the inherent underlying assumptions on which the foundation stone of the theory has been laid i.e. the knowledge of priori probabilities $P(\omega_j)$ and the class-conditional density $p(\mathbf{x}/\omega_j)$. Thus for the application of this model their estimation becomes inevitable.

Though there are several methods to estimate these parameters. The most common being the

1. Maximum-likelihood estimation

2. Bayesian parameter estimation

The maximum likelihood estimation method assumes that though the distribution $p(\mathbf{x}/\omega_j)$ is not known precisely, it has a known parametric form uniquely defined by a parameter vector $\boldsymbol{\theta_j}$ thus $p(\mathbf{x}/\omega_j)$ is represented in parametric form as $p(\mathbf{x}/\omega_j, \boldsymbol{\theta_j})$ and to have an estimate of this we use the training set $\mathbb{F}$ to estimate the parameter vector $\boldsymbol{\theta}$

Let us consider a single and independent set $\mathbb{F}_j$ containing n samples $\{\mathbf{x}_1, \mathbf{x}_2 \ldots \ldots \mathbf{x}_n\}$ drawn independently and belonging to class ω_j

$$p(\mathbb{F}/\boldsymbol{\theta_j}) = \prod_{k=1}^{n} p(\mathbf{x_k}/\boldsymbol{\theta}) \tag{2.6}$$

The maximum-likelihood estimate of $\boldsymbol{\theta}$ is the value $\boldsymbol{\theta}^m$ that maximizes the above equation i.e.

$$\boldsymbol{\theta}^m = arg \max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) \tag{2.7}$$

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln p(\mathbf{x_k}/\boldsymbol{\theta}) \tag{2.8}$$

For maximization $$\nabla_{\boldsymbol{\theta}} l = 0 \text{ where } \nabla_{\boldsymbol{\theta}} \equiv \begin{matrix} \frac{\partial}{\partial \boldsymbol{\theta}_1} \\ \vdots \\ \frac{\partial}{\partial \boldsymbol{\theta}_p} \end{matrix} \tag{2.9}$$

It can easily be observed that the solution requires differentials which may be quite cumbersome.

In the Bayesian parameter estimation method we consider theta to be a random variable with known distribution function unlike the maximum likelihood estimation in which the vector theta is considered to be a constant. Writing formally

$$p(\mathbf{x}/\mathbb{F}) = \int p(\mathbf{x}, \boldsymbol{\theta}/\mathbb{F})\, d\boldsymbol{\theta} \tag{2.10}$$

$$p(\mathbf{x}/\mathbb{F}) = \int p(\mathbf{x}/\boldsymbol{\theta})\, p(theta/\mathbb{F}) d\boldsymbol{\theta} \tag{2.11}$$

$$P(\omega_i/\mathbf{x}, \mathbb{F}) = \frac{p(\mathbf{x}/\omega_i, \mathbb{F}) P(\omega_i/\mathbb{F})}{\sum_{j=1}^{c} p(\mathbf{x}/\omega_j, \mathbb{F}) P(\omega_j/\mathbb{F})} \tag{2.12}$$

$$P(\omega_i/\mathbf{x},\mathbb{F}) = \frac{p(\mathbf{x}/\omega_{\boldsymbol{i}},\mathbb{F}_i)P(\omega_i)}{p(\mathbf{x}/\omega_{\boldsymbol{j}},\mathbb{F}_j)P(\omega_j)} \tag{2.13}$$

$$P(\omega_i/\mathbf{x},\mathbb{F}) = \frac{p(\mathbf{x}/\mathbb{F}_i)P(\omega_i)}{p(\mathbf{x}/\omega_{\boldsymbol{j}},\mathbb{F}_j)P(\omega_j)} \tag{2.14}$$

$$P(\boldsymbol{\theta}/\mathbb{F}) = \frac{p(\mathbb{F}/\boldsymbol{\theta})P(\boldsymbol{\theta})}{p(\mathbb{F},\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{2.15}$$

Even though such parameter estimation techniques exist their application is tedious and computationally complex. In most of the practical cases even the basic assumptions that these two methods make are not fulfilled. The Bayesian estimation method is quite complex and its applicability is very limited. The results obtained by Bayesian estimation method are more complicated and harder to understand. As a result in practical cases where these priori probabilities are unavailable, these classifiers are not of much help.

2.2 Non Parametric Estimation Techniques/Classifiers [38]

Non-parametric estimation method generally involve two most important methods/algorithms namely the

1. Parzen Window method and

2. k- Nearest Neighbour method

In non-parametric techniques there is no such requirement of prior knowledge of probability distributions. All we require is a labelled data set and that too can be of arbitrary random nature as a result no assumption is required to be made about the underlying densities.

Non parametric techniques try to estimate the density function from the given samples. Larger the data set better is the distribution estimation.

The basic idea behind any classification theme is to calculate the probability of occurrence or falling of a feature vector **x** in various regions and classifying that vector to the region with highest probability. Let $R1$ & $R2$ be the boundaries of a given region. Then the probability of the feature vector lying in the region is given by

$$P = \int_{R1}^{R2} p(\mathbf{x}')\, d\mathbf{x}' \tag{2.16}$$

Let us suppose n samples $\{\mathbf{x}_1, \mathbf{x}_2 \dots \dots \mathbf{x}_n\}$ are drawn independently with some probability distribution. The probability that k of these fall in some region with boundaries $R1$ & $R2$ can be considered as a binomial distribution and in the limiting case the probability distribution

$$p(\mathbf{x}) \cong \frac{k/n}{V} \tag{2.17}$$

The binomial distribution peaks at mean as a result k/n is a good estimate of the probability P and in the limiting case with very small region where distribution is considered constant the above equation can be written as

$$P = \int_{R1}^{R2} p(\mathbf{x}')\, d\mathbf{x}' \cong p(\mathbf{x})V \tag{2.18}$$

The binomial distribution peaks at the expected value for k that is nP. Equating we get the function of probability distribution given above.

Formally stating the concept of non-parametric distribution, the probability function

$$P_n(\mathbf{x}) \cong \frac{k_n/n}{V_n} \tag{2.19}$$

Where k_n is the number of samples falling in the region R_n with volume V_n and $P_n(\mathbf{x})$ being the estimate of probability distribution. The three necessary conditions must for convergence of probability $P_n(\mathbf{x})$ to $P(\mathbf{x})$ are

1. $\lim_{n \to \infty} v_n = 0$
2. $\lim_{n \to \infty} k_n = \infty$
3. $\lim_{n \to \infty} k_n/n = 0$

There are in general two common ways of satisfying these two conditions

The first one is to shrink an initial region by specifying the volume v_n as some function of n and in the second method we specify n and let the volume grow until it captures k_n samples. The former explains the essence of Parzen Window method while the later is the k-Nearest Neighbour method for pattern classification.

Although both these methods are classical examples of non-parametric estimation, due to lesser complexity of the algorithm the k-nearest neighbour approach is a better option as compared to its counterpart

The k-nearest neighbour rule relies on the assumption that the probability that any unclassified feature vector belongs to a particular class is directly proportional to the number of training vectors of that particular class lying near to it in a given neighborhood. Finally for any test vector, the class having the highest probability wins and the test vector is assigned to that particular class.

In Parzen window estimation method we consider a region R_n which is a d-dimensional hypercube. If h_n is the length of an edge of that hypercube then its volume is given by

$$V_n = {h_n}^d \tag{2.20}$$

$$\varphi(\mathrm{u}) = \begin{cases} 1 & \text{if } |u_j| \leq 0.5; \\ 0 & \text{otherwise} \end{cases} \qquad j = 1,2 \ldots \ldots d \tag{2.21}$$

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x} - \mathbf{x_i}}{h_n}\right) \tag{2.22}$$

$$P_n(\mathbf{x}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{1}{h_n} \varphi\left(\frac{\mathbf{x} - \mathbf{x_i}}{h_n}\right) \tag{2.23}$$

The k-nearest neighbour method is one of the most intuitive and simple approach to solve the problem of classification

k-Nearest Neighbor Algorithm [29]

BEGIN

Input **x**, of unknown classification

Set k, $1 \leq k \leq n$

Initialize i =1.

DO UNITIL (k - nearest neighbors to **x** found)

Compute distance from **x** to $\mathbf{x}_i$.

IF ($i \leq$ K) THEN

Include $\mathbf{x}_i$ in the set of k-nearest neighbors

ELSE IF ($\mathbf{x}_i$ closer to **x** than any previous nearest neighbor) THEN

Delete the farthest of the **x**-nearest neighbors

Include $\mathbf{x}_i$ in the set of K-nearest neighbors

END IF

Increment i

END DO UNTIL

Determine majority class among all using K nearest neighbors

IF(a tie is there) THEN

Compute sum of distances of neighbors in tied classes and assign to the one having lesser

ELSE

Classify **x** to the dominant class

END IF

END

Thus in other words Let $\mathbf{x}_i \in \mathbf{X} \ \forall \ i = \{1,2 \ldots \ldots . n\}$ be the set of n labelled prototypes and let $k = \{1,2 \ldots . . n\}$ and let number of classes $c = \{2 \ldots . . n\}$. Let k_i be the number of feature training vectors for the j_{th} class near to the test vector. The testing data is assigned to the class k_j if and only if $k_i > k_j \ \ \forall \ j \neq i$. The situation of tie can be handled by minimum distance criteria

The k nearest neighbour rule can achieve consistently high performance without priori assumptions about the probability distribution/mass functions of the training samples. The complexity of the k nearest neighbour approach can be improved by certain assumptions and methods. The ease of application and high efficiency of this technique easily overcomes the complexity issue involved in the algorithm. As stated above this technique does not require a priori knowledge of the distribution/mass function, it becomes a favourable condition for quite a large class of classification problems.

2.3 Support Vector Machine [39]

Support Vector Machine can be considered as a black box which can take features in data space as inputs and constructs a decision surface in the form of a decision hyperplane capable enough to maximize the margin of separation between positive and negative features.

$$w^T\mathbf{x} + \mathrm{b} = 0 \tag{2.24}$$

where $\mathbf{x}$ is an input vector, w is the weight vector and b is the bias applied

This can be further divided into two categories i.e. Linear SVM and Non-linear SVM

Linear Support Vector Machines are a useful tools in finding optimal hyperplane as decision boundry in case of linearly separable patterns,

The constraint equation becomes:

$$d_i(\boldsymbol{w^T}\mathbf{x_i} + \mathrm{b}) \geq 1 \qquad \text{for } i = 1,2 \ldots \ldots, N \tag{2.25}$$

where d_i is the class label with values ∓ 1

The SVM finds the optimal hyperplane in the form of the above equation so as to maximize the margin of separation i.e. distance between closest data points.

In nonlinear case such a simple implementation like in linear SVM is not possible. In this case due to the intrinsic features of the input pattern no such plane is possible that can act as a decision surface without inducing any error. As a result an optimization to find the least error hyperplane is the objective. The above equation in nonlinear case becomes

$$d_i(\boldsymbol{w^T x_i} + \mathrm{b}) \geq 1 - \xi_i \qquad \text{for } i = 1,2 \ldots \ldots, N \tag{2.26}$$

where ξ_i are called slack variable and this variable denotes the deviation from ideal behaviour i.e. the linearly separable case. It is suffice to comment here that the overall goal is to find a decision surface such that the error of misclassification averaged over the entire data space is minimized mathematically it is the minimization of equation:

$$\Phi(\xi_i) = \sum_{i=1}^{N} I(\xi_i - 1) \tag{2.27}$$

$$\text{Where } I(\xi_i) = \begin{cases} 0, & \xi \leq 0 \\ 1, & \xi > 0 \end{cases} \tag{2.28}$$

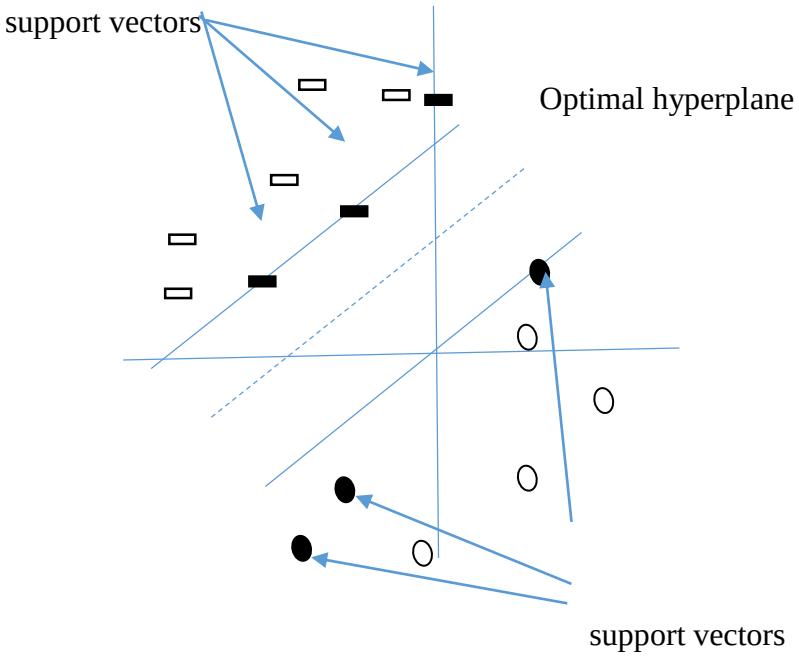


Fig. 2.1 Linear Support Vector Machine

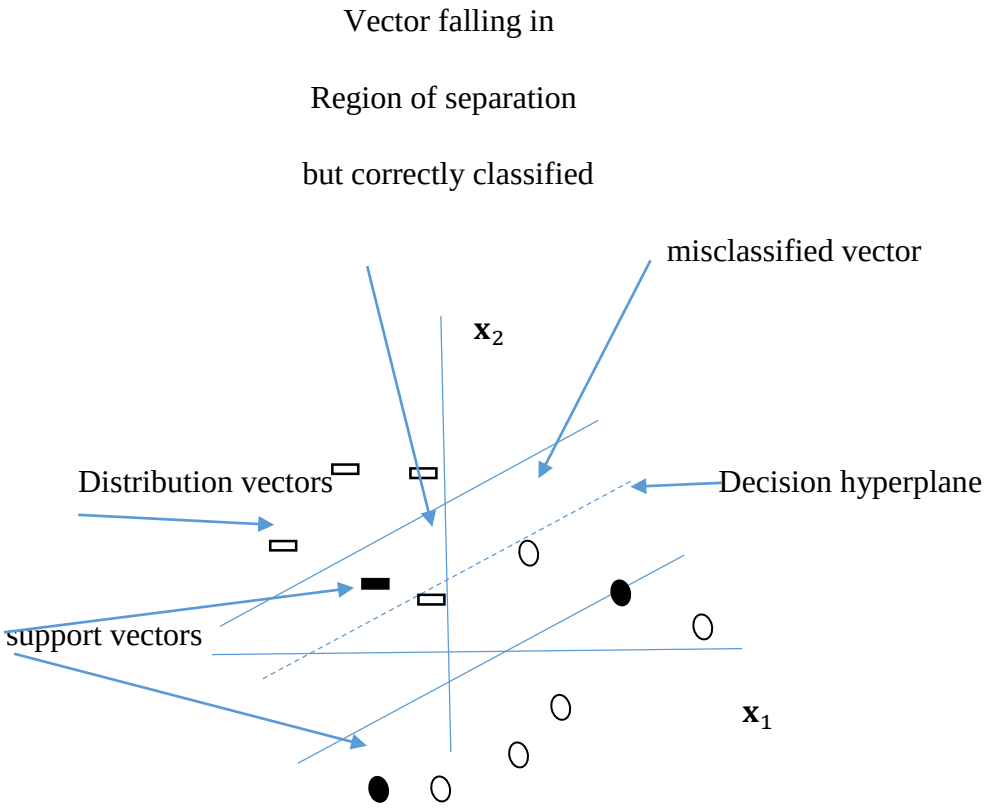


Fig. 2.2 Non-linear Support Vector Machine

Type of Classifier	Distribution Function	Complexity & Efficiency
1.Bayesian Decision Theory a).Maximum Likelihood Estimation	Some Priori Knowledge always required	Highly complex but lesser than Bayesian, involves differentials and give results better than the Bayesian estimation method
b). Bayesian Estimation	---------do----------	Highly complex The overall theory gives best efficiency rate

2. Non-Parametric Estimation a) Parzen's Window Method	No priori-knowledge of distributions is required. However some conditions for convergence need to be fulfilled	High computational burden, moderate results
b) k- Nearest Neighbour method	---------do----------	Highly complex but certain techniques can decrease them considerably. Most practical approach. Probability of error twice than Bayesian (quite acceptable)
3) Support Vector Machine	No priori knowledge of distributions is required. Only training and testing samples required	One of the most optimum technique however heavily dependent upon the choice of kernel for classification of nonlinear vectors.

Table 1.1: Comparison of various classifiers

It is evident from the above discussion that k-nearest neighbour approach along with kernel techniques like SVM have an advantage whenever the data set is limited and distribution functions are unknown. In this work these two classifiers are chosen for classification and further analysis.

Chapter 3- Performance Comparison of PCA and ICA as Pre-processors to k-NN Classifier Using a Poorly Selective Sensor Array Response

3.1 Introduction

The input data vector in an odour classification system is the response of a sensor array. Being high dimensional in nature, this data cannot be fed directly to the classifier. With an increase in the dimensionality of the training data, there is an exponential increase in the number of training vectors required by the classifier to learn [7]. As the training data available to us is limited, to avoid performance degradation dimensionality reduction with least information loss is of utmost importance. This task of dimensionality reduction is generally performed using second order statistical tools. These tools are not applicable in certain cases. As a result higher order statistical tools are often required. The problem of redundancy becomes significant due to the cross-selectivity of the chemical gas sensors used. Co linearity of dimensions makes covariance matrix singular and noninvertible [7]. Some other small pre-processing steps like whitening etc. are also required in order to make mathematical analysis simpler and to discard insignificant input data.

Many of the feature extraction techniques for E-nose data have been based upon PCA. As a classification technique PCA has shown its efficacy both of its own and as a pre-processing stage to a subsequent classifier [14] [16-18]. On the other hand ICA too has shown promising results in the classification of multivariate data [16, 25]. Due to the higher order statistics involved in it, ICA handles sensor drift more efficiently as compared to PCA [25]. With the integration of K-nearest neighbour algorithm with PCA and ICA even better and steady results are possible.

3.2 PCA and ICA for Unsupervised Clustering/learning

3.2.1 PCA as an Unsupervised Learning Technique

PCA is considered as one of the most important tool in the study of unsupervised neural networks. It can easily extract important and meaningful information from a high

dimensional data with low class separability by reducing to a low dimensional one in order to reveal relevant underlying features

The goal of PCA is to re-express the given data in terms of a new basis so as to maximize the variances along the principal directions so computed.

Let $\boldsymbol{x}$ be a realization of a zero mean random variable $\mathbf{x}$ such that $\mathbf{x} \in R^m$

$$\mathbb{E}[\mathbf{x}] = 0 \tag{3.1}$$

And **C** be an m-by-m correlation matrix given by

$$C = \mathbb{E}[\mathbf{x}\mathbf{x}^{\boldsymbol{T}}] \tag{3.2}$$

The i^{th} principal component of any input vector $\boldsymbol{x}$

$$\boldsymbol{a_i} = \boldsymbol{x^T q_i} \quad i = 1,2 \dots \dots \dots \dots m \tag{3.3}$$

such that $\boldsymbol{q_i}$ is a vector satisfying two conditions

$$\|\boldsymbol{q_i}\| = 1 \tag{3.4}$$

$$\boldsymbol{C q_i} = \lambda_i \boldsymbol{q_i} \quad i = 1,2 \dots \dots \dots \dots m \tag{3.5}$$

Here λ_i is called the i^{th} eigenvalue corresponding to the i^{th} eigenvector $\boldsymbol{q_i}$ subject to the constraint $\lambda_i > \lambda_{i+1}$where each eigenvalue gives a measure of the variance along the principal direction been computed given by the corresponding eigenvector. Thus the data vector $\boldsymbol{x}$ can be truncated knowing the percentage variance along each principal direction or the relative importance of each principal direction so computed in terms of the amount of information it contains is given by

$$v_k = \lambda_k / \sum_{j=1}^{m} \lambda_j \quad k = 1,2 \dots \dots \dots m \tag{3.6}$$

3.2.2 ICA as an Unsupervised Learning Technique

3.2.2.1 Blind Source Separation

The essence of ICA algorithm lies in its ability to project any non-Gaussian data vector linearly onto new but not necessarily orthogonal directions so that the resulting components are as statistically independent as possible. Absence of a covariance structure in higher order statistical tool like ICA limits the application of Gaussian data as input.

Let there be a data vector **x** such that

$$\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \dots\dots\dots \mathbf{x_m}] \tag{3.7}$$

such that each individual component $\mathbf{x_i}$ is correlated with each other. The aim of the ICA algorithm is to estimate the mixing matrix $\boldsymbol{A}$ such that

$$\mathbf{x} = \boldsymbol{AS} \tag{3.8}$$

gives the previously unknown statistically independent unmixed source vector $\boldsymbol{S}$ where

$$\boldsymbol{S} = [\boldsymbol{S_1}, \boldsymbol{S_2}, \boldsymbol{S_3}, \dots\dots\dots \boldsymbol{S_m}] \tag{3.9}$$

Multiplying both sides of equation by $\boldsymbol{A^{-1}}$ we get

$$\boldsymbol{A^{-1}}\mathbf{x} = \boldsymbol{S} \tag{3.10}$$

Writing $\boldsymbol{A^{-1}} = \mathbf{W}$ and calling it as the un-mixing matrix we can obtain the original source vectors back

$$\boldsymbol{S} = \boldsymbol{WX} \tag{3.11}$$

The above theory gives the essence of Blind Source Separation problem

3.2.2.2 ICA Algorithms Classification

The ICA algorithm can be broadly classified into two families i.e. ICA algorithms based upon minimization of mutual information and the second being rooted upon the maximization of non- Gaussianity. The first family includes algorithm developed by Amari et al. [40] which is based upon Kullback-Leibler divergence, the algorithm given by Phan [41] which works on the maximum-likelihood estimation and the third being the information maximization algorithm given by Bell and Sejnowski [42] which tries to maximize the entropy of the system.

Second family includes the Fast ICA algorithm which uses negentropy as the measure of non- Gaussianity [43]. In our present work we have used the Fast ICA algorithm to solve the BSS problem. The actual algorithm proceeds as

Let $\boldsymbol{x}$ be a realization of a zero mean, whitened random variable **x** where

$$\mathbb{E}[\mathbf{x}] = 0 \tag{3.12}$$

$$\mathbb{E}[\mathbf{x}\mathbf{x}^{\boldsymbol{T}}] = I \tag{3.13}$$

If Condition (12) is not satisfied then subtract mean from **x** before proceeding further and for the fulfilment of (3.13) condition, **x** should be replaced by

$$\mathbf{x} = \boldsymbol{E}\boldsymbol{D}^{-1/2}\,\boldsymbol{E}^{T}\,\mathbf{x} \tag{3.14}$$

where $\boldsymbol{E}$ is the orthogonal matrix of eigenvector of $\mathbb{E}[\mathbf{x}\mathbf{x}^{T}]$ and $\boldsymbol{D}$ is the diagonal matrix of the corresponding eigenvalues i.e.

$$\boldsymbol{D} = \boldsymbol{diag}(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3 \ldots\ldots\ldots \boldsymbol{d}_m) \tag{3.15}$$

$$\boldsymbol{D}^{-1/2} = \boldsymbol{diag}\left(\boldsymbol{d}_1^{-1/2}, \boldsymbol{d}_2^{-1/2}, \boldsymbol{d}_3^{-1/2} \ldots\ldots \boldsymbol{d}_m^{-1/2}\right) \tag{3.16}$$

After the estimation of independent components the task of addition of mean back to the data can be performed by adding back $\boldsymbol{A}^{-1}\boldsymbol{u}$ to the estimated independent component matrix in case if first condition was not satisfied, where $\boldsymbol{u}$ is the mean of $\boldsymbol{X}$ that was subtracted initially.

3.2.2.3 ICA algorithm

Let there be any random value of weight vector $\boldsymbol{w}_i$ such that

$$\|\boldsymbol{w}_i\| = \mathbf{1} \text{ and } \boldsymbol{w}_0 = \mathbf{0} \qquad \boldsymbol{i} = \mathbf{1}, \mathbf{2} \ldots\ldots\ldots\ldots \boldsymbol{m} \tag{3.17}$$

$$\boldsymbol{w}_i^{+} = \mathbb{E}[\varphi'(\boldsymbol{w}_i^{T}\boldsymbol{x})]\boldsymbol{w} - \mathbb{E}[\boldsymbol{x}\varphi(\boldsymbol{w}_i^{T}\boldsymbol{x})] \tag{3.18}$$

$$\boldsymbol{w}_i^{+} = \boldsymbol{w}_i^{+} - \sum_{j=0}^{i-1} \boldsymbol{w}_i^{T}\boldsymbol{w}_j\boldsymbol{w}_j \tag{3.19}$$

$$\boldsymbol{w}_i^{+} = \boldsymbol{w}_i^{+} / \|\boldsymbol{w}_i^{+}\| \tag{3.20}$$

3.3 Data Interpretation

The response recovery curves shown in Fig.1.2 have been used as sample data vectors have been sampled at regular intervals of time to yield a raw data vector. Figs. 3.1(a) and 3.1(b) show the three dimensional scattered plots for the raw data taking different combinations of sensors at a time. In total ten such plots can be drawn for the five sensors used but for the sake of convenience only two such plots have been shown. These plots reveal the poor class separability of the raw data. Hence, it is necessary to apply some suitable pre-processing technique before the data is fed to the final classifier. Correlation plots between samples of any two gases have also been plotted. Out of six such plots possible, two are

being shown her for the sake of convenience as Fig. 3.1(a) and 3.1(b).

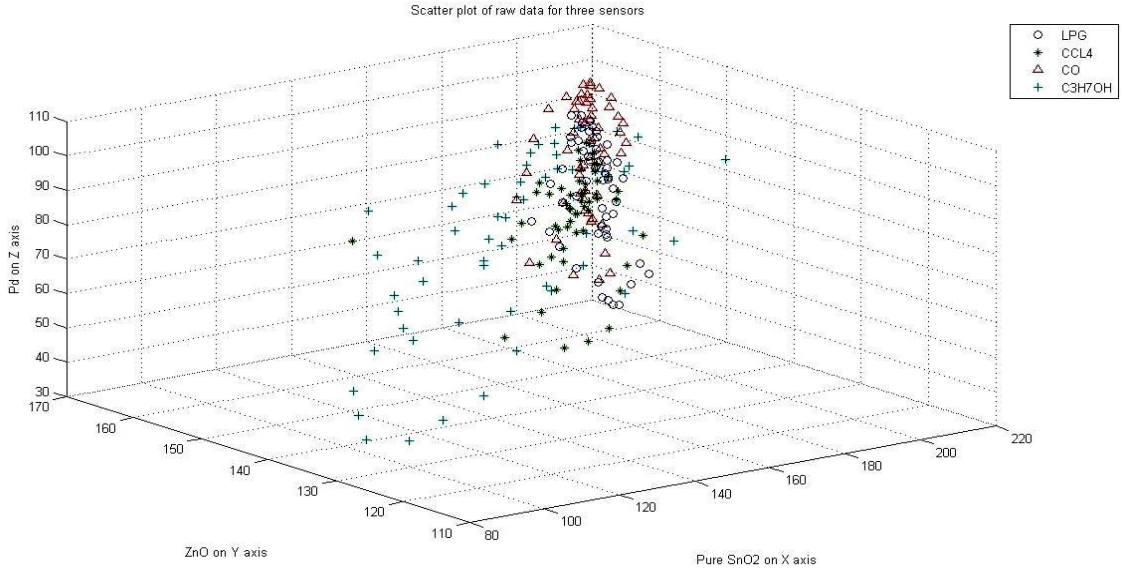


Fig. 3.1(a). Three dimensional raw data scattered plot with three sensors chosen at a time namely: SnO_2, ZnO and Pd

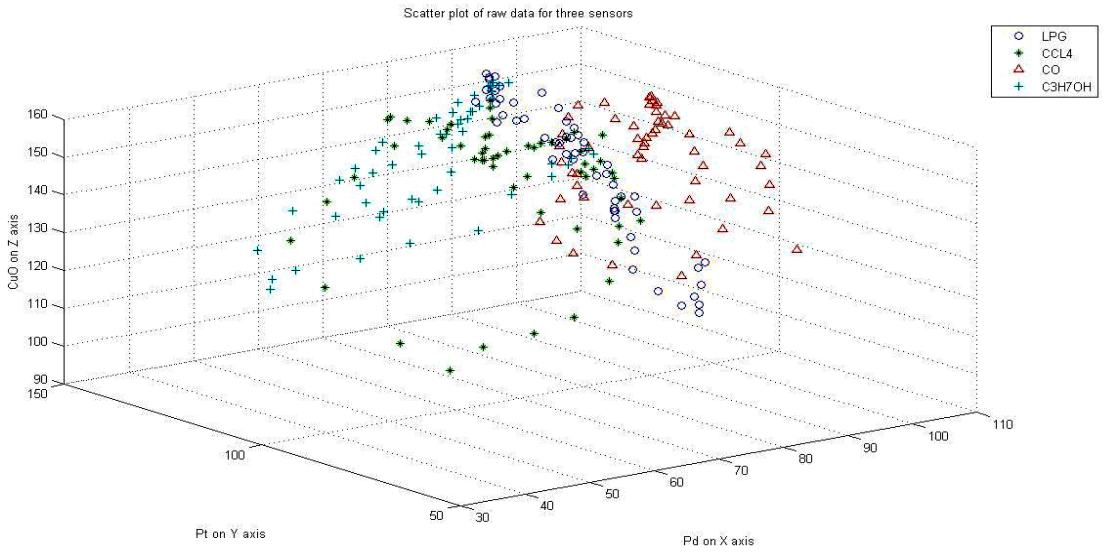
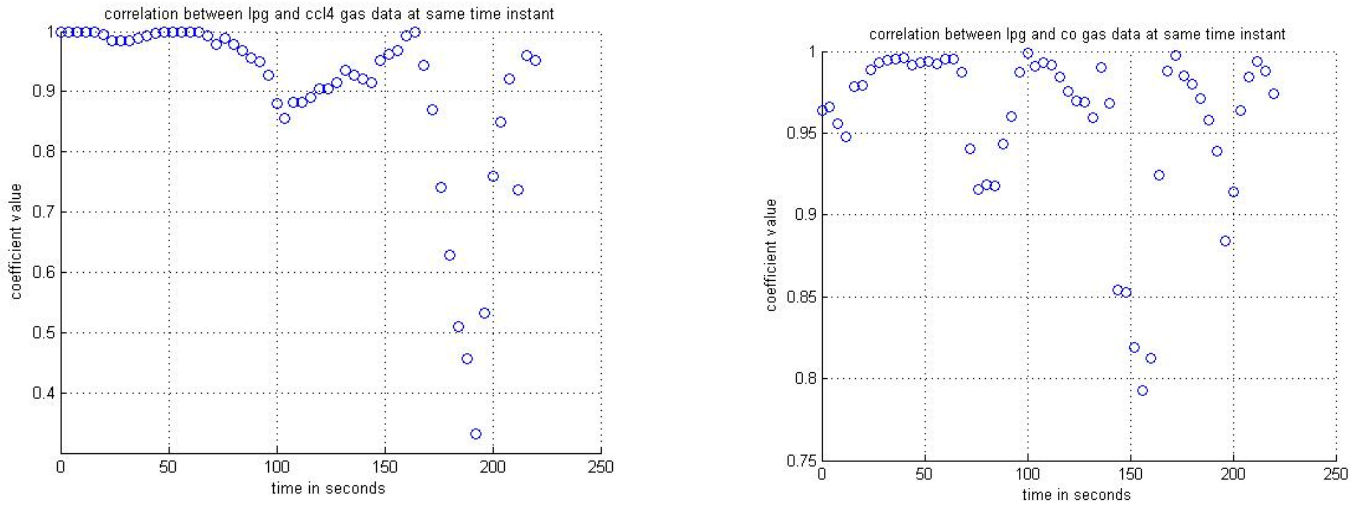


Fig. 3.1 (b). Three dimensional raw data scattered plot with three sensors chosen at a time namely: Pt, Pd, CuO

The present problem can be viewed as a blind source separation (BSS) problem, the motivation for which arises due to the following facts:

1) The raw data is highly correlated as evident from the correlation plots depicted in Fig. 3.2

2) The Non-Gaussianity of the data, making the present problem a suitable one for higher order statistical analysis as evident from the histogram plots in Fig. 3.3



(a) (b)

Fig. 3.2 Correlation between two randomly chosen gases from the four individual gases: (a) LPG and CCl_4 , (b) LPG and CO.

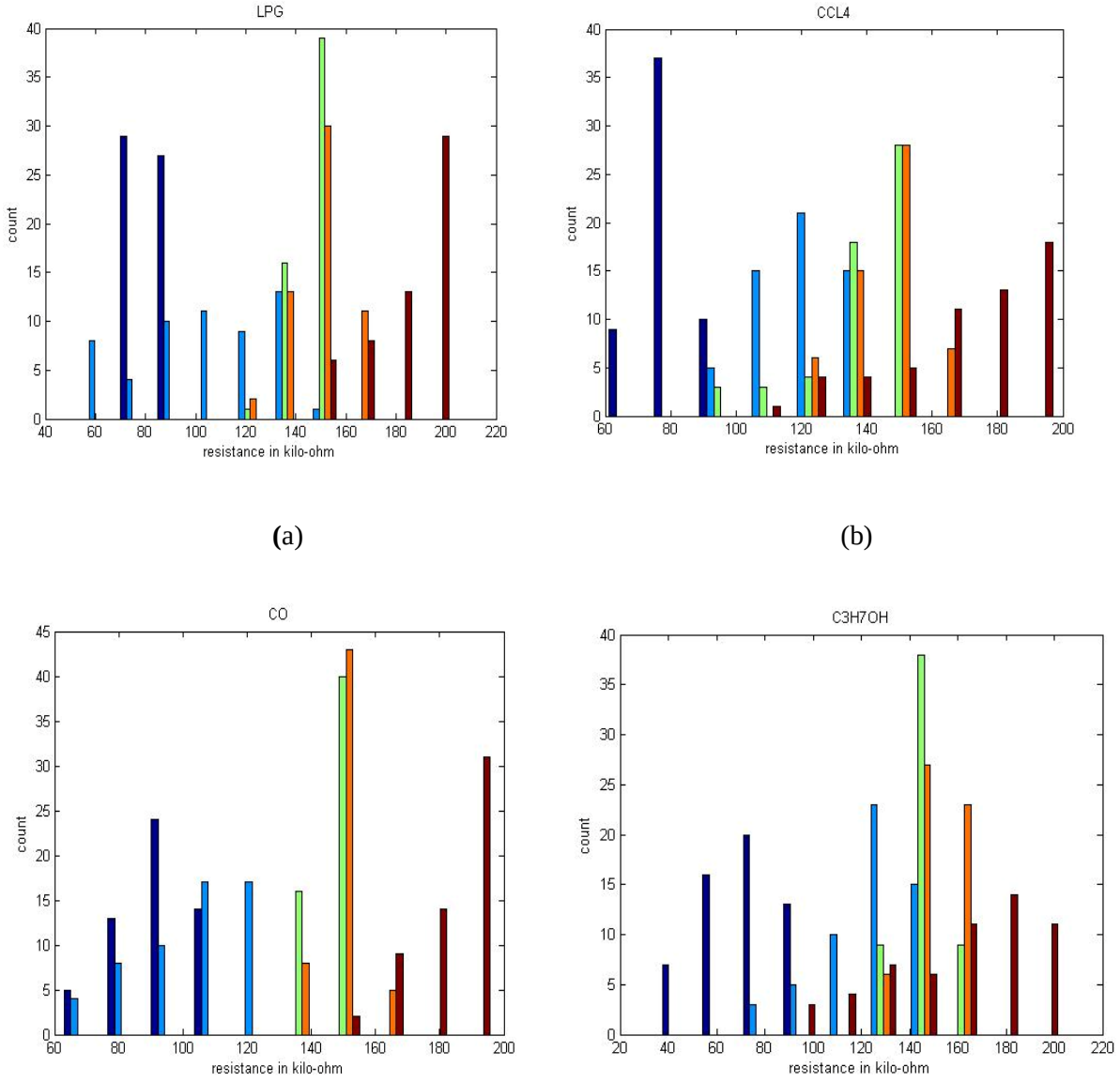
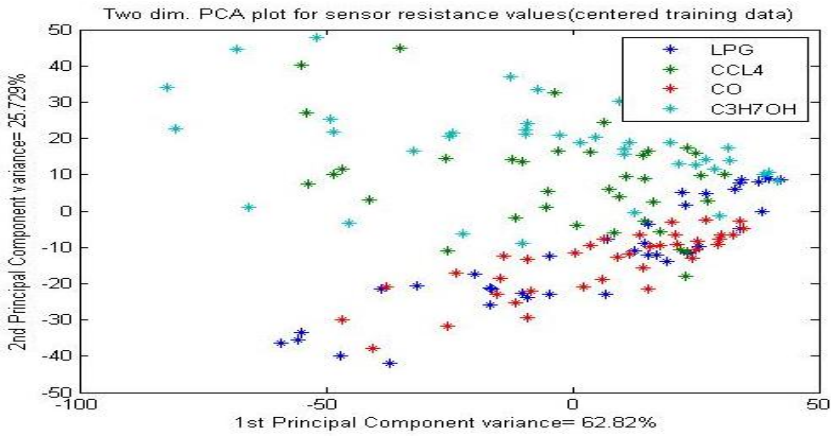


Fig. 3.3 Histogram plots for four individual gases (a) LPG, (b) CCL4 (c) CO and (d) C3H7OH

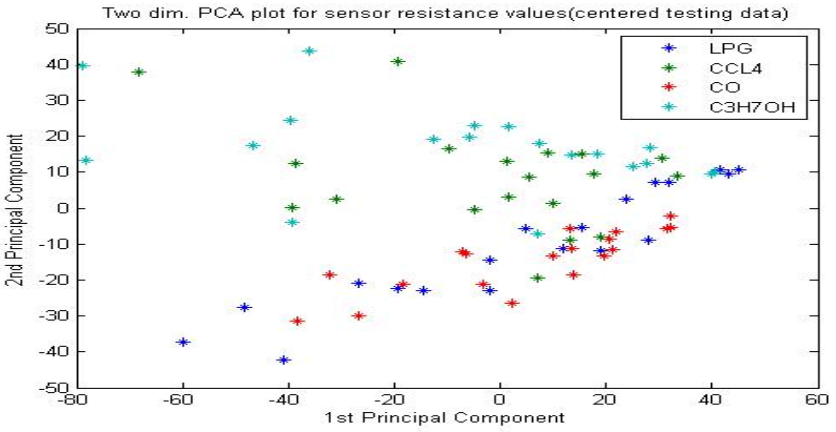
3.4 Simulation Results & Discussion

This work proceeds first with the application of PCA on the raw data followed by ICA on the same. Fig.3.4 show the results of PCA pre-processing for both training and testing vectors. Similarly, Fig.3.5 depicts two dimensional plots for individual independent components when ICA was applied. Though in the case of ICA ten two dimensional plots between five independent components can be plotted, only one has been plotted for convenience. Separate plots have been plotted for training as well as testing data. After this

the KNN algorithm has been applied on the data pre-processed by PCA and ICA respectively. The results of application of KNN algorithm can be seen in terms of classification efficiency plots for both the cases. It can be inferred from Fig. 7 (a) and 7 (b) that PCA pre-processed data gives higher classification efficiency only at higher values of K, making the classification process more computationally intensive. However, ICA pre-processed data gives a higher success rate even at lower values of K. Furthermore, the classification efficiency for the ICA- KNN classifier is never zero for any value of K for any gas unlike PCA-KNN classifier whose classification efficiency becomes zero for some gases at higher values of K. From the KNN plots it can be concluded that in general, KNN on ICA performs better in terms of classification efficiency as compared to KNN on PCA



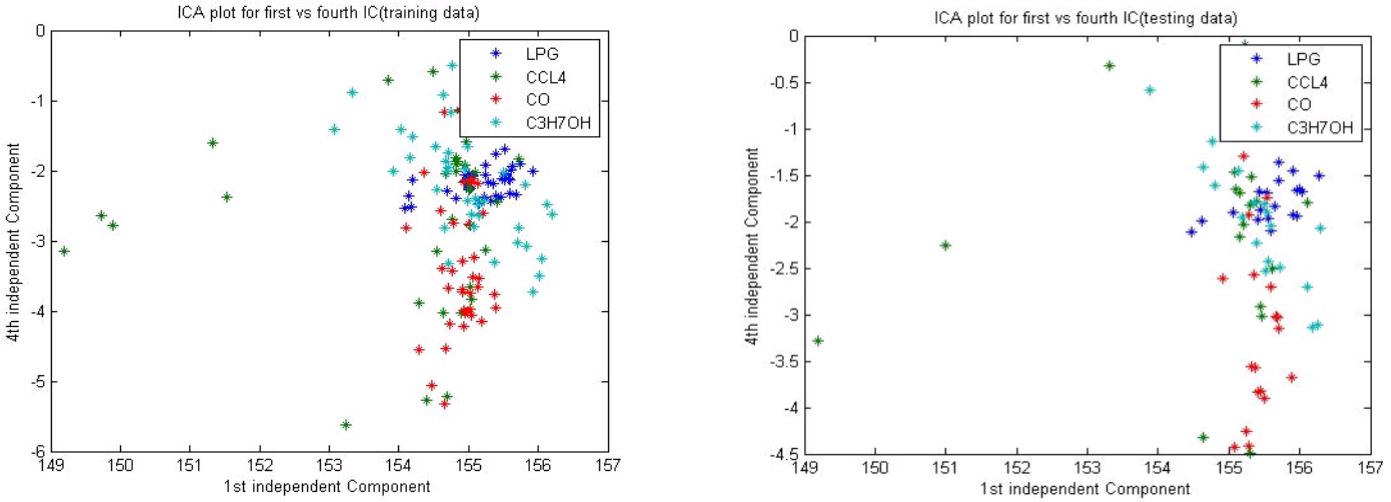
(a)



(b)

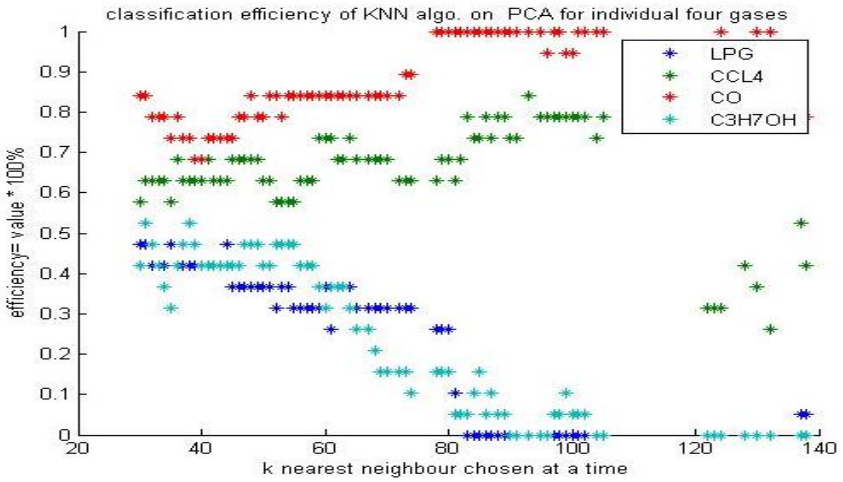
Fig. 3.4 PCA plot for (a) training data (b) testing data

From Fig. 3.6, it can also be concluded that intra-class spread in case of ICA pre-processed data is much less as compared to the PCA pre-processed one. Though none of the above techniques i.e. ICA-KNN and PCA-KNN has been able to completely classify all the data into four classes, the results are promising given the highly jumbled and correlated nature of the raw data. It can be concluded that with some improvements, the ICA-KNN classifier can be used as a popular alternative to the more widely used PCA-KNN classifier.

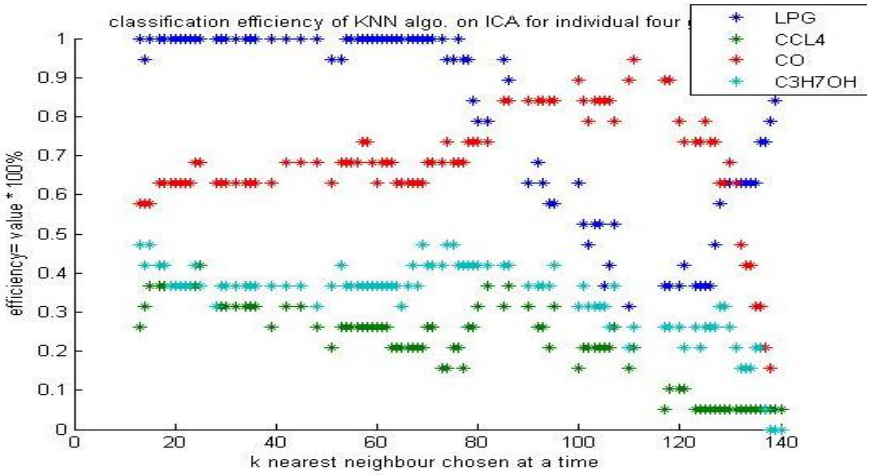


(a) (b)

Fig. 3.5 ICA plots for various for 1st and 4th independent components: (a) Training data, (b) Testing data.



(a)



(b)

Fig. 3.6 Classification efficiency plot for KNN on (a) PCA, (b) ICA for various values of K.

Chapter 4 - Fuzzy *K*NN Classifier for Identification of Individual Gases

4.1 Introduction

Though the application of solid-state sensors is highly dependent upon the pattern classification system used, their ability to sense at room temperature positions them as one of the front runners among all the odour sensing materials available for artificial olfaction systems. As already stated in the previous chapters that this vulnerability arises due to the poorly selective nature of these sensors resulting into high correlation among data samples. As a result crisp classification boundaries are hard to draw and exploration along with exploitation of fuzzy properties in data becomes inevitable. In this work a fuzzy-KNN classifier has been used in identification of four different types of gases using recovery response of a sensor array. Raw data pre-processed with PCA was firstly divided into training and testing sets, normalized and finally fed to the classifier. For each trial mean classification efficiency value has been obtained by taking the average of efficiency values at various values of K and then these values are again averaged by considering hundred such independent trials. Further this whole process has been repeated for different sized training and testing vectors.

Same dynamic response recovery curves of fig.1.2 are used to acquire the sample data. Ideally these curves should have been quite different from each other because each gas behaves in a different manner and possess different physical and chemical properties. However, as already stated in the above paragraph that due to the poorly selective nature of these sensors high correlation is induced among data samples. As a result no clear cut separation exists between the individual classes. Thus data can be assumed to exhibit fuzzy nature and this served as a motivation to apply an appropriate multivariate fuzzy technique to solve the present problem.

4.2 Introduction to Fuzzy Theory

It is a form of knowledge representation suitable for notions that cannot be defined precisely, but which depend upon their contexts. It can also be seen as the generalization

of the conventional crisp set theory. It measures the degree to which an event occurs [44]. Each element of a fuzzy set has a degree of membership assigned to it in accordance with a membership function i.e. it has some degree of presence in each. The most commonly used membership functions in the literature being triangular and trapezoidal membership functions.

The membership function used in the present work is given in equation

$$u_{ij} = \frac{D_{max,i} - D_{ij}}{D_{max,i} - D_{min,i}} \tag{4.1}$$

$$D_{ij} = \left\| \boldsymbol{s_i} - \mathbf{x_j} \right\| \tag{4.2}$$

$\mathbf{x_j}$ is any data sample in the normalized labelled space

Where $\boldsymbol{s_i}$ is the centroid of the i^{th} class data samples

$D_{max,i}$ The maximum Euclidean between mean of $\boldsymbol{s_i}$ and all the samples of i^{th} class

$D_{min,i}$ The minimum Euclidean between $\boldsymbol{s_i}$ and all the samples of i^{th} class

$u_{ij} > 1$ denotes an intruder, set $u_{ij} = 0$ in this case

$u_{ij} < 0$ denotes an outlier , set $u_{ij} = 0$ in this case

4.3 Fuzzy k-Nearest-Neighbor Theory [29]

k-nearest neighbor theory is a very powerful tool for the classification of feature vectors in cases where probability distributions are hard to obtain. However the rule makes an assumption that is all the labelled samples are given equal importance in deciding the degree of membership of an individual vector to be classified, in a particular class. The fuzzy k-nearest-neighbor algorithm assigns class membership to a sample vector rather than assigning a vector to a particular class. Each vector is assigned a membership value in a particular class. The larger this value more are the chances of that particular vector absorbing the given vector.

$$U_i(\mathbf{x}) = \frac{\sum_{j=1}^{K} u_{ij} \left(1/\left\|\mathbf{x} - \mathbf{x_j}\right\|^{2/(m-1)}\right)}{\sum_{j=1}^{K} \left(1/\left\|\mathbf{x} - \mathbf{x_j}\right\|^{2/(m-1)}\right)} \tag{4.3}$$

This function assigns membership based on the fuzzy values obtained using some function based upon the distances (Euclidean in our case). To be precise, the training labelled data samples are assigned membership in each class values based upon some distance criterion. Now all the test vectors are assigned to any particular class based upon these obtained values. The degree of membership of any test vector in a particular class is calculated using a function written in a manner such that the inverse distance serves to weight a vector's membership more if it is closer and less if it is farther from the vector under consideration.

The variable m is the weighing factor and decided the degree of importance when calculating each neighbor's contribution to the membership value. If m is two, the contribution of each neighboring point is weighted by the reciprocal of its distance from the point being classified. With an increase in m, the neighbors are more evenly weighted and their relative distances from the point being classified have less effect. As m approaches one, the closer neighbors are weighted far more heavily than those farther away, which has the effect of reducing the number of points that contribute to the membership value of the point being classified [29].

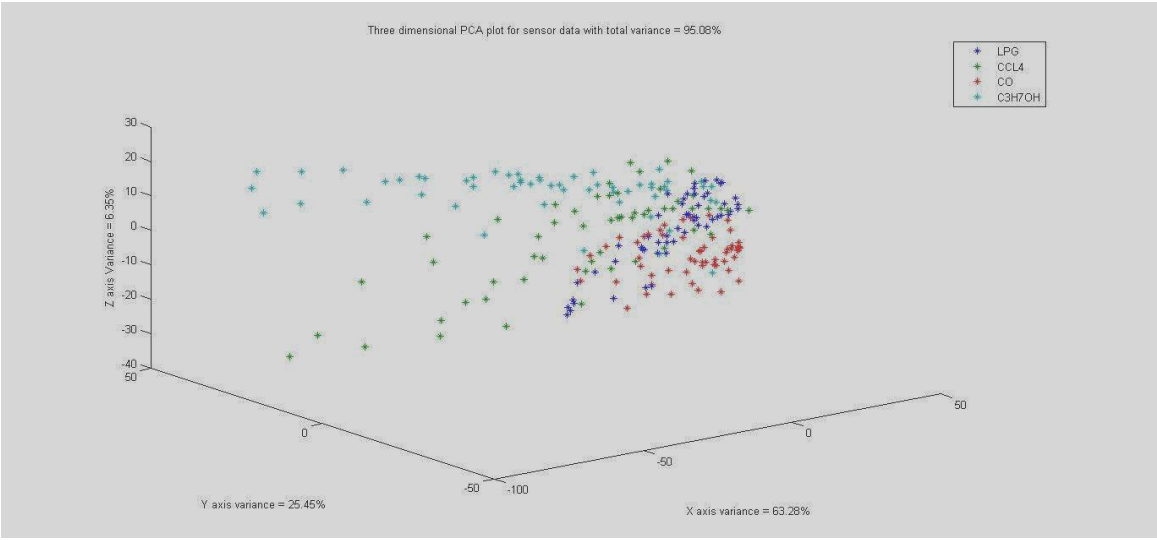


Fig. 4.1 Three dimensional PCA plot for entire data set

4.4 Algorithm [29]

BEGIN

Let **X** be the entire data set in three dimensional space extracted from higher dimensional space using Principal Component Analysis

Set $C =$ cardinality of each class entire data set

Choose the size of the testing vector R

Set $R = C \div 4$

DO UNITIL ($R \leq C \div 2$)

Set $constant$

Initialize z=1

WHILE (z $\leq constant$)

Separate training and testing data by randomly choose equal number of testing data set/unlabelled data set from each class data set having cardinality C

Normalize labelled and unlabelled data set

Assign fuzzy memberships to labelled training samples in each class/gas using equation (4.1)

Input **x** of unknown classification in normalized three dimensional space

Set k, $1 \leq k \leq n$

Initialize i =1.

DO UNITIL (k - nearest neighbors to **x** found)

Compute distance from **x** to $\mathbf{x}_i$.

IF ($i \leq k$) THEN

Include $\mathbf{x}_i$ in the set of k-nearest neighbors

ELSE IF ($\mathbf{x}_i$ closer to **x** than any previous nearest neighbor) THEN

Delete the farthest of the **x**-nearest neighbors

Include $\mathbf{x}_i$ in the set of K-nearest neighbors

END IF

END DO UNTIL

Initialize i=1

DO UNTIL (**x** assigned membership in all classes)

Compute $U_i(\mathbf{x})$ using equation (4.3)

END DO UNTIL

END WHILE

Increment R by 1

END DO UNTILL

END

In this case due to high correlation between data samples along with intra class spread, the application of above algorithm in naïve form may not prove to be fruitful. Some classes have highly sparsely distributed data samples and their distribution is in a very directional manner. As a result when these samples are taken in while calculating the fuzzy values of training samples using distances in omnidirectional space many erroneous values may be encountered. The solution to this is directional fuzziness or normalized distances. Thus in order to have proper fuzzy values the individual class samples need to be normalized in a manner so as to take into picture the spread along any particular direction.

4.5 Simulation Result & Discussion

From the mean efficiency plots a rough idea can be obtained about the maximum and minimum efficiency values for each gas. In the case of CCL_4 these values can lie anywhere in the range 0.6 to 0.9 which is quite high. These values are also in an acceptable limit in the case of CO and C_3H_7OH. However for LPG due to the intrinsic properties of the data performance is degraded and only average results are obtained. From the results it can be concluded that even with such a high inter-cluster correlation along with intra-cluster

sparseness in the data the classification efficiency values obtained using this technique are highly consistent for such large number of permutations.

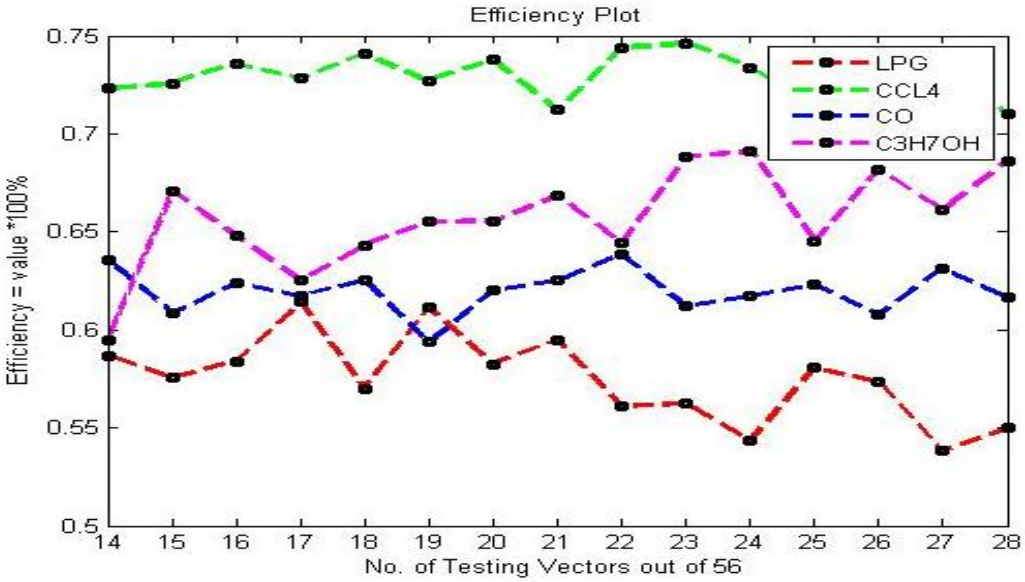


Fig. 4.2: Mean efficiency plot

Chapter 5 - SVM Classifier for Identification of Individual Gases

5.1 Introduction [39]

As already discussed briefly in chapter two the Support Vector Machine is a tool to construct an optimal hyper plane for in a manner so as to maximize the margin of separation. For critical analysis this hyper plane can be classified into two classes. The first one for linearly separable pattern and the second one for non-linearly separable patterns.

Case 1: Optimal Hyper plane for linearly separable patterns

Consider two classes $d1\ and\ d2$ denoting the binary classification problem. The equation is of the form:

$$\boldsymbol{w}^T\mathbf{x} + b = 0 \tag{5.1}$$

Where the used terms are obvious that is $\boldsymbol{w}$ denotes the weight vector, $\mathbf{x}$ feature vector and b be the bias.

The two most basic assumptions or steps which define the essence of the concept of support vector machines are

1. A unique mapping of the input vector based on some criterion so as to transform a low dimensional input vector into a hidden high-dimensional feature space
2. To somehow construct an hyperplane or a surface which could separate the discovered features in this space in the most optimal manner

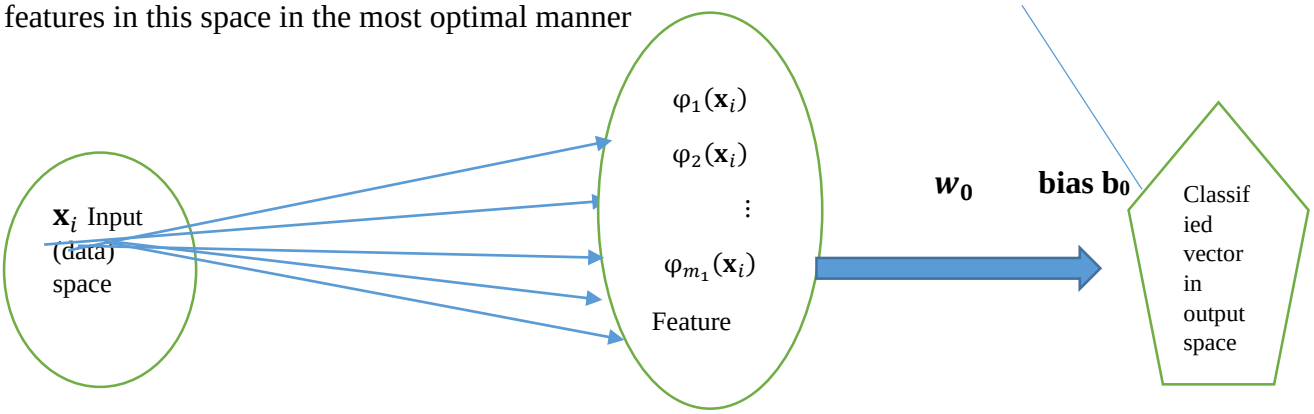


Fig. 5.1: Support Vector Machine principle

Let $\mathbf{x}$ denote a vector drawn from the input space of dimension m

Let $\{\varphi_j(\mathbf{x})\}$ denote a set of nonlinear functions that, between them, transform the input space of dimension m to a feature space of infinite dimensionality.

Given this transformation a hyperplane acting as the decision surface is defined as:

$$\sum_{j=1}^{\infty} w_j \varphi_j(\mathbf{x}) = 0 \tag{5.2}$$

$$\boldsymbol{w}^T \Phi(\mathbf{x}) = 0 \tag{5.3}$$

Where $\Phi(\mathbf{x})$ is the feature vector and w is the corresponding weight vector

For linear-separability of the transformed patterns in the feature space

$$\boldsymbol{w} = \sum_{1=1}^{N_s} \alpha_i \, d_i \Phi(\mathbf{x}_i) \tag{5.4}$$

Where the transformed feature vector is $\Phi(\mathbf{x}_i) = [\varphi_1(\mathbf{x}_i), \varphi_2(\mathbf{x}_i), \ldots \ldots]^T$

And N_s is the number of support vectors. Substituting the above equations we may express the final decision making hyperplane as

$$\sum_{1=1}^{N_s} \alpha_i \, d_i \Phi^T(\mathbf{x}_i) \Phi^T(\mathbf{x}) = 0 \tag{5.5}$$

The scalar term $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x})$ represents an inner product. Let this inner product be

$$k(\mathbf{x}, \mathbf{x}_i) = \Phi^T(\mathbf{x}_i)\Phi^T(\mathbf{x}) \tag{5.6}$$

$$k(\mathbf{x}, \mathbf{x}_i) = \sum_{J=1}^{\infty} \varphi_j(\mathbf{x}_i)\varphi_j(\mathbf{x}), \qquad i = 1,2, \ldots \ldots N_s \tag{5.7}$$

$$\sum_{1=1}^{N_s} \alpha_i \, d_i k(\mathbf{x}, \mathbf{x}_i) = 0 \tag{5.8}$$

Where $k(\mathbf{x}, \mathbf{x}_i)$ is called the inner product kernel

5.2 Design of Support Vector Machines [39]

The expansion of the kernel $k(\mathbf{x}, \mathbf{x_i})$ gives us to construct a decision surface that is non-linear in the input space, but whose image in the feature space is linear. The form for obtaining the optimized surface is:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, d_i d_j \, k(\mathbf{x_i}, \mathbf{x_j}) \tag{5.9}$$

Subject to the constraints

1. $\sum_{i=1}^{N} \alpha_i d_i = 0$

2. $0 \leq \alpha_i \leq C$ ..... for $i = 1,2, \ldots \ldots N$

Where C is a user specified positive parameter

The three most common types of kernels

Type of support vector machine	Mercer Kernel $k(\mathbf{x}, \mathbf{x_i})$, $i = 1,2 \ldots \ldots N$	Comments
1.Polynomial Learning Machine	$(\mathbf{x}^T \mathbf{x}_i + 1)^P$	Power p is user defined
2.Radial-basis-function network	$e^{\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}_i\|^2\right)}$	The width σ^2, common to all the kernels, is user defined
3.Two-layer perceptron	$\tanh(\beta_0 \mathbf{x}^T \mathbf{x_i} + \beta_1)$	Valid only for certain value of β_0 and β_1

Table 5.1**:** Types of Support Vector Machine [39]

Out of these the Radial-Basis-function network is the most generalized support vector machine. In the present work this type of RBF network along with the polynomial one has been adopted as the area of discussion

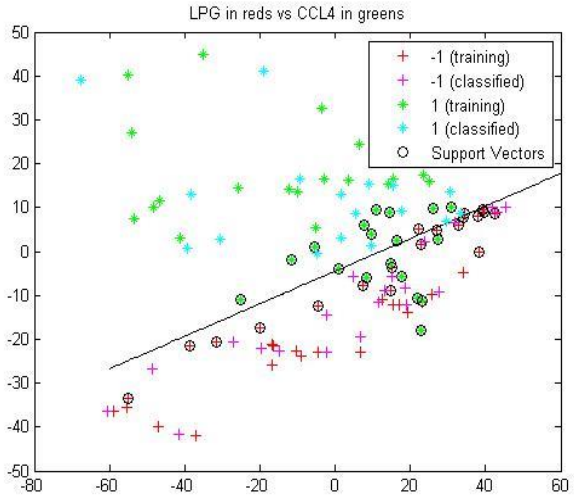
Some of its salient features of RBF networks are [39]:

1. They always satisfy the Mercer's theorem which is a basic condition for any kernel to be a SVM

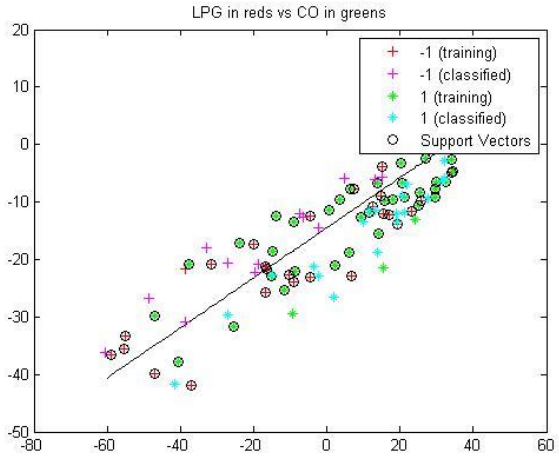
2. Like the other two types of machines the support vectors are extracted from the training data. Correspondingly the dimensionality of the feature space is determined by these number of extracted support vectors.

3. The number of support vectors and their values determine the number of radial basis functions along with their centres.

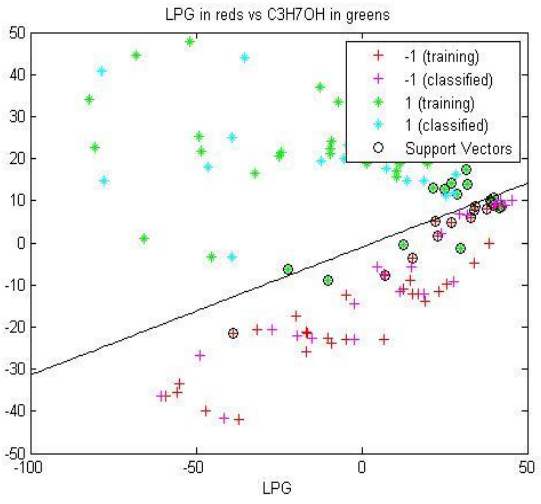
5.3 Simulation Results & Discussion



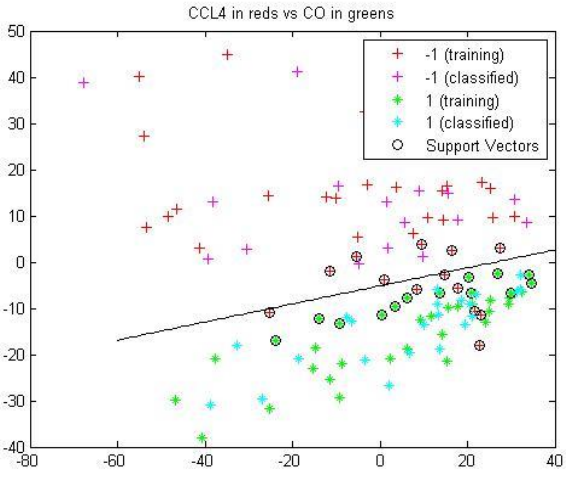
(a)



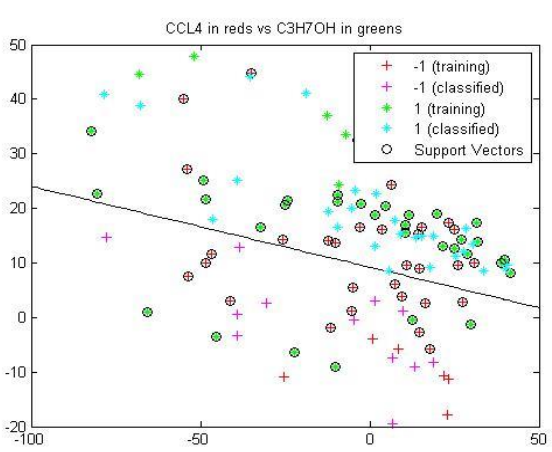
(b)



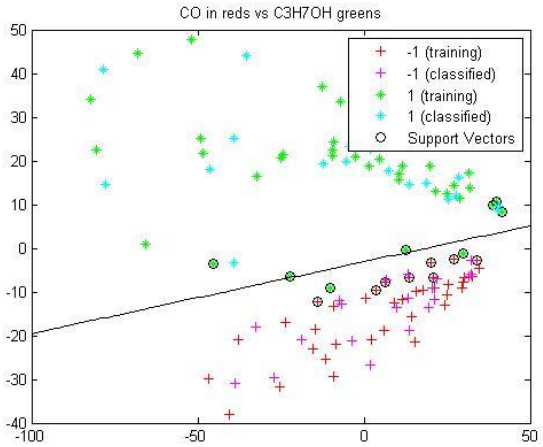
(c)



(d)



(e)



(f)

Fig. 5.2 Support Vector Machine with polynomial Kernel

The above figure show support vector machine constructed using polynomial kernel. The efficiency of classification increases with increasing value of p in the kernel. Thus increasing the value increases the efficiency but price is kernel being more complex.

Using the majority voting scheme each sample was classified in one of the class. The following table shows the results

Number of classes/gases = 4 (LPG, CCL_4, CO and C_3H_7OH)

Number of samples in each class = 56

Number of training samples in each class = 37

Number of testing samples in each class =19

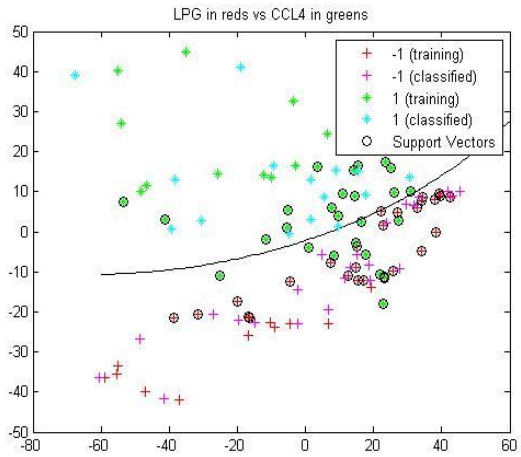
Pre-processor used = PCA

Number of samples of LPG classified as LPG = 19

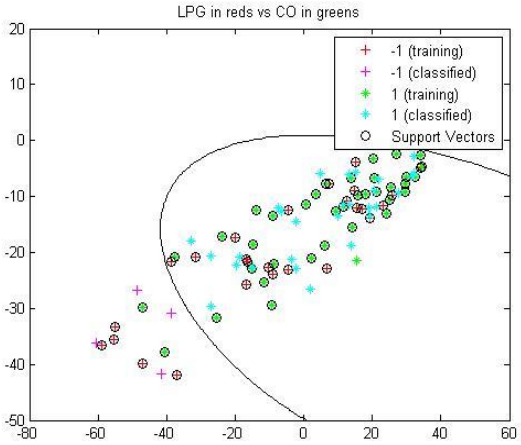
Number of samples of CCL_4 correctly classified =16

Number of samples of CO correctly classified =19

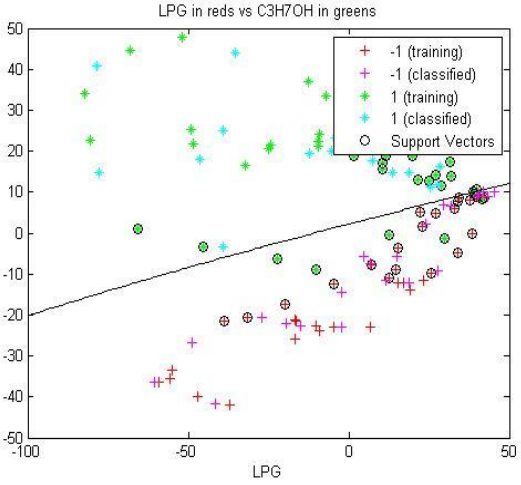
Number of samples of C_3H_7OH correctly classified =18



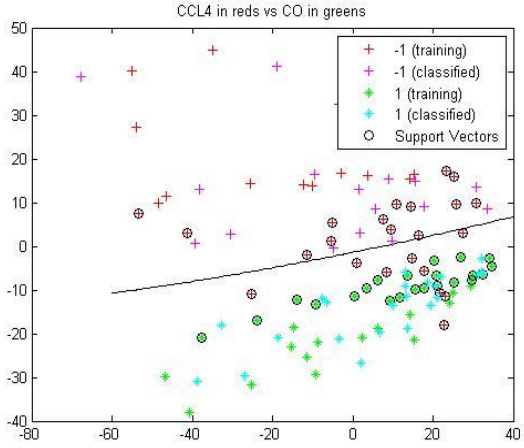
(a)



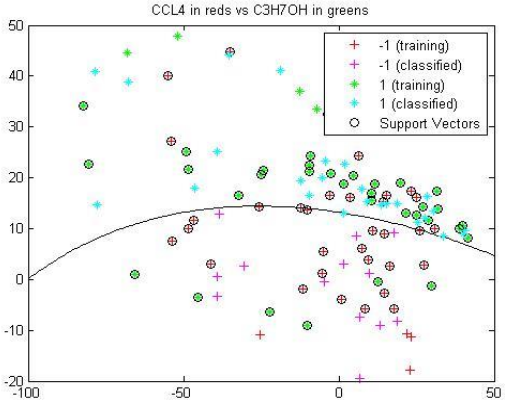
(b)



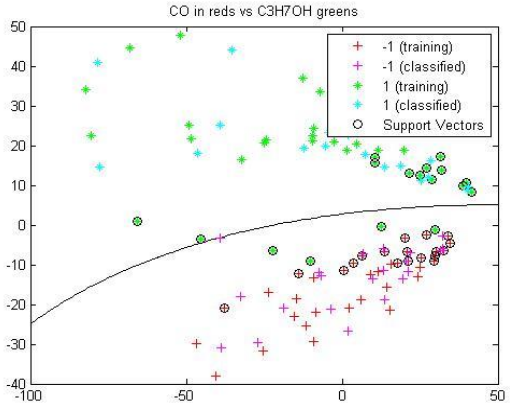
(c)



(d)



(e)



(f)

Fig 5.3: Decision hyperplane for combination of gas with data pre-processed with PCA fed to SVM with RBF kernel

Using the majority voting scheme each sample was classified in one of the class. The following table shows the results

Number of classes/gases = 4 (LPG, CCL_4, CO and C_3H_7OH)

Number of samples in each class = 56

Number of training samples in each class = 37

Number of testing samples in each class =19

Pre-processor used = PCA

Number of samples of LPG classified as LPG = 19

Number of samples of CCL_4 correctly classified =17

Number of samples of CO correctly classified =19

Number of samples of C_3H_7OH correctly classified =17

In the figures presented above the application of kernel technique i.e. Support Vector Machine is presented. With an increase in dimensionality the complexity increases exponentially as a result with increasing dimensions the performance efficiency may increase but the design of such a machine is quite a challenging task. Thus raw training data pre-processed with PCA has been fed to the network. Bases upon the data the machine has been trained and the decision hyperplane has been constructed in each case. It is easy to visualize that both the SVM are able to efficiently classify while RBF one is marginally better as compared to the linear straight line case. With RBF being more general and more efficient, this difference increases with more difficult problems.

Chapter 6- Eigenfiltering Using Generalized Hebbian Algorithm

6.1 Introduction

The Generalized Hebbian Algorithm is one of its kind because of the simple but unique features it possess. It is a neural network approach to extract principal components. The neural model act as an eigenfilter for the input data i.e. each output neuron in the network extracts and filter out the principal component from the input sample space.

There are two unique aspect of this type of this algorithm:

1. Any network of neurons with random initial weights and following the given rule will converge to and act as the desired filter with a probability equal to one as the number of iterations approaches infinity [39]

2. Second unique feature lies in the fact that this algorithm is a much better choice for extracting the principal components as compared to the one discussed in chapter 3 from hardware implementation point of view. This is evident from the fact that the other method i.e. PCA using correlation firstly computes a matrix to obtain the correlation structure of the input data. Which is quite computationally expensive as it requires large number of multiplications and additions and this complexity increases in a manner proportional to the number of samples and their dimensionality

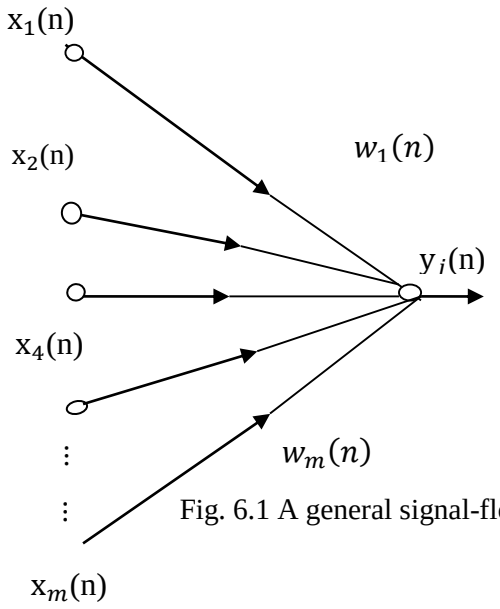


Fig. 6.1 A general signal-flow graph

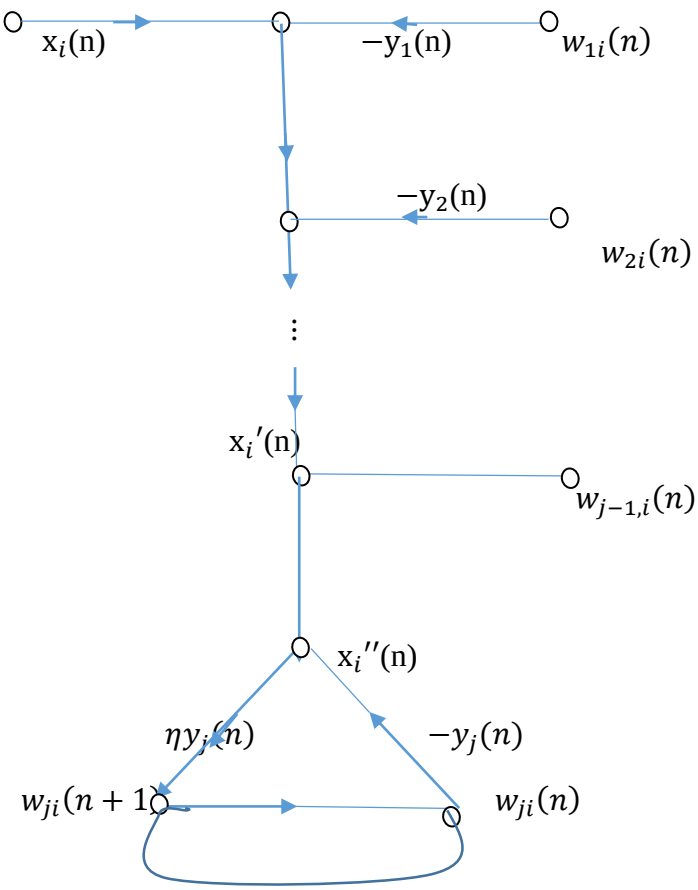


Fig. 6.2 Signal-flow graph for GHA

Looking closely at the equation we can comment that

$$\Delta w_{ji}(n) = \eta y_j(n) x_i''(n) \tag{6.1}$$

$$\text{Where} \quad x_i''(n) = x_i'(n) - w_{ji}(n) y_j(n) \tag{6.2}$$

From graph it can be noted that

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{6.3}$$

$$w_{ji}(n) = z^{-1}\left[w_{ji}(n+1)\right] \tag{6.4}$$

Re-writing in another form

$$\Delta w_{ji}(n) = \eta y_j(n)\, \mathbf{x}'(n) - \eta y_j(n)^2 w_j(n) \ldots\ldots\ldots\ldots\ldots\ldots j = 1,2 \ldots \ldots l \tag{6.5}$$

$$x'(n) = x(n) - \sum_{k=1}^{j-1} w_k(n) y_k(n) \tag{6.6}$$

6.2 Algorithm [39]

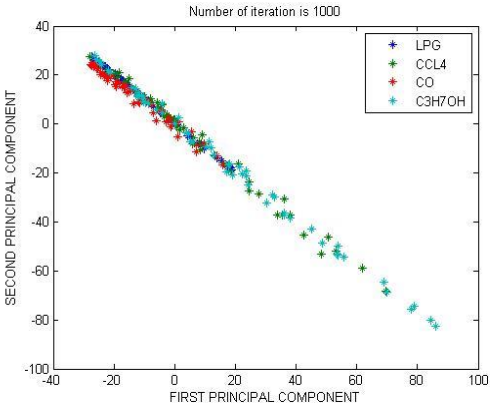
1. Initialize the synaptic weights of the network, w_{ji} and learning rate parameter η randomly with any small value.

2. Initialize $n = 1$ & implement the equations followed be incrementing n by 1

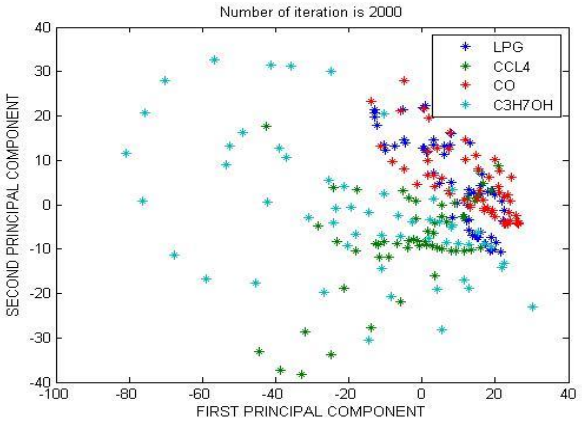
$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n)\, \mathrm{x}_i(n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots j = 1,2 \ldots \ldots . l \qquad (6.7)$$

$$\Delta w_{ji}(n) = \eta \left(y_j(n)\mathrm{x}_i(n) - y_j(n) \sum_{k=1}^{j} w_{ki}(n)\, y_k(n) \right), j = 1,2 \ldots l\,, i = 1,2 \ldots m \qquad (6.8)$$

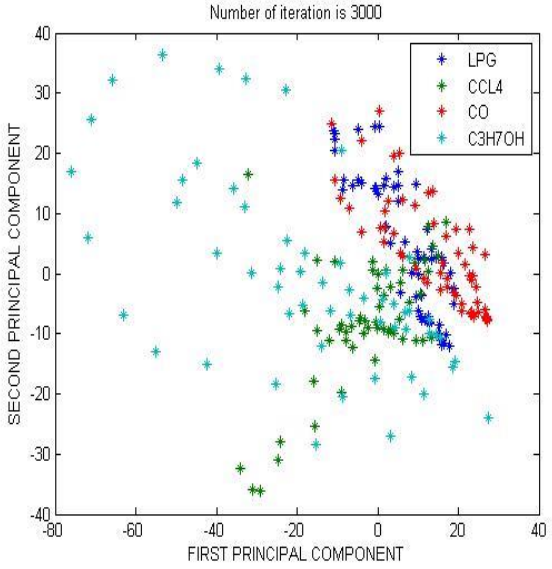
6.3 Simulation Results & Discussion



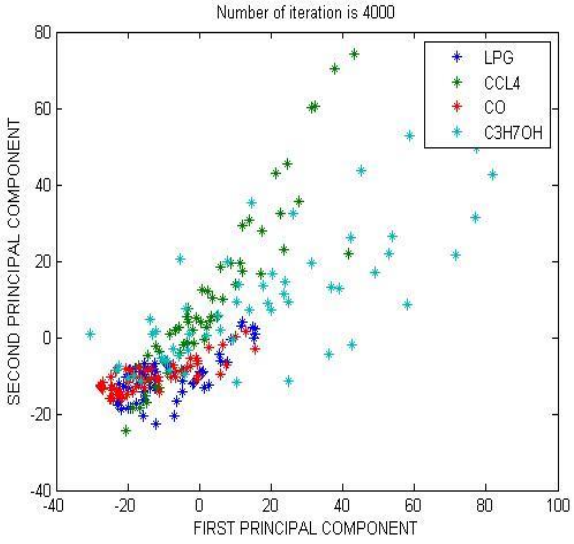
(a)



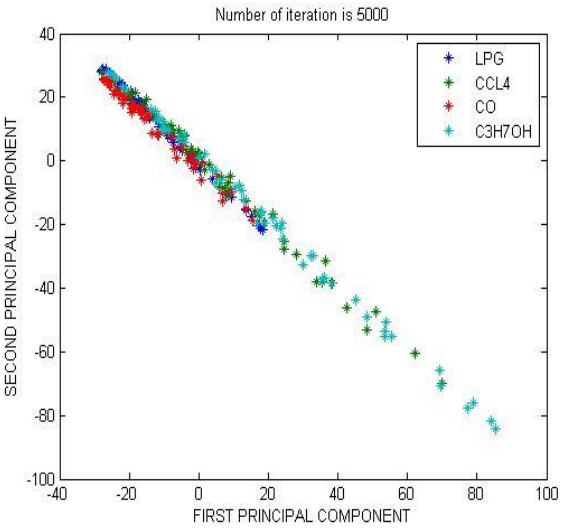
(b)



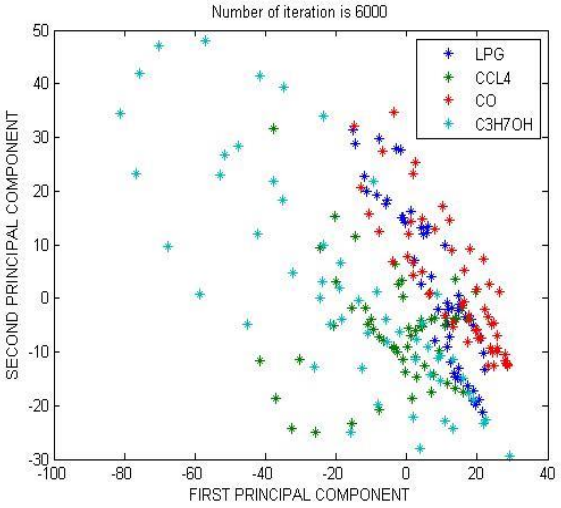
(c)



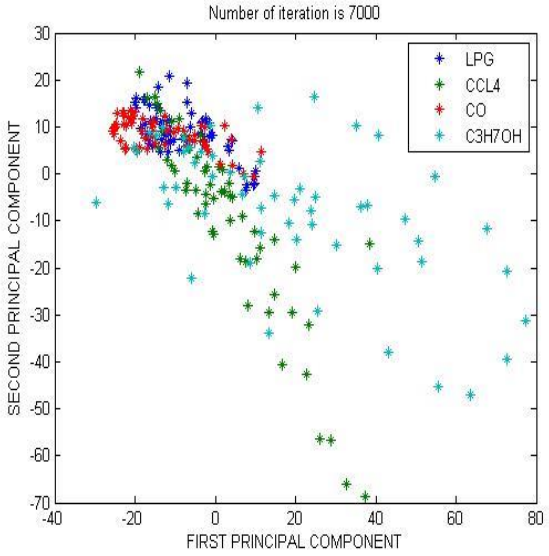
(d)


Number of iteration is 5000
LPG
CCL4
CO
C3H7OH
SECOND PRINCIPAL COMPONENT
FIRST PRINCIPAL COMPONENT

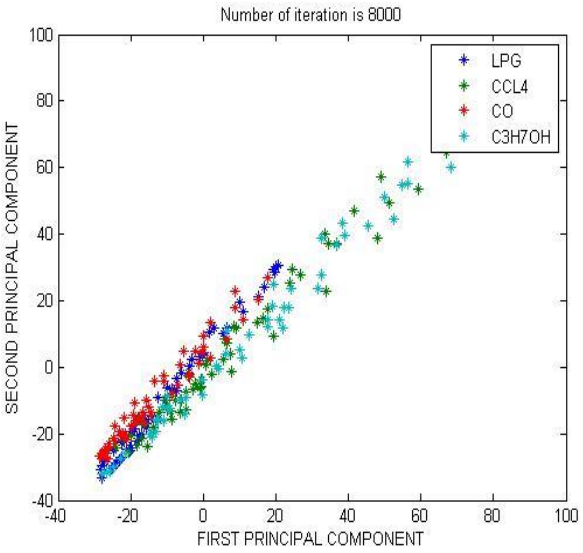
(e)


Number of iteration is 6000
LPG
CCL4
CO
C3H7OH
SECOND PRINCIPAL COMPONENT
FIRST PRINCIPAL COMPONENT

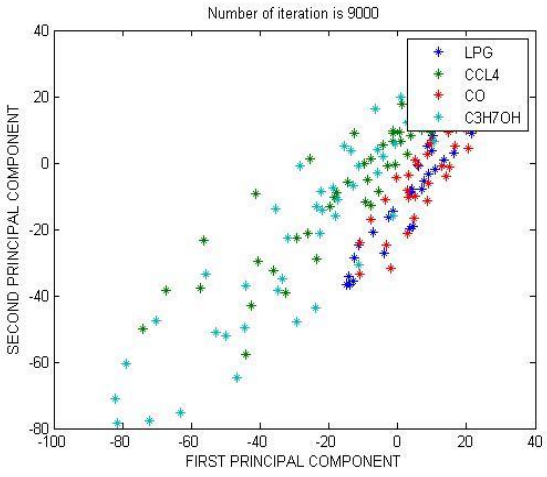
(f)


Number of iteration is 7000
LPG
CCL4
CO
C3H7OH
SECOND PRINCIPAL COMPONENT
FIRST PRINCIPAL COMPONENT

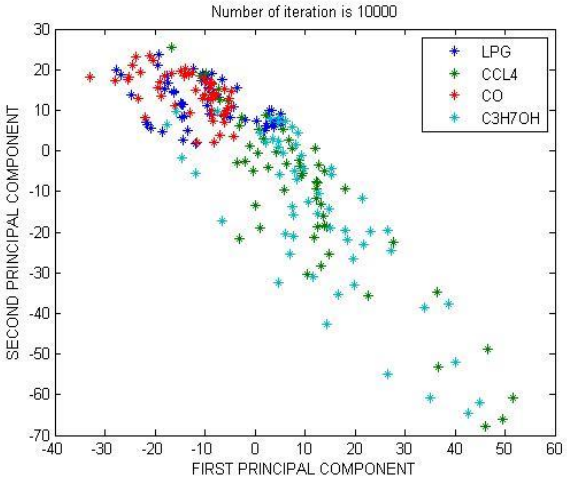
(g)


Number of iteration is 8000
LPG
CCL4
CO
C3H7OH
SECOND PRINCIPAL COMPONENT
FIRST PRINCIPAL COMPONENT

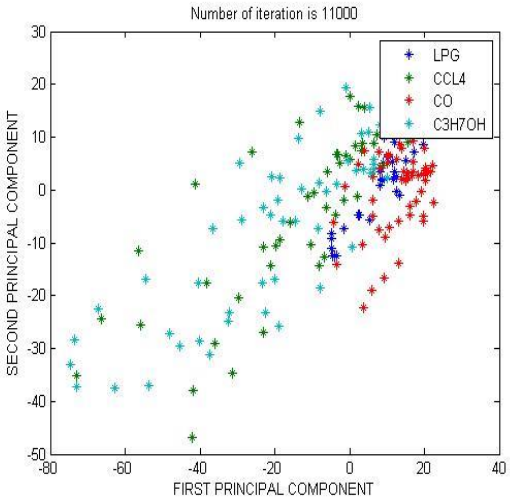
(h)


Number of iteration is 9000
LPG
CCL4
CO
C3H7OH
SECOND PRINCIPAL COMPONENT
FIRST PRINCIPAL COMPONENT

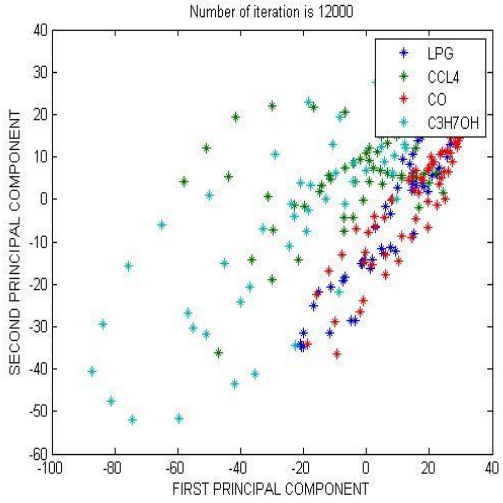
(i)


Number of iteration is 10000
LPG
CCL4
CO
C3H7OH
SECOND PRINCIPAL COMPONENT
FIRST PRINCIPAL COMPONENT

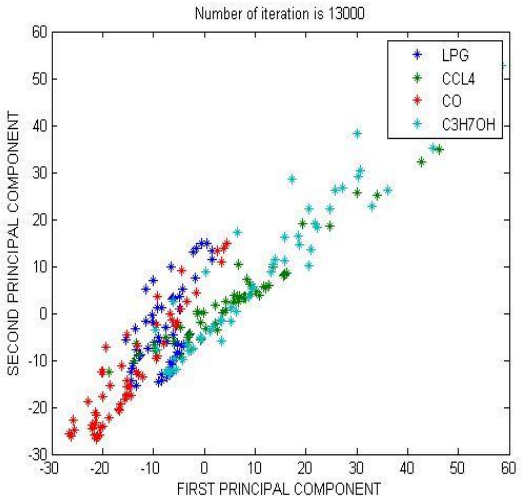
(j)



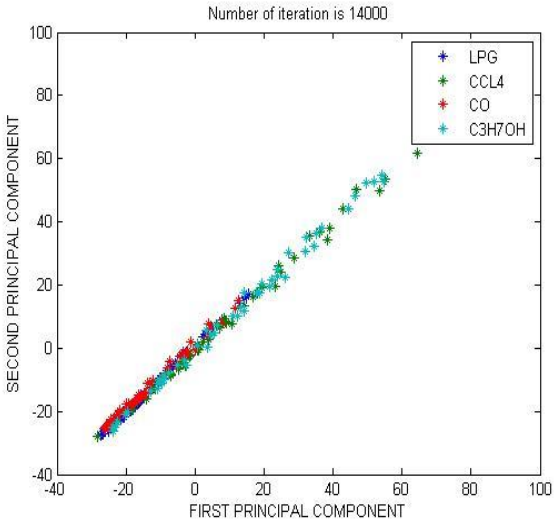
(k)



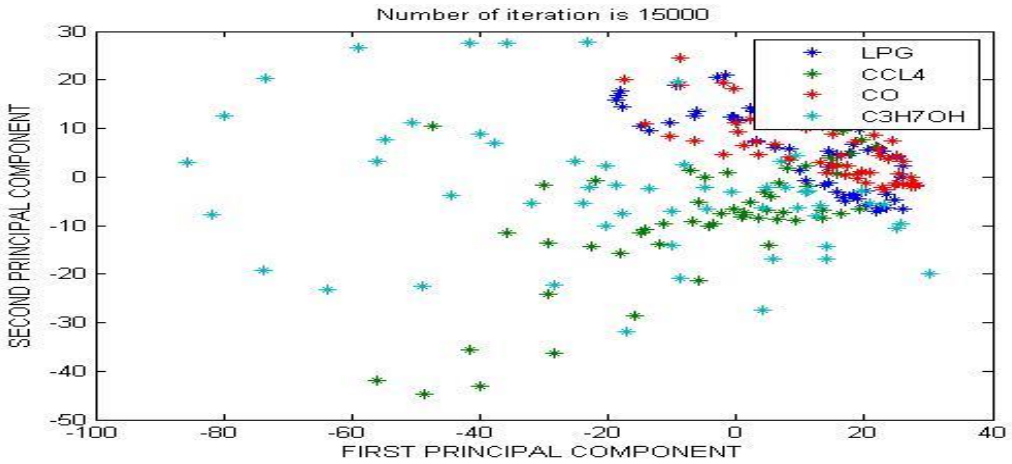
(l)



(m)



(n)



(o)

Fig. 6.3 (a)-(o): Resulting PCA plots at each iteration

The above PCA plots show the resulting plot at each iteration from 1000 to 15000.
With increasing number of iterations the plot converges to the one in chapter 3 i.e PCA using correlation matrix. This algorithm is a better choice from hardware implementation point of view because of large number additions and multiplications involved in the first method while calculating the correlation structure.

Chap7 *Self-Organizing Map for Identification of Individual Gases*

7.1 Introduction [39]

Self-Organizing Map is a type of neural network having special characteristics of its own that no other network possess. Among all the ANN networks it is the only algorithm which tries to depict the inherent spatial features of any distribution and organizes them in a coherent manner. The map resembles a topographically organized map found in the cortices of the more developed animal brains i.e. in an animal brain different parts respond to different types of input be it visual, auditory etc. These inputs provides excitement in that region only. Thus the brain can be seen as a summation of non-linear and localized self-organizing structures.

After fine tuning of its weight vectors, the Self-Organizing Map has been particularly successful in various pattern recognition tasks involving very noisy signals. These networks are based on competitive learning; the output neurons of the network compete among themselves to be activated or fired, with the result that only one output neuron, or one neuron per group, is on at any one time. An output neuron that wins the competition is called winner-takes-all neuron or simply a winning neuron

In a self-organizing map, the neurons are placed at the nodes of a structure or a lattice that is generally low dimensional in space (generally two dimensional but there is no constraint on this). The neurons become selectively tuned to various input patterns (stimuli) or classes of input patterns of a competitive-learning process. In a self-organizing map the location of neurons changes every time an input comes to it as a result after tuning it becomes ordered in a sense that can be considered as a visualization of input space feature lattice. A self-organizing map is therefore characterized by the formation of a tomographic map of the input patterns, in which the spatial locations i.e. co-ordinates of the neurons in the lattice are indicative on intrinsic statistical features contained in the input patterns.

The principal goal of Self-Organizing Map is to transform an incoming signal pattern of arbitrary dimension into a one- or two- dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion. Thus it can also be viewed as a mode of Principal Component Analysis.

7.2 Self- Organizing Map: Algorithm [39]

The algorithm starts by initializing the synaptic weights in the network. Any random small but different value can be picked randomly as initial synaptic weight vector. After initialization there are three important processes that follow are:

Competition: For each input pattern, the neurons in the network need to compete among themselves and compute their respective values of a discriminant function. The winner is declared according to this function that is the neuron with the largest value of discriminant function is declared winner of the competition.

Cooperation: These neurons after competition tend to cooperate with each other. The winner assumes the position of most cooperative member. The winning neuron determines the spatial location of a topological neighborhood of excited neurons so as to initiate cooperation

Synaptic Adaptation: The excited neurons tend to increase their individual values of the discriminant function in accordance to the input pattern through suitable synaptic weight adjustments. The adjustments made are such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced. Neurons closer to the winner are able to adjust more as compared to the distant ones.

7.2.1 Competitive Process

Let us consider an input pattern (vector) selected at random from the input space in a m dimensional Euclidean feature space denoted by

$$\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \dots \dots \dots \mathbf{x_m}]^T \tag{7.1}$$

The synaptic-weight vector of each neuron in the network has the same dimension m as the input feature space denoted by

$$\boldsymbol{w_j} = \left[\boldsymbol{w_{j1}}, \boldsymbol{w_{j2}} \dots \dots \dots \boldsymbol{w_{jm}}\right]^{\boldsymbol{T}}, \boldsymbol{j} = \mathbf{1}, \mathbf{2}, \mathbf{3} \dots \dots \boldsymbol{l} \tag{7.2}$$

Here l is the number of neurons in the network. The best match of the input vector $\mathbf{x}$ with the synaptic weight vector $\boldsymbol{w}$ can be found by comparing the inner product $\boldsymbol{w_j}^T\mathbf{x}$ for $all\ \boldsymbol{j} = \mathbf{1}, \mathbf{2}, \mathbf{3} \dots \dots \boldsymbol{l}$ and selecting the largest. For fair competition same threshold is applied to all the neurons; the threshold is the negative of the bias. Thus, by the selection of the neuron

with the largest inner product $\boldsymbol{w_j}^T\mathbf{x}$, we could easily determine the centre of the topological neighborhood of excited neurons.

Mathematically the criterion for finding the winner or the most suitable candidate is equivalent to minimizing the Euclidean distance between the vectors $\mathbf{x}$ and $\boldsymbol{w_j}$subject to the constraint that $\boldsymbol{w_j}$ has unit length for all $\boldsymbol{j}$. Let $i(\mathbf{x})$ denotes the function which identify the best matching neuron to the input vector $\mathbf{x}$

$$i(\mathbf{x}) = \arg\min\|\mathbf{x} - \boldsymbol{w_j}\| \quad \boldsymbol{j} = \mathbf{1}, \mathbf{2}, \mathbf{3} \ldots \ldots \boldsymbol{l} \tag{7.3}$$

The particular neuron i that satisfies this condition is called the best-matching or winning neuron for the input vector $\mathbf{x}$ it is this neuron which will be the centre of the neighborhood. Because of the fact that the number of neurons are finite and many input patterns may get mapped to the same neuron, the above mentioned process can also be considered as a mapping from a continuous input space of patterns onto a discrete output space of neurons by a process of most suitable candidate selection through competition.

7.2.2 Cooperative Process

The winning neuron is chosen as the center of a topological neighborhood of excited neurons. These neurons cooperate among themselves to adjust their respective weights. Choice of a topological neighborhood is user dependent and is quite an important task. In particular, neurons closer to the firing neuron get excited more than those farther away from it. Thus a topological neighborhood smoothly decaying with lateral distance is suffice to perform the given task

Let $h_{j,i}$ denote the topological neighborhood with winner as the center denoted by i and surrounded by a set of excited (cooperating) neurons j with $d_{j,i}$ denoting the lateral distance between them.

Being translation invariant (i.e., independent of the location of the winning neuron, i) a Gaussian neighborhood is a good choice as it is symmetric about the maximum point defined by $d_{j,i} = 0$; and it attains its maximum value at the winning neuron i for which the distance is zero along with its amplitude monotonically decreasing with increasing lateral distance $d_{j,i}$, decaying to zero for $d_{j,i}$ tending to infinity, intuitively denoting weaker relationship between distant ones.

A typical example choice for this function is:

$$h_{j,i}(\mathbf{x}) = \boldsymbol{e}^{\left(-d_{j,i}^{2}/2\sigma^{2}\right)},\ \boldsymbol{j = 1, 2, 3 \ldots . l} \tag{7.4}$$

The parameter σ is the "effective width" of the topological neighborhood. It measures the degree to which excited neurons in the vicinity of the winning neuron participate in the learning process.

Another important aspect of the SOM is the continuous but slowly shrinking nature of the topological neighborhood with time. With a small shrinking rate the neighborhood is allowed to change so that lesser neurons are excited in the process. σ_0 is the value of σ at the initiation of the SOM algorithm and τ_1 is a time constant to be chosen by the designer. Thus the above equation can be re-written as

$$h_{j,i(\mathbf{x})} = \boldsymbol{e}^{\left(-d_{j,i}^{2}/2\sigma(\boldsymbol{n})^{2}\right)},\ \boldsymbol{j = 1, 2, 3 \ldots . l} \tag{7.5}$$

Discrete time step n denoting the number of iterations, the width σ (n) decreases at an exponential rate with increasing n and thus the topological neighborhood shrinks correspondingly.

7.2.3 Adaptive Process

Hebb's postulate in its basic form results in saturation. Reason being synaptic weight increment in one direction only. This increment occurs whenever there is simultaneous occurrence of pre-synaptic and post-synaptic activities around a neuron. To overcome this problem forgetting term given by term $g(y_j)w_j$ has been included in the Hebbian hypothesis. This factor can also be visualized in practical conditions. Here w_j is the synaptic weight vector of neuron j and the other term is some positive scalar function of the response y_j. For logical reasons function $g(y_j)$ should satisfy one condition i.e. the constant term in the Taylor series expansion of be zero i.e. is $g(y_j) = 0$ for y_j = 0 i.e. no forgetting in case of no learning.

$$\Delta \boldsymbol{w_j} = \boldsymbol{\eta} h_{j,i(\mathbf{x})}(\mathbf{x} - \boldsymbol{w_j}) \tag{7.6}$$

$$\boldsymbol{w_j}(n+1) = \boldsymbol{w_j}(n) + \eta(n) h_{j,i(\mathbf{x})}(n)(\mathbf{x}(\mathrm{n}) - \boldsymbol{w_j}(n)) \tag{7.7}$$

Equation 7.7 is the SOM in form here i is the winning neuron and j is the excited neuronThus the final equation has the effect of moving the synaptic weight w_i of winning neuron i towards the input vector **x.**

Upon repeated presentations of the training data, the synaptic weight vectors tend to follow the distribution of the input vectors. Along with all the features the learning rate parameter should also be time varying. In general an exponential decay function for this learning rate parameter is a feasible option

7.2.3.1 Phases of the adaptive process

There are basically two phases of the adaptive process

Self Organizing or ordering phase: It is the topological weight ordering phase of the adaptive process. The ordering phase may take as many as 1,000 iterations of the SOM algorithm, and possibly more. During this phase the learning rate parameter and neighborhood function described here:

1. The learning rate parameter η(n***)*** should begin with a value close to 0.1; there after it should decrease gradually, but remain above 0.01(it should never be allowed to reduce to zero).

2. The neighborhood function should initially include all neurons in the network centered on the winning neuron I and then shrink slowly with time

Convergence Phase: It is the fine tuning phase of the adaptive process providing an accurate statistical representation of the input space: The number of iterations needed for convergence depends strongly on the dimensionality of the input space. As a general rule, the number of iterations must be at least 500 times the number of neurons in the network. For good output results the learning parameter η(n) should be maintained at a very small value and must not be allowed to decrease to zero. The neighborhood function should only contain the nearest neighbor of a winning neuron.

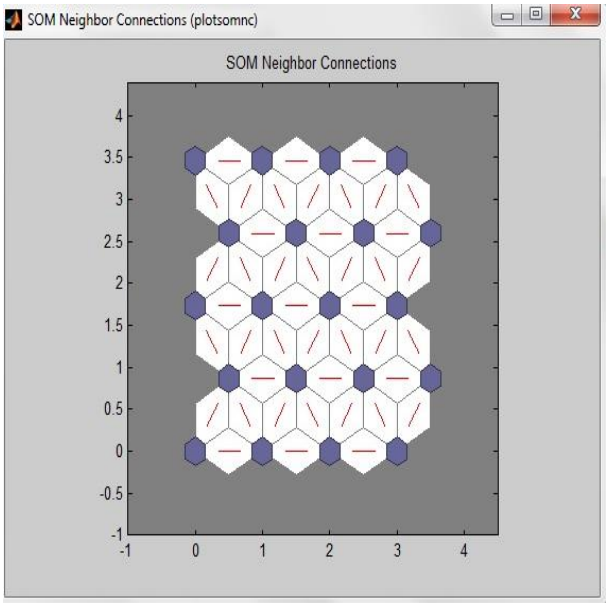
7.3 Simulation Results & Discussion

A self-organizing map for raw input data has been created with number of iterations set to 10000 initial neighborhood size equal to 3 and a neural configuration of 4*5

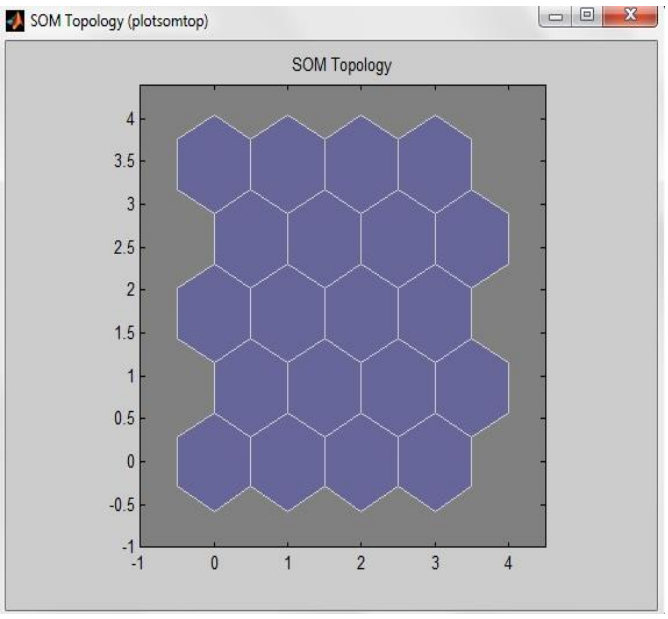
Figure 7.1(a) and (b): Show the neighbour connections and SOM topology. In the present case hexagonal structure has been chosen.[45]

Figure 7.1(c): This figure uses a particular color coding : 1) The blue hexagons represent the neurons. 2) The red lines connect neighboring neurons. 3) The colors in the regions containing the red lines indicate the distances between neurons. 4) The darker colors represent larger distances. 5) The lighter colors represent smaller distances.[45]

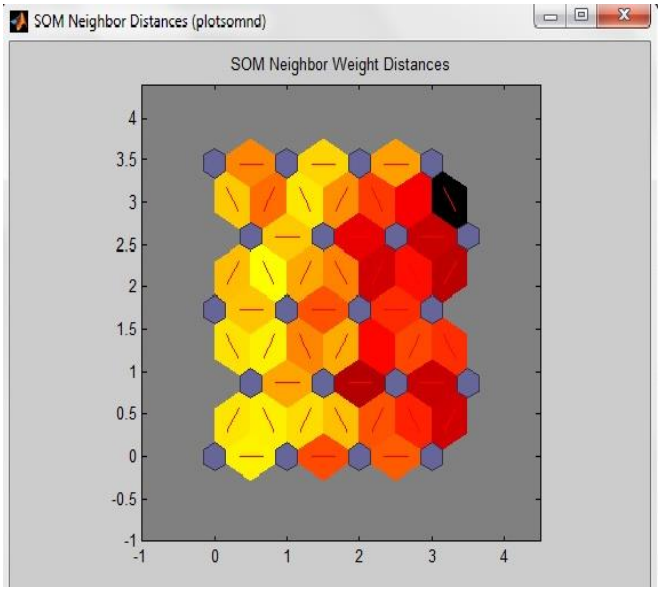
Figure 7.1 (d): This figure shows the weight plane. There is a weight plane for each element of the input vector (five, in this case). They are visualizations of the weights that connect each input to each of the neurons. (Darker colors represent larger weights.) If the connection patterns of the inputs are very similar, then it denotes that the inputs are highly correlated [45]



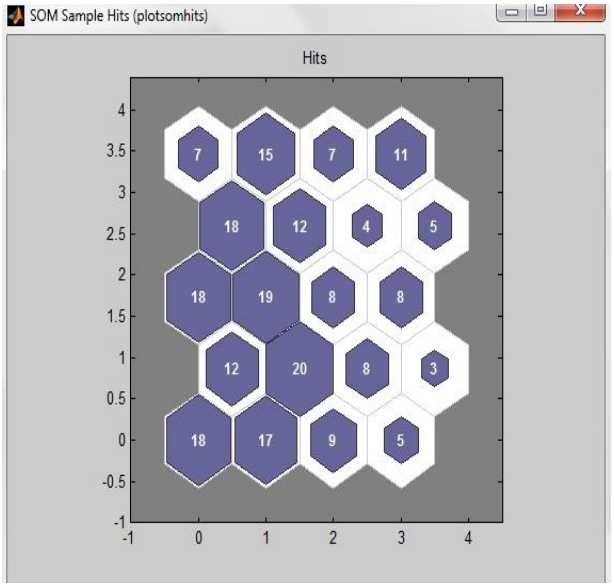
(a)



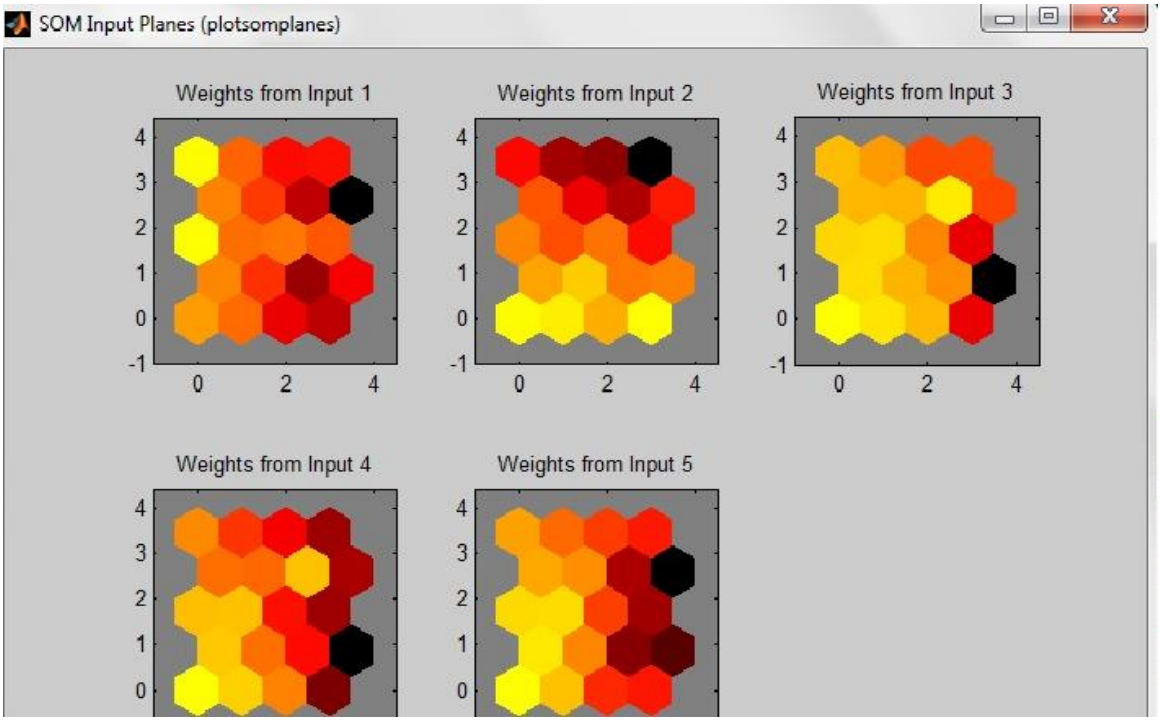
(b)



(c)

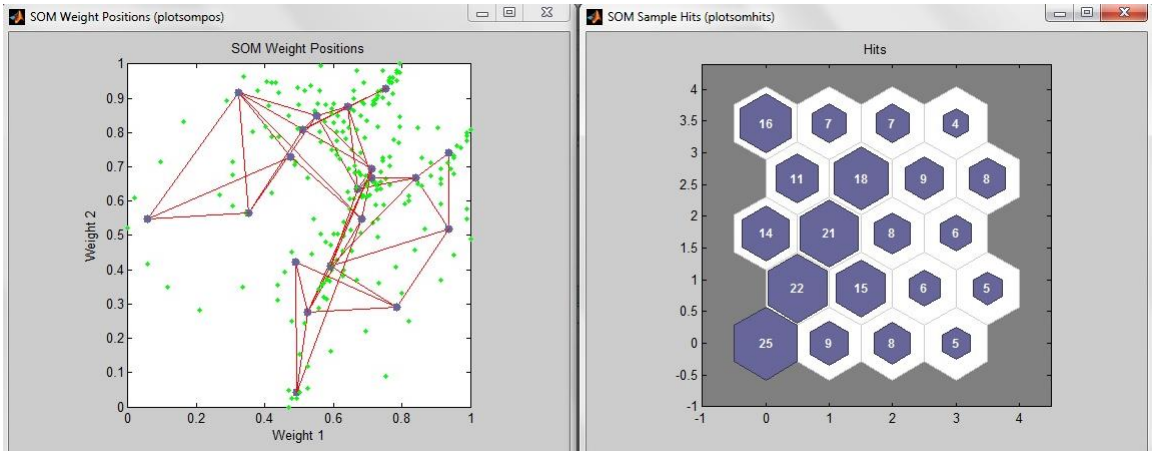


(d)

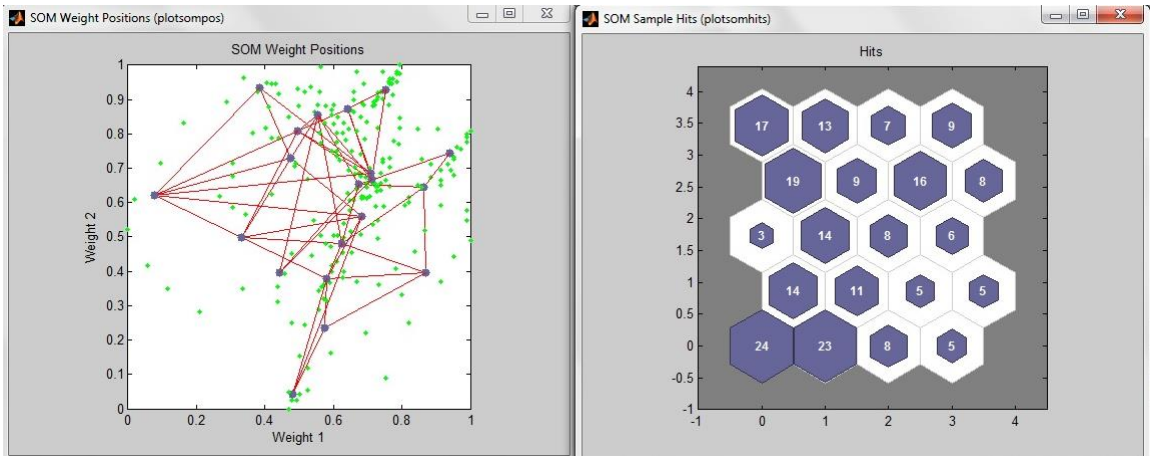


(e)

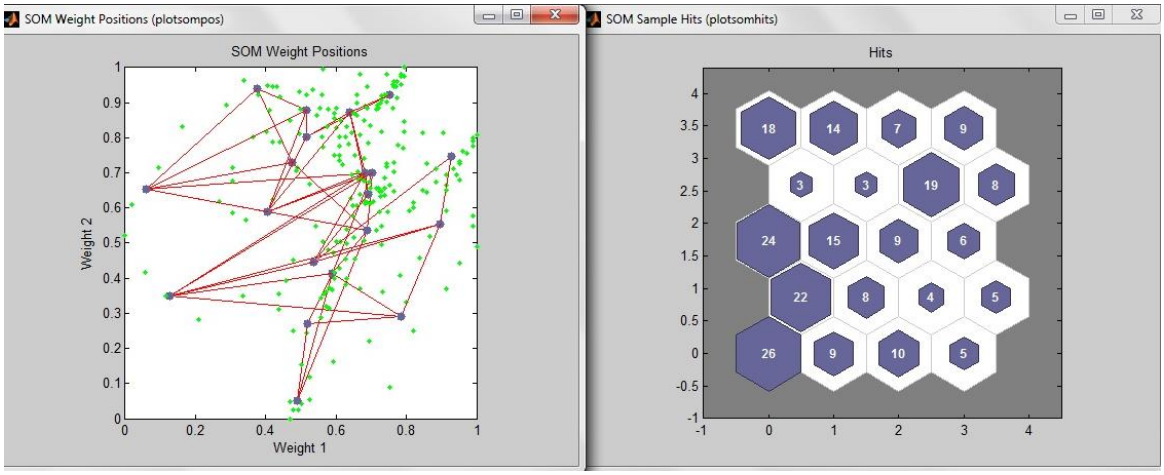
Fig. 7.1(a)-(d): SOM representation in MATLAB



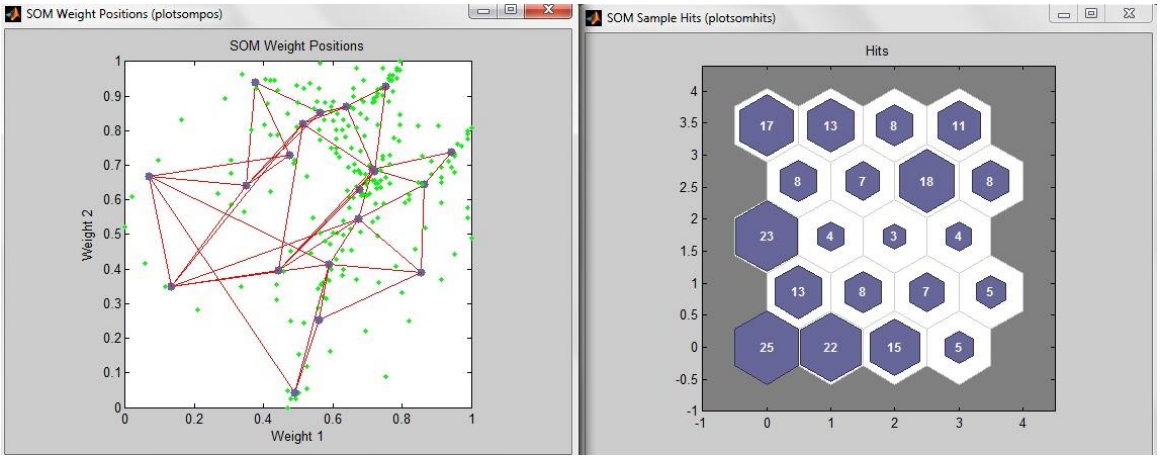
(a) SOM at 200 iterations



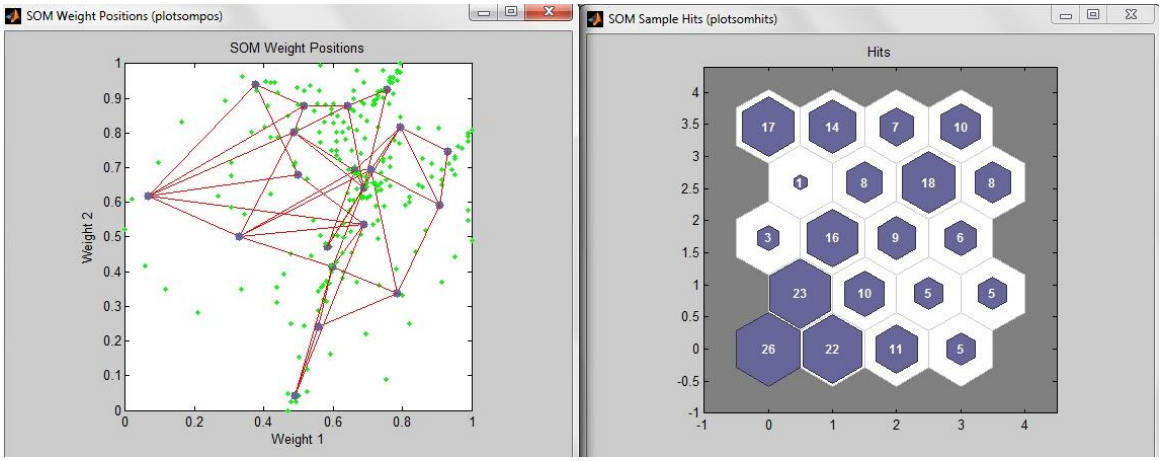
(b) SOM at 500 iterations



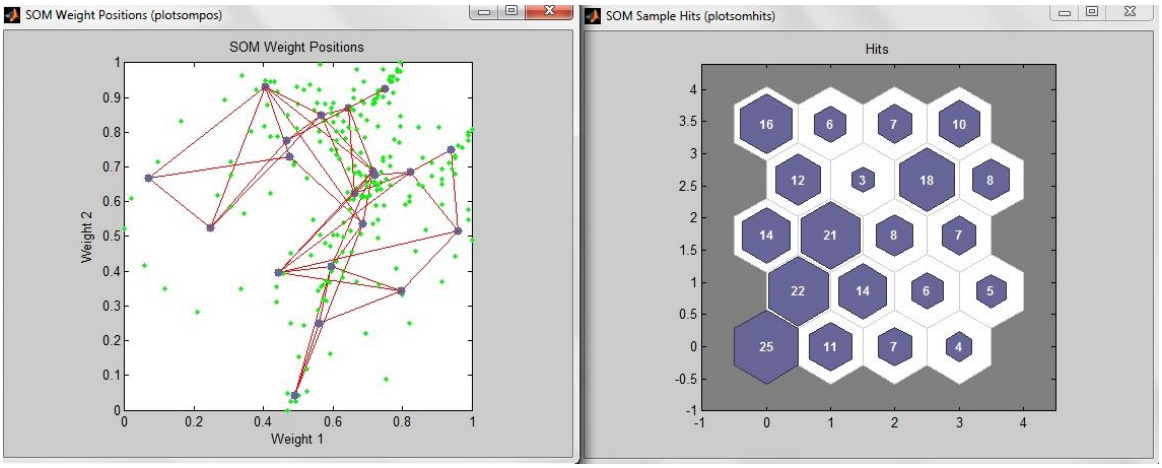
(c) SOM at 1000 iterations



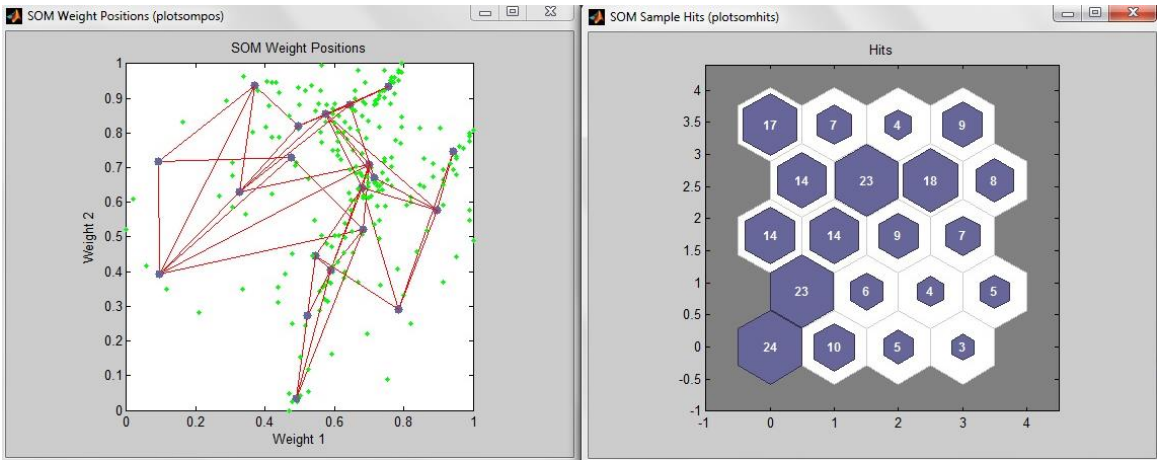
(d) SOM at 2000 iterations



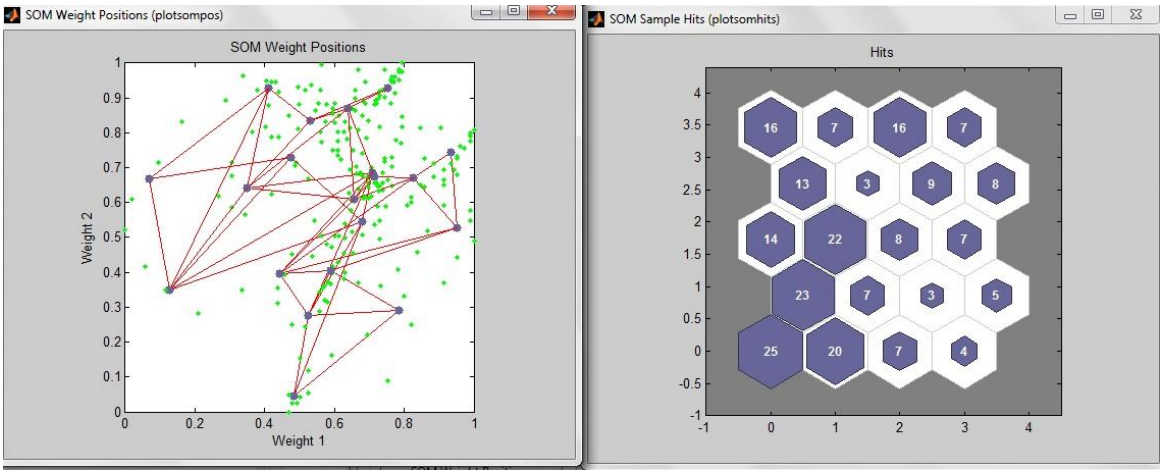
(e) SOM at 3000 iterations



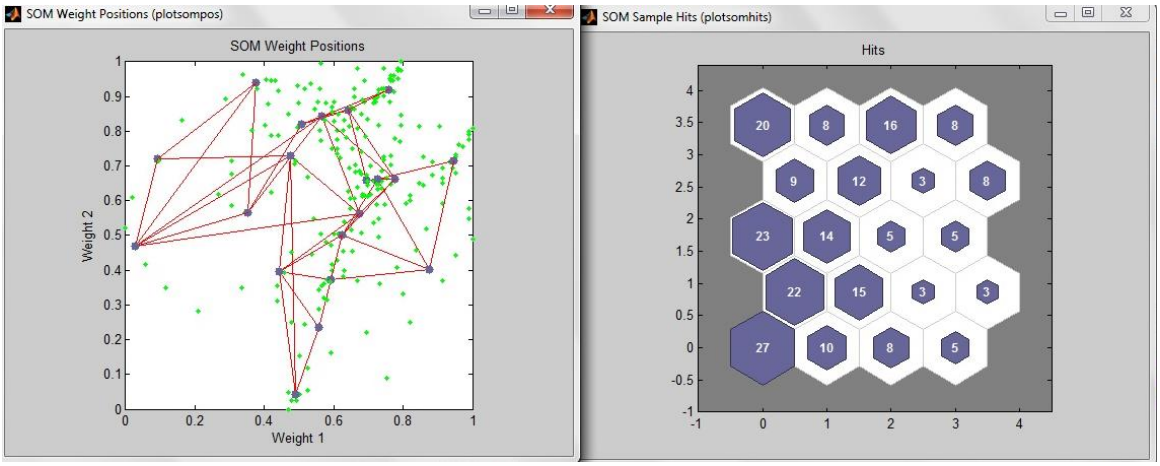
(f) SOM at 4000 iterations



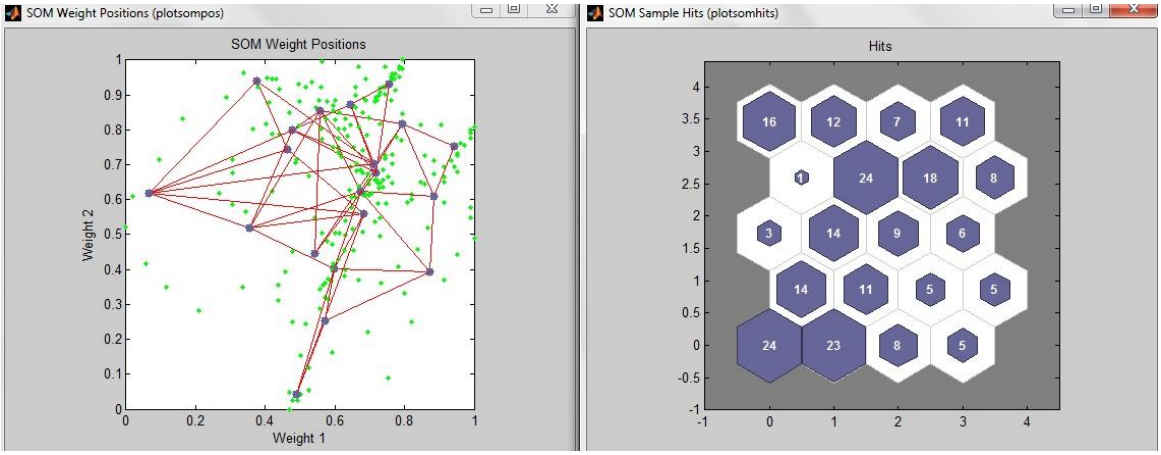
(g) SOM at 5000 iterations



(h) SOM at 10000 iterations



(i) SOM at 15000 iterations



(j) SOM at 20000 iterations

Fig 7.2 (a)-(j) Self Organizing Map for the given sensor array response

Looking at the above figures it can be asserted that with increasing number of iterations the map approaches to the input data spatial distribution. Thus SOM has been realized for our poorly selective sensor data. However there is always a convergence issue with the development of a self-organizing feature map. Computational complexity can be an intriguing feature in the design of such maps.

Chapter 8: Future Scope of this Work

After the study of the pattern analysis techniques presented in the present work following assertions can be made regarding the possible future work.

1) Hardware implementation of the algorithms used PCA using GHA in particular
2) Further development of techniques so as to further increase efficiency of ICA algorithm for high to extremely high dimensional data. These may include integration of ICA algorithm with Fuzzy KNN, SVM and Fuzzy SVM classifiers.
3) Real Time Identification of Individual Gases.

References

[1] K. Persaud and G.H. Dodd, “Analysis of discrimination mechanisms of the mammalian olfactory system using a model nose”, *Nature*, vol. 299, 352-355, 1982

[2] A. Ikagami and M. Kaneyasu, “Olfactory detection using integrated sensors”, In Proc. 3rd International Conference on Solid State Sensors and Actuators (*Transducers ‘85’*), Philadelphia, PA, USA, 136-139, June 7-11, 1985

[3] S. Zaromb and J.R.Steller, “Theoretical basis for identification and measurement of contaminants using an array of sensors having partially overlapping sensitivities”, *Sensors and Actuators*, vol. 6, 225-243, 1984

[4] J.W. Gardner, “Pattern recognition in the Warwick Electronic Nose”, 8th International Congress of European Chemoreception Research Organization, University of Warwick, UK, July, 1987

[5] D. Dably and H.V. Shurmer, “Pattern recognition system for gas sensor arrays”, In Proc. Eurosensors II, Enschede, The Nehterlands, 147, 1988

[6] H.V. Shurmer and J.W. Gardner, “Odor discrimination with an Electronic Nose”, *Sensors and Actuators*, vol. 8, 1-11, 1992

[7] R.G. Osuna, “Pattern analysis for machine olfaction: A review”, *IEEE Sensors Journa*l, vol. 2, no. 3, 189-202, 2002

[8] B.S. Hoffheins and R.J. Lauf, “Gas sensor arrays for olfactory analysis: Issues and opportunities”, *In Proc. Sensors Expo.*, 205-1-205-7, 1988

[9] H.V. Shurmer, J.W. Gardner, and P. Corcoran, “Intelligent vapour discrimination using a composite twelve element sensor array”, *Sensors and Actuators B*, vol. 1, 256-260, 1990

[10] H.V. Shurmer, J.W. Gardner and H.T. Chan, “The application of discriminating techniques to alcohols and tobaccos using tin oxide sensors”, *Sensors and Actuators*, vol. 18, 361-371, 1989

[11] J.W. Gardner, E.L. Hines, and H.C. Tang, “Detection of vapours and odours from a multisensor array using pattern recognition techniques, Part 2, Artificial Neural Networks”, *Sensors and Actuators B*, vol. 9, 9-15, 1992

[12] J.W. Gardner, H.V. Shurmer, and T.T. Tan, "Application of Electronic Nose to the discrimination of coffee", *Sensors and Actuators B,* vol. 6, 71-75, 1992

[13] J.W. Gardner, "Detection of vapours and odours from a multisensor array using pattern recognition techniques, Part 1, Principal Component and Cluster Analysis", *Sensors and Actuators B,* vol. 4, 109-115, 1991

[14] M. Pardo, G. Sberveglieri, S. Gardini, E. Dalcanale, "A hierarchical classification scheme for an Electronic Nose", *Sensors and Actuators B,* Vol. 69, 2000, pp. 359-365

[15] P. Ciosek, Z. Brzozka and Wroblewski, "Classification of Beverages Using a Reduced Sensor Array", *Sensors and Actuators B,* vol. 103, 76-83, 2004

[16] Jesús Brezmes, Ma. Luisa López Fructuoso, Eduard Llobet, Xavier Vilanova, Inmaculada Recasens, Jorge Orts, Guillermo Saiz, and Xavier Correig, "Evaluation of an Electronic Nose to assess fruit ripeness", *IEEE Sensors Journal,* Vol. 5, No. 1, 2005, pp. 97-108

[17].Jesús Lozano, José Pedro Santos, Manuel Aleixandre, Isabel Sayago, Javier Gutiérrez, and Maria Carmen Horrillo, "Identification of typical wine aromas by means of an Electronic Nose", *IEEE Sensors Journal,* Vol. 6, No. 1 2006, pp. 173-178

[18].Sunil K. Jha, R. D. S. Yadava, "Denoising by singular value decomposition and its application to Electronic Nose data processing", *IEEE Sensors Journal,* Vol. 11, No. 1, 2011, pp. 35-44

[19] G. Horner, C. Hierod, "Gas analysis by partial model building", *Sensors and Actuators B,* Vol. 2, 1990, pp. 173–174

[20] W. M. Sears, K. Colbow, "Selective thermally cycled gas sensing using fast Fourier transform techniques", *Sensors and Actuators B,* Vol. 2, 1990, pp. 283–289

[21] J. C. Legras, M. Priel, C. Ranson, "Application of multiple regression method and automatic testing equipment to the characterization of smart sensors", *Sensors and Actuators B,* 12, 1997, pp. 235–243

[22] J. W. Gardner, H. V. Shurmer, T. T. Tan, "Application of electronic nose to the discrimination of coffees", *Sensors and Actuators B,* Vol. 6, 1992, pp. 71–75

[23] M. S. Nayak, “Transformed cluster analysis: an approach to the identification of gases/odors using integrated gas sensor array”, *Sensors and Actuators B,* Vol. 12, 1993, pp. 103–110

[24] Corrado Di Natale, Eugenio Martinelli, Arnaldo D’ Amico, ”Counteraction of environmental disturbances of electronic nose data by independent component analysis”, *Sensors and Actuators B,* Vol. 82, 2002, pp. 158-165

[25] Martin Kermit, Oliver Tomic, “Independent component analysis applied on gas sensor array measurement data”, *IEEE Sensors Journal*, Vol. 3, No. 2, 2003, pp. 218-228.

[26] P. Wide, F. Winquist and D. Driankov, “An air quality sensor system with fuzzy classification”, *Meas. Sci. Technol*., vol. 8, 138-146, 1997

[27] G. Bargagna, B. Lazzerini and A.C. Partridge, “Fuzzy Logic Classification of Olive Oils”, In Electronic Noses and Olfaction 2000, J.W. Gardner and K.C., Persaud, Eds., Bristol, UK, IOP, 2000

[28] T. Sundic, A. Perera, S. Marco, A. Pardo, A. Ortego and J. Samitier, “Fuzzy logic processing in combined carbon monoxide and methane domestic gas alarms”, In Electronic Noses and Olfaction 2000, J.W. Gardner and K.C., Persaud, Eds., Bristol, UK, IOP, 2000

[29] James M. Keller, Michael R. Gray, James A. Givens, Jr.,”A Fuzzy K-Nearest Neighbor Algorithm”, *IEEE Transactions on Systems, Man and Cybernatics*, Vol. SMC-15, No.4,July/August 1985

[30] T. Kohonen, “Self-organized formation of topologically correct feature maps”, *Biol. Cybern.,* vol. 43, 59-69, 1982

[31] S. Marco, A. Ortego, A. Pardo and J. Samitier, “Gas identification with tin oxide sensor arrays and self-organizing maps: Adaptive correction of sensor drifts”, *IEEE Trans. Instrum. Measurement*, vol. 47, 316-320, 1998

[32] C. Di Natale, A. Macagnano, R. Paolesse, E. Tarizzo, A. D’Amico, F. Davide, T. Boxhi, M.Faccio, G. Ferri, F. Sinesio, F.M. Bucarelli, E. Moneta, G.B. Quaglia, “A Comparison Between an Electronic Nose and Human Olfaction in a Selected Case Study”, International Conference on Solid State Sensors and Actuators, 1997. *TRANSDUCERS '97 (Volume 2,)* Chicago., 1997

[33] S. Zampolli, I. Elmi, F. Ahmed, M. Passini, G. C. Cardinali, S. Nicoletti, L. Dori, "An electronic nose based on solid state sensor arrays for low-cost indoor air quality monitoring applications", *Sensors and Actuators B,* Vol. 101, 2004, pp. 39-46.

[34] Hyung-Ki Honga, Chul Han Kwona Seung-Ryeol Kima, Dong Hyun Yuna, Kyuchung Lee, Yung Kwon Sung, "Portable electronic nose system with gas sensor array and artificial neural network", *Sensors and Actuators B,* Vol. 66, 2000, pp. 49-52.

[35] N. El Barbria, A. Amaria, M. Vinaixab, B. Bouchikhia, X. Correigb, E. Llobetb, "Building of a metal oxide gas sensor-based electronic nose to assess the freshness of sardines under cold storage", *Sensors and Actuators B,* Vol. 1, 2007, pp. 235-244.

[36] V. Simakov, A. Voroshilov, A. Grebinnikov, N. Kucherenko, O. Yakusheva, V. Kisin, "Gas identification by quantitative analysis of conductivity-vs-concentration dependence for SnO_2 sensors", *Sensors and Actuators B,* Vol. 137, 2009, pp. 456-461.

[37] A. Chaturvedi, V. N. Mishra, R. Dwivedi, and S. K. Srivastava, "Response of oxygen plasma treated thick film tin oxide sensor array for LPG, CCl_4, CO, and C_3H_7OH", *Microelectronics Journal,* Vol. 30, 1999, pp. 259–264.

[38] R.O. Duda, P.E. Hart, D.G. Stork, "*Pattern Classification",* 2nd Edition, Wiley, New York, 2000

[39] Simon Haykin, "Neural Networks and Learning Machines", 3rd Edition, PHI, New Delhi, 2010

[40] S. Amari, A. Cochoki, H.H. Yang, "A new learning algorithm for blind source separation", *Advances in Neural Information Processing Systems*, Vol.8, pp.757-763, Cambridge, MA:MIT Press, 1996

[41] D.T. Pham, P. Garrat, C. Jutten, "Separation of a mixture of independent sources through a maximum likelihood approach", Proceedings of EUSIPCO, pp. 771-774, 1992

[42] A.J. Bell, T.J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution", Neural Computation, Vol. 6,pp.1129-1159, 1995

[43] A Hyvärinen, E. Oja, "A fast fixed-point algorithm for independent component analysis", Neural Computation, Vol.9, pp. 1483-1492, 1997

[44] L.A. Zadeh, "Fuzzy sets", *Inf. Control,* vol. 8, 338-353, 1965

[45] MATLAB® *Neural Network Toolbox Documentation*, MATHWORKS Inc., USA, 2011

Publications:

1. **Akash Agarwal**, Ravi Kumar, Amit K. Kohli, R. Dwivedi, "Performance Comparison of PCA and ICA Pre-processors in Identification of Individual Gases Using Response of a Poorly Selective Solid-state Sensor Array", *Sensors & Transducers Journal*, Vol. 145, Issue 10, October 2012, pp. 191-202 (ISSN 1726-5479)

2. **Akash Agarwal**, Ravi Kumar, "Fuzzy-KNN Classifier for Identification of Individual Gases Using Response of a Poorly Selective Solid-state Sensor Array", *ICECIT-2013*, Thapar University, Patiala, Punjab, India, 4-5th October 2013 (Accepted)

Appendix A

MATLAB code for Independent Component Analysis

%PART1: MIXING & UNMIXING MATRIX CALCULATION

```
clc
close all
clear all
train=importdata('anytrainingdata.txt');
test=importdata('anytestingdata.txt')
r1=size(train);
s1=r1(1,1);
r2=size(test);
s2=r2(1,1);
d1=r1(1,2);
d2=r2(1,2);
c=input('Number of Independent Components')
if(c<min(d1,d2) || c==min(d1,d2))
if(d1==d2)
d=d1;
display('It is a valid data set')
m=mean(train);
z=mean(test);
for i=1:s1
x(i,:)=train(i,:)-m;
end
for i=1:s2
y(i,:)=test(i,:)-z;
end
yy=cov(y);
[E,D]=eig(yy);
DD=D^-0.5;
Y=E*DD*E'*y';
Y=Y';
xx=cov(x);
[E,D]=eig(xx);
DD=D^-0.5;
X=E*DD*E'*x';
X=X';
w=rand(c,d);
for i=1:c
k=sqrt(sum(w(i,:).^2));
w(i,:)=w(i,:)/k;
end
wn(1:c,1:d)=0;
delta(1:c,1)=0.1;
p=0;
```

```
zzi=1;
while(delta(zzi,1)<0.995)
p=p+1;
if(p<2000)
for i=1:s1
J(i,:)=(X(i,:).*(tanh(X(i,:)*w(zzi,:)')));
K(i,:)= (sech(X(i,:)*w(zzi,:)'))^2;
end
wn(zzi,:)=(mean(K)*w(zzi,:))-mean(J);
k=sqrt(sum(wn(zzi,:).^2));
wn(zzi,:)=wn(zzi,:)/k;
delta(zzi,1)=(abs(wn(zzi,:)*w(zzi,:)'));
w(zzi,:)=wn(zzi,:);
else
break;
end
end
delta(zzi,1)
zzi=2;
for zzi=2:c
p=0;
while(delta(zzi,1)<0.995)
p=p+1;
if (p<2000)
for i=1:s1
J(i,:)=(X(i,:).*(tanh(X(i,:)*w(zzi,:)')));
K(i,:)= (sech(X(i,:)*w(zzi,:)'))^2;
end
wn(zzi,:)=(mean(K)*w(zzi,:))-mean(J);
for jji=1:zzi-1
wn(zzi,:)=wn(zzi,:)-(((wn(zzi,:)*wn(jji,:)')*wn(jji,:)));
end
k=sqrt(sum(wn(zzi,:).^2));
wn(zzi,:)=wn(zzi,:)/k;
delta(zzi,1)=(abs(wn(zzi,:)*w(zzi,:)'));
w(zzi,:)=wn(zzi,:);
else
break;
end
end
end
W=wn;
A=inv(W);
display('Unmixing Matrix is:')
W
display('Mixing Matrix is:')
A
else
display('invalid data dimensions of training and testing sets dont match')
end
```

```
else
display('Invalid number of independent components')
end
```

%PART2: COMPONENT AND ERROR VECTOR PLOTTING FOR SENSOR DATA OF 224 SAMPLES AND 5 DIMENSION EACH WITH EACH CLASS HAHING 56 SAMPLES (37 TRAINING + 19 TESTING)

```
ica=(W*X');
u=W*m';
v=W*z';

for i=1:s1
    icam(:,i)=ica(:,i)+u;
end

ica=ica';
icam=icam';
tm=W*Y';

for i=1:s2
   tzm(:,i)=tm(:,i)+v;
end

tm=tm';
tzm=tzm';

plot(icam(1:37,1),icam(1:37,2),'*',icam(38:74,1),icam(38:74,2),'*',icam(75:111,1),icam(7
5:111,2),'*',icam(112:148,1),icam(112:148,2),'*')
xlabel('1st independent Component')
ylabel('2nd independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs second IC(training data)')
figure
plot(tzm(1:19,1),tzm(1:19,2),'*',tzm(20:38,1),tzm(20:38,2),'*',tzm(39:57,1),tzm(39:57,2),'
*',tzm(58:76,1),tzm(58:76,2),'*')
xlabel('1st independent Component')
ylabel('2nd independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs second IC(testing data)')
figure
plot(icam(1:37,1),icam(1:37,3),'*',icam(38:74,1),icam(38:74,3),'*',icam(75:111,1),icam(7
5:111,3),'*',icam(112:148,1),icam(112:148,3),'*')
xlabel('1st independent Component')
ylabel('3rd independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs third IC(training data)')
figure
plot(tzm(1:19,1),tzm(1:19,3),'*',tzm(20:38,1),tzm(20:38,3),'*',tzm(39:57,1),tzm(39:57,3),'
*',tzm(58:76,1),tzm(58:76,3),'*')
```

```
xlabel('1st independent Component')
ylabel('3rd independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs third IC(testing data)')
figure
plot(icam(1:37,1),icam(1:37,4),'*',icam(38:74,1),icam(38:74,4),'*',icam(75:111,1),icam(7
5:111,4),'*',icam(112:148,1),icam(112:148,4),'*')
xlabel('1st independent Component')
ylabel('4th independent Component')
title('ICA plot for first vs fourth IC(training data)')
legend('LPG','CCL4','CO','C3H7OH')
figure
plot(tzm(1:19,1),tzm(1:19,4),'*',tzm(20:38,1),tzm(20:38,4),'*',tzm(39:57,1),tzm(39:57,4),'
*',tzm(58:76,1),tzm(58:76,4),'*')
xlabel('1st independent Component')
ylabel('4th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs fourth IC(testing data)')
figure
plot(icam(1:37,1),icam(1:37,5),'*',icam(38:74,1),icam(38:74,5),'*',icam(75:111,1),icam(7
5:111,5),'*',icam(112:148,1),icam(112:148,5),'*')
xlabel('1st independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs fifth IC(training data)')
figure
plot(tzm(1:19,1),tzm(1:19,5),'*',tzm(20:38,1),tzm(20:38,5),'*',tzm(39:57,1),tzm(39:57,5),'
*',tzm(58:76,1),tzm(58:76,5),'*')
xlabel('ist independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for first vs fifth(testing data)')
figure
plot(icam(1:37,2),icam(1:37,3),'*',icam(38:74,2),icam(38:74,3),'*',icam(75:111,2),icam(7
5:111,3),'*',icam(112:148,2),icam(112:148,3),'*')
xlabel('2nd independent Component')
ylabel('3rd independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for second vs third IC(training data)')
figure
plot(tzm(1:19,2),tzm(1:19,3),'*',tzm(20:38,2),tzm(20:38,3),'*',tzm(39:57,2),tzm(39:57,3),'
*',tzm(58:76,2),tzm(58:76,3),'*')
xlabel('2nd independent Component')
ylabel('3rd independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for second vs third(testing data)')
figure
plot(icam(1:37,2),icam(1:37,4),'*',icam(38:74,2),icam(38:74,4),'*',icam(75:111,2),icam(7
5:111,4),'*',icam(112:148,2),icam(112:148,4),'*')
xlabel('2nd independent Component')
```

```
ylabel('4th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for second vs fourth IC(training data)')
figure
plot(tzm(1:19,2),tzm(1:19,4),'*',tzm(20:38,2),tzm(20:38,4),'*',tzm(39:57,2),tzm(39:57,4),'
*',tzm(58:76,2),tzm(58:76,4),'*')
xlabel('2nd independent Component')
ylabel('4th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for second vs fourth IC(testing data)')
figure
plot(icam(1:37,2),icam(1:37,5),'*',icam(38:74,2),icam(38:74,5),'*',icam(75:111,2),icam(7
5:111,5),'*',icam(112:148,2),icam(112:148,5),'*')
xlabel('2nd independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for second vs fifth IC(training data)')
figure
plot(tzm(1:19,2),tzm(1:19,5),'*',tzm(20:38,2),tzm(20:38,5),'*',tzm(39:57,2),tzm(39:57,5),'
*',tzm(58:76,2),tzm(58:76,5),'*')
xlabel('2nd independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for second vs fifth IC(testing data)')
figure
plot(icam(1:37,3),icam(1:37,4),'*',icam(38:74,3),icam(38:74,4),'*',icam(75:111,3),icam(7
5:111,4),'*',icam(112:148,3),icam(112:148,4),'*')
xlabel('3rd independent Component')
ylabel('4th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for third vs fourth IC(training data)')
figure
plot(tzm(1:19,3),tzm(1:19,4),'*',tzm(20:38,3),tzm(20:38,4),'*',tzm(39:57,3),tzm(39:57,4),'
*',tzm(58:76,3),tzm(58:76,4),'*')
xlabel('3rd independent Component')
ylabel('4th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for third vs fourth IC(testing data)')
figure
plot(icam(1:37,3),icam(1:37,5),'*',icam(38:74,3),icam(38:74,5),'*',icam(75:111,3),icam(7
5:111,5),'*',icam(112:148,3),icam(112:148,5),'*')
xlabel('3rd independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for third vs fifth IC(training data)')
figure
plot(tzm(1:19,3),tzm(1:19,5),'*',tzm(20:38,3),tzm(20:38,5),'*',tzm(39:57,3),tzm(39:57,5),'
*',tzm(58:76,3),tzm(58:76,5),'*')
xlabel('3rd independent Component')
ylabel('5th independent Component')
```

```
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for third vs fifth IC(testing data)')

figure
plot(icam(1:37,4),icam(1:37,5),'*',icam(38:74,4),icam(38:74,5),'*',icam(75:111,4),icam(7
5:111,5),'*',icam(112:148,4),icam(112:148,5),'*')
xlabel('4th independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for fourth vs fifth IC(training data)')
figure
plot(tzm(1:19,4),tzm(1:19,5),'*',tzm(20:38,4),tzm(20:38,5),'*',tzm(39:57,4),tzm(39:57,5),'
*',tzm(58:76,4),tzm(58:76,5),'*')
xlabel('4th independent Component')
ylabel('5th independent Component')
legend('LPG','CCL4','CO','C3H7OH')
title('ICA plot for fourth vs fifth IC(testing data)')
obs=A*icam';
obs=obs';
obs_test=A*tzm';
obs_test=obs_test';
error=train-obs;
error_test=test-obs_test;
[errorcoeff,errorscore,errorlatent]=princomp(error);

[errorcoeff_test,errorscore_test,errorlatent_test]=princomp(error_test)

figure
plot(errorscore(1:37,1),errorscore(1:37,2),'*',errorscore(38:74,1),errorscore(38:74,2),'*',er
rorscore(75:111,1),errorscore(75:111,2),'*',errorscore(112:148,1),errorscore(112:148,2),'*
')
xlabel('First Principal Component of error score ‘)
ylabel('Second Principal Component of error score ')
legend('LPG','CCL4','CO','C3H7OH')
title('Two dim. PCA plot for training data error vector')
figure
plot3(errorscore(1:37,1),errorscore(1:37,2),errorscore(1:37,3),'*',errorscore(38:74,1),error
score(38:74,2),errorscore(38:74,3),'*',errorscore(75:111,1),errorscore(75:111,2),errorscor
e(75:111,3),'*',errorscore(112:148,1),errorscore(112:148,2),errorscore(112:148,3),'*')
xlabel('First Principal Component of error score ‘)
ylabel('Second Principal Component of error score’ )
zlabel('Third Principal Component of error score’')
legend('LPG','CCL4','CO','C3H7OH')
title('Three dimensional PCA plot for training data error vector')
figure
plot(errorscore_test(1:19,1),errorscore_test(1:19,2),'*',errorscore_test(20:38,1),errorscore
_test(20:38,2),'*',errorscore_test(39:57,1),errorscore_test(39:57,2),'*',errorscore_test(58:7
6,1),errorscore_test(58:76,2),'*')
legend('LPG','CCL4','CO','C3H7OH')
title('Two dim. PCA plot for testing data error vector')
```

```
xlabel('First Principal Component ')
ylabel('Second Principal Component ')
figure
plot3(errorscore_test(1:19,1),errorscore_test(1:19,2),errorscore_test(1:19,3),'*',errorscore
_test(20:38,1),errorscore_test(20:38,2),errorscore_test(20:38,3),'*',errorscore_test(39:57,1
),errorscore_test(39:57,2),errorscore_test(39:57,3),'*',errorscore_test(58:76,1),errorscore_
test(58:76,2),errorscore_test(58:76,3),'*')
xlabel('First Principal Component of error score ')
ylabel('Second Principal Component of error score')
zlabel('Third Principal Component of error score ')
legend('LPG','CCL4','CO','C3H7OH')
title('Three dimensional PCA plot for testing data error vector')
```

Appendix B

MATLAB Code: PCA on the Sensor Array Data

```
clc
close all
clear all
train=importdata('trainingdata.txt');  %
test=importdata('testingdata.txt')
mean_testing_data=mean(testing_data);
for i=1:76
centered_testing_data(i,:)=testing_data(i,:)-mean_testing_data ;
end
[pca,score,latent]=princomp(training_data)
testing_score= centered_testing_data*pca;
plot(score(1:37,1),score(1:37,2),'*',score(38:74,1),score(38:74,2),'*',score(75:111,1),score
(75:111,2),'*',score(112:148,1),score(112:148,2),'*')
title('PCA using PRINCOMP(Training data)')
xlabel('1st Principal Component variance= 62.82%')
ylabel('2nd Principal Component variance= 25.79%')
legend('LPG','CCL4','CO','C3H7OH')
figure
plot(testing_score(1:19,1),testing_score(1:19,2),'*',testing_score(20:38,1),testing_score(2
0:38,2),'*',testing_score(39:57,1),testing_score(39:57,2),'*',testing_score(58:76,1),testing
_score(58:76,2),'*')
title('PCA using PRINCOMP(Testing data)')
xlabel('1st Principal Component’)
ylabel('2nd Principal Component’)
legend('LPG','CCL4','CO','C3H7OH')
toc
```

Appendix C

MATLAB Code: Fuzzy k-Nearest-Neighbor Classifier for Identification of Individual Gases Using Response of a Poorly Selective Solid-state Sensor Array

```
clc
close all
clear all
importdata('dataset')  % 224 samples of 5 dimension each with 56 samples of each class
[princ,X,latent]=princomp(Y);
for gh=1:15
n=42-gh+1;
m=14+gh-1;
for yu=1:100

Z=randperm(56);
r=Z(1,1:m);
s=Z(1,m+1:56);
l=1;
for l=1:n
icam(l,:)=X(s(l),1:3);
icam(n+l,:)=X(56+s(l),1:3);
icam((2*n+l),:)=X(112+s(l),1:3);
icam((3*n+l),:)=X(168+s(l),1:3);
end
for o=1:m
tzm(o,:)=X(r(o),1:3);
tzm(m+o,:)=X(56+r(o),1:3);
tzm((2*m+o),:)=X(112+r(o),1:3);
tzm((3*m+o),:)=X(168+r(o),1:3);
end
for i=1:n
icam(i,:)=icam(i,:)./(std(icam(1:n,:)));
end
for i=n+1:2*n
icam(i,:)=icam(i,:)./(std(icam(n+1:2*n,:)));
end
for i=(2*n+1):3*n
icam(i,:)=icam(i,:)./(std(icam((2*n+1):(3*n),:)));
end
for i=(3*n+1):4*n
icam(i,:)=icam(i,:)./(std(icam((3*n+1):4*n,:)));
end
```

```
for i=1:m
tzm(i,:)=tzm(i,:)./(std((tzm)));
end
for i=m+1:2*m
tzm(i,:)=tzm(i,:)./(std((tzm)));
end
for i=(2*m+1):3*m
tzm(i,:)=tzm(i,:)./(std((tzm)));
end
for i=(3*m+1):4*m
tzm(i,:)=tzm(i,:)./(std((tzm)));
end
m_lpg=mean(icam(1:n,:));
m_ccl4=mean(icam(n+1:2*n,:));
m_co=mean(icam((2*n+1):3*n,:));
m_c3h7oh=mean(icam((3*n+1):(4*n),:));

d1=pdist2(icam,m_lpg);
d2=pdist2(icam,m_ccl4);
d3=pdist2(icam,m_co);
d4=pdist2(icam,m_c3h7oh);

dmax_lpg=max(d1(1:n,1));
dmin_lpg=min(d1(1:n,1));
c_lpg=dmax_lpg-dmin_lpg;
for i=1:4*n
fuzzy1(i,1)=((dmax_lpg-d1(i,1))/c_lpg);
end

dmax_ccl4=max(d2(n+1:2*n,1));
dmin_ccl4=min(d2(n+1:2*n,1));
c_ccl4=dmax_ccl4-dmin_ccl4;
for i=1:4*n
fuzzy1(i,2)=((dmax_ccl4-d2(i,1))/c_ccl4);
end

dmax_co=max(d3(2*n+1:3*n,1));
dmin_co=min(d3(2*n+1:3*n,1));
c_co=dmax_co-dmin_co;
for i=1:4*n
fuzzy1(i,3)=((dmax_co-d3(i,1))/c_co);
end

dmax_c3h7oh=max(d4(3*n+1:4*n,1));
dmin_c3h7oh=min(d4(3*n+1:4*n,1));
```

```
c_c3h7oh=dmax_c3h7oh-dmin_c3h7oh;
for i=1:4*n
fuzzy1(i,4)=((dmax_c3h7oh-d4(i,1))/c_c3h7oh);
end


count1=0;count=0;
for i=1:4*n
for j=1:4
if(fuzzy1(i,j)==0)
fuzzy1(i,j)=0.0001;
elseif(fuzzy1(i,j)<0)
fuzzy1(i,j)=0;

elseif(fuzzy1(i,j)>1)
fuzzy1(i,j)=0;
count=count+1;
else
fuzzy1(i,j)=fuzzy1(i,j);
end
end
end

for i=1:4*n
for j=1:4
if(fuzzy1(i,j)==1)
for k=1:4
if(fuzzy1(i,k)~=1)
fuzzy1(i,k)=0;
end
end
end
end
end

for i=1:4*n
su(i,1)=sum(fuzzy1(i,:));
fuzzy1(i,:)=(fuzzy1(i,:))/(su(i,1));
end

for i=1:4*n
sn(i,1)=min(fuzzy1(i,:));
for j=1:4
if(fuzzy1(i,j)==sn(i,1))
fuzzy(i,j)=0;
end
end
end
```

```
for t=1:40 %shows t nearest neighbours are taken at a time
[a,D]=knnsearch(icam,tzm,'k',t);

for q=1:4*m


for i=1:4
f(q,i)=0;
g(q,i)=0;
for j=1:t

f(q,i)=f(q,i)+(fuzzy1(a(q,j),i)*(1/D(q,j)));
g(q,i)=g(q,i)+(1/D(q,j));
end
fuzzn(q,i)=f(q,i)/g(q,i);
end
end
c(t,1)=0;d(t,1)=0;e(t,1)=0;l(t,1)=0;
for h=1:m
if(max(fuzzn(h,:))==fuzzn(h,1))
c(t,1)=c(t,1)+1;
end
end
for h=m+1:2*m
if(max(fuzzn(h,:))==fuzzn(h,2))
d(t,1)=d(t,1)+1;
end
end

for h=2*m+1:3*m
if(max(fuzzn(h,:))==fuzzn(h,3))
e(t,1)=e(t,1)+1;
end
end
for h=3*m+1:4*m
if(max(fuzzn(h,:))==fuzzn(h,4))
l(t,1)=l(t,1)+1;
end
end
end
```

```
za(:,yu)=c;
zb(:,yu)=d;
zc(:,yu)=e;
zd(:,yu)=l;
end

e1=mean(za,2);
e2=mean(zb,2);
e3=mean(zc,2);
e4=mean(zd,2);
ef1(gh,1)=(mean(e1)/m)*100;
ef2(gh,1)=(mean(e2)/m)*100;
ef3(gh,1)=(mean(e3)/m)*100;
ef4(gh,1)=(mean(e4)/m)*100;
end

plot(1:1:15,ef1/100,'--rs','linewidth',2.5,'markeredgecolor','k','markersize',4)
hold on
plot(1:1:15,ef2/100,'--gs','linewidth',2.5,'markeredgecolor','k','markersize',4)
hold on
plot(1:1:15,ef3/100,'--bs','linewidth',2.5,'markeredgecolor','k','markersize',4)
hold on
plot(1:1:15,ef4/100,'--ms','linewidth',2.5,'markeredgecolor','k','markersize',4)
title('Efficiency plot')
xlabel('Size of the testing vector')
ylabel('Efficiency = value *100%')
```

Appendix D

PART1: MATLAB Code for SVM Classifier for same data set using Polynomial kernel

```
clc
close all
clear all

X=importdata('train.txt') %training data of 148 samples with each sample 5 dimensional in nature and each class having 37 samples
Y=importdata('testingdata.txt') %testing data of 76 samples with sample 5 dimensionsal in nature and each class having 19 sample

[coeff_train,score_train,latent_train]=princomp(X);
[coeff_test,score_test,latent_test]=princomp(Y);

for i=1:37
y(i,1)=-1;
end
for i=38:74
y(i,1)=1;
end
svmstruct=svmtrain(score_train(1:74,1:2),y,'showplot',true)
group1=svmclassify(svmstruct,score_test(1:38,1:2),'showplot',true)
title('LPG in reds vs CCL4 in greens')
figure

for i=1:37
y(i,1)=-1;
end
for i=38:74
y(i,1)=1;
end
svmstruct=svmtrain(score_train(38:111,1:2),y,'showplot',true)
group4=svmclassify(svmstruct,score_test(20:57,1:2),'showplot',true)
title('CCL4 in reds vs CO in greens')
figure
for i=1:37
y(i,1)=-1;
end
for i=38:74
y(i,1)=1;
end
svmstruct=svmtrain(score_train(75:148,1:2),y,'showplot',true)
group6=svmclassify(svmstruct,score_test(39:76,1:2),'showplot',true)
title('CO in reds vs C3H7OH greens')
```

```
figure

for i=1:37
y(i,1)=-1;
end
for i=38:74
y(i,1)=1;
end

for j=1:2

for i=1:37
LPG_CO_train(((j-1)*37)+i,1:2)= score_train(i+((j-1)*74),1:2);
end
for k=1:19
LPG_CO_test(((j-1)*19)+k,1:2)= score_test(k+((j-1)*38),1:2);
end

end
svmstruct=svmtrain(LPG_CO_train,y,'showplot',true)
group2=svmclassify(svmstruct,LPG_CO_test,'showplot',true)
title('LPG in reds vs CO in greens')
figure


for i=1:37
y(i,1)=-1;
end
for i=38:74
y(i,1)=1;
end

for j=1:2

for i=1:37
LPG_C3H7OH_train(((j-1)*37)+i,1:2)= score_train(i+((j-1)*111),1:2);
end
for k=1:19
LPG_C3H7OH_test(((j-1)*19)+k,1:2)= score_test(k+((j-1)*57),1:2);
end

end
svmstruct=svmtrain(LPG_C3H7OH_train,y,'showplot',true)
group3=svmclassify(svmstruct,LPG_C3H7OH_test,'showplot',true)
title('LPG in reds vs C3H7OH in greens')
xlabel('LPG')
figure

for i=1:37
y(i,1)=-1;
```

```
end
for i=38:74
y(i,1)=1;
end

for j=1:2

 for i=1:37
 CCL4_C3H7OH_train(((j-1)*37)+i,1:2)= score_train(i+(((2*j)-1)*37),1:2);
 end
 for k=1:19
 CCL4_C3H7OH_test(((j-1)*19)+k,1:2)= score_test(k+(((2*j)-1)*19),1:2);
 end

end
svmstruct=svmtrain(CCL4_C3H7OH_train,y,'showplot',true)
group5=svmclassify(svmstruct,CCL4_C3H7OH_test,'showplot',true)
title('CCL4 in reds vs C3H7OH in greens')
```

PART2: MATLAB Code for SVM Classifier for same data set using RBF kernel

```
for i=1:37
y(i,1)=-1;
end
for i=38:74
y(i,1)=1;
end
svmstruct=svmtrain(score_train(1:74,1:2),y,'showplot',true,'kernel_function','rbf','rbf_sig
ma',3)
group1=svmclassify(svmstruct,score_test(1:38,1:2),'showplot',true)
title('LPG in reds vs CCL4 in greens')
figure
for j=1:2
for i=1:37
LPG_CO_train(((j-1)*37)+i,1:2)= score_train(i+((j-1)*74),1:2);
end
for k=1:19
LPG_CO_test(((j-1)*19)+k,1:2)= score_test(k+((j-1)*38),1:2);
end
end
svmstruct=svmtrain(LPG_CO_train,y,'showplot',true,'kernel_function','rbf','rbf_sigma',3)
group2=svmclassify(svmstruct,LPG_CO_test,'showplot',true)
title('LPG in reds vs CO in greens')
figure
for j=1:2
for i=1:37
LPG_C3H7OH_train(((j-1)*37)+i,1:2)= score_train(i+((j-1)*111),1:2);
```

```
end
for k=1:19
LPG_C3H7OH_test(((j-1)*19)+k,1:2)= score_test(k+((j-1)*57),1:2);
end
end
svmstruct=svmtrain(LPG_C3H7OH_train,y,'showplot',true,'kernel_function','rbf','rbf_sig
ma',3)
group3=svmclassify(svmstruct,LPG_C3H7OH_test,'showplot',true)
title('LPG in reds vs C3H7OH in greens')
xlabel('LPG')
figure
svmstruct=svmtrain(score_train(38:111,1:2),y,'showplot',true,'kernel_function','rbf','rbf_si
gma',3)
group4=svmclassify(svmstruct,score_test(20:57,1:2),'showplot',true)
title('CCL4 in reds vs CO in greens')
figure
for j=1:2
for i=1:37
CCL4_C3H7OH_train(((j-1)*37)+i,1:2)= score_train(i+(((2*j)-1)*37),1:2);
end
for k=1:19
CCL4_C3H7OH_test(((j-1)*19)+k,1:2)= score_test(k+(((2*j)-1)*19),1:2);
end
end
svmstruct=svmtrain(CCL4_C3H7OH_train,y,'showplot',true,'kernel_function','rbf','rbf_si
gma',3)
group5=svmclassify(svmstruct,CCL4_C3H7OH_test,'showplot',true)
title('CCL4 in reds vs C3H7OH in greens')
figure
svmstruct=svmtrain(score_train(75:148,1:2),y,'showplot',true,'kernel_function','rbf','rbf_si
gma',3)
group6=svmclassify(svmstruct,score_test(39:76,1:2),'showplot',true)
title('CO in reds vs C3H7OH greens')
```

Appendix E

MATLAB code for PCA Using GHA

```
clc
close all
clear all
Y=importdata('dataset');% Input data of 224*5 dimensions with each class having 56
samples
mean_data= mean(Y);
for i=1:224
centered_data(i,:)= Y(i,:)-mean__data;
end
w1=rand(1,5);
k=sqrt(sum(w1.^2));
w1=w1/k
w2=rand(1,5);
k=sqrt(sum(w2.^2));
w2=w2/k
for ri=1:15
for z=1:1000
for i=1:224
y1=centered_data(i,:)* w1';
y2=centered_data(i,:)* w2';
dw1=((0.1)*y1*[centered_data(i,:)-(y1*w1)]);
dw2=((0.1)*y2*[(centered_data(i,:))-(y1*w1)-(y2*w2)]);
wn1=w1+dw1;
wn2=w2+dw2;
k=sqrt(sum(wn1.^2));
wn1=wn1/k;
k=sqrt(sum(wn2.^2));
wn2=wn2/k;
w1=wn1;
w2=wn2;
end
end

pca1=wn1
pca2=wn2
score(:,1)=centered_data*pca1';
score(:,2)=centered__data*pca2';

plot(score(1:56,1),score(1:56,2),'*',score(57:112,1),score(57:112,2),'*',score(113:168,1),s
core(113:168,2),'*',score(169:224,1),score(169:224,2),'*')
c=ri*1000;
title(['Number of iteration is ',num2str(c)])
```

```
xlabel('FIRST PRINCIPAL COMPONENT')
ylabel('SECOND PRINCIPAL COMPONENT')
legend('LPG','CCL4','CO','C3H7OH')
figure
end
```

Appendix F

MATLAB code for Self Organizing Map for Sensor Response

```
clc
close all
clear all
Y=importdata('dataset') % 224 samples of 5 dimension each such that each class having
56 samples in the sat
Y(1:224,1)=(Y(1:224,1)-min(Y(1:224,1)))/(max(Y(1:224,1)-min(Y(1:224,1))));
Y(1:224,2)=(Y(1:224,2)-min(Y(1:224,2)))/(max(Y(1:224,2)-min(Y(1:224,2))));
Y(1:224,3)=(Y(1:224,3)-min(Y(1:224,3)))/(max(Y(1:224,3)-min(Y(1:224,3))));
Y(1:224,4)=(Y(1:224,4)-min(Y(1:224,4)))/(max(Y(1:224,4)-min(Y(1:224,4))));
Y(1:224,5)=(Y(1:224,5)-min(Y(1:224,5)))/(max(Y(1:224,5)-min(Y(1:224,5))));
net = newsom(Y',[4 5]);
s=[200 500 1000 2000 3000 4000 5000 10000 15000 20000]
for i=1:11
net.trainParam.epochs = s(i);
net = train(net,Y');
plotsompos(net,Y')
figure
plotsomhits(net,Y')
end
```