Classification Models and Algorithms in Application of Multi–Sensor Systems to Detection and Identification of Gases

Walaa Khalaf

2006–2007

To my family, and
the woman who gave me
the patience and the courage
to finish this work, my wife.

Acknowledgement

It seems that the long way is coming to its end and it is time to say "*Thanks*" to all of you who walked along with me and shared "the good and the bad". I thank you a lot my dearest friend professor Alfredo Eisenberg, for all the assistance that you offered me to finish my study and my dream to get the Ph.D. degree, also to my great supervisor professor Manlio Gaudioso, really I do not have enough words to appreciate what they offered me, but I am sure they know my feelings toward them.

I would like to present my thanks to my brother Dr. Falah Alwaely, without him I could not get this chance to study here, also to my wife Noor Hashim, for all her support. Big kiss to my dear friend Mohammed Issa and I am thanking him for not complaining about many destroyed weekends, helping me in translations and grammar corrections.

Coming to the "professional" part, that was much more than just professional, one big "*Thank*" to professor Giuseppe Cocorullo and professor Calogero Pace, whom I learned a lot from them. "*Regards and thanks*" to everybody at the Department of D.E.I.S. (Dipartimento di Elettronica, Informatica e Sistemistica)–University of Calabria, especially my colleagues Giuseppe Fedele, Annabella Astorino, Enrico Gorgone, Antonio Fuduli, Marcello Sammarra, Luigi Moccia, Giovanni Giallombardo, Gregorio Sorrentino, Flavia Monaco, Giuseppina Bonavita, and Giovanna Miglionico. Finally, I should thank the people who accommodated me here, at the University of Calabria, and made me feel at home.

Abstract

The objective of the thesis is to adopt advanced machine learning techniques in the analysis of the output of sensor systems. In particular we have focused on the SVM (Support Vector Machine) approach to classification and regression, and we have tailored such approach for the area of sensor systems of the "electronic nose" type.

We designed an Electronic Nose (ENose), containing 8 sensors, 5 of them being gas sensors, and the other 3 being a Temperature, a Humidity, and a Pressure sensor, respectively. Our system (Electronic Nose) has the ability to identify the type of gas, and then to estimate its concentration.

To identify the type of gas we used as classification and regression technique the so called Support Vector Machine (SVM) approach, which is based on statistical learning theory and has been proposed in the broad learning field. The Kernel methods are applied in the context of SVM, to improve the classification quality. Classification means finding the best divider (separator) between two or more different classes without or with minimum number of errors. Many methods for pattern recognition or classification are based on neural network or other complex mathematical models.

In this thesis we describe the hardware equipment which has been designed and implemented. We survey the SVM approach for machine learning and report on our experimentation.

Contents

List of Tables

List of Figures

List of Symbols

n	Number of samples
d	Number of input variables
$\mathbb{R}^d$	Euclidean Space d-dimensional
$\mathcal{G}$	Finite set of classes
Ω	Set of parameters, as in $\mathbf{w} \in \mathbf{\Omega}$
$F(x)$	Cumulative probability distribution function (cdf)
$p(x)$	Probability density function (pdf)
L	Loss function
$P(\mathbf{x}, y)$	Joint probability density function
$p(\mathbf{x}\|y)$	Conditional density
R	Risk function
Λ	Set of abstract parameters
$\mathcal{H}$	Hypothesis space
h	VC–dimension
$\|.\|$	Norm
γ	Margin
d	Signed distance
F	Feature space
α	Lagrange multipliers
$\mathcal{L}$	Primal Lagrangian
$\mathbf{W}$	Dual Lagrangian

C	Regularization parameter
ξ_i	Slack variable
$\langle \mathbf{x}.\mathbf{z} \rangle$	Inner (dot) product between $\mathbf{x}$ and $\mathbf{z}$
$\phi : \mathrm{X} \rightarrow F$	Mapping to feature space
$\mathrm{K}(\mathbf{x}, \mathbf{z})$	Kernel $\langle \phi(\mathbf{x}).\phi(\mathbf{z}) \rangle$
ppm	Parts per million
cc	Cubic centimeter
MW	Molecular weight of gas in gram/mol
ρ	Liquid density in gram/cm^3
$\mathfrak{d}$	Gas density in gram/liter
r	Correlation coefficient

Introduction

We present an Electronic Nose (ENose) which is aimed both at identifying the type of gas and at estimating its concentration. Our system contains 8 sensors, 5 of them being gas sensors, whose sensing element is a tin dioxide (SnO_2) semiconductor, the remaining being a temperature sensor, a humidity sensor, and a pressure sensor. Our integrated hardware-software system uses some machine learning principles to identify at first a new gas sample, and then to estimate its concentration.

In particular we adopt a training model using the Support Vector Machine (SVM) approach to teach the system how to discriminate among different gases, this mean working as classifier, then we apply another training model using also the SVM approach, but here working as an estimator, for each type of gas, to predict its concentration.

We deal with the problems of gases detection and recognition as well as with the estimation of their concentrations. In fact, detection and recognition can be seen as a two–class and a multi–class classification problem, respectively. The detection of volatile organic compounds (VOCs) has become a serious task in many fields, because the fast evaporation rate and toxic nature of VOCs could be dangerous at high concentration levels in air and working ambient for the health of human beings. In fact, the VOCs are also considered as the main reason for allergic pathologies, skin and lung diseases.

To identify the type of gas we use the support vector machine (SVM) approach which was introduced by Vapnik as a classification tool. The SVM method strongly relies on statistical learning theory. Classification is based on the idea of finding the best separating hyperplane (in terms of classification error and separation margin) of two point–sets in the

sample space (which in our case is the Euclidean eight–dimension vector space). Our classification approach includes the possibility of adopting Kernel transformations within the SVM context.

We adopt a multi–sensor scheme and useful information is gathered by combining the outputs of the different sensors. The use of just one sensor does not allow in general to identify the gas. In fact the same sensor output may correspond to different concentrations of many different gases. On the other hand by combining the information coming from several sensors of diverse types we identify the gas and estimate its concentration. We will present the description of our system, producing the details of its construction.

The results of our experiments on four different types of gases (Methanol, Ethanol, Acetone, and Benzene) have been particularly encouraging both in terms of classification errors and concentration prediction.

Chapter 1

Machine Learning and Classification

This chapter introduces the principles and importance of *learning* methodology which is the approach of using examples to synthesize knowledge. The learning problems can be subdivided into four classes:

- Density Estimation: Let f be an unknown density function in $\mathbb{R}^d$, and $X_1, \ldots, X_N$ a random sample with distribution f: provide an estimator $\hat{f}_N$ based on the data.

- Clustering: The process of organizing objects into groups whose members are similar. A cluster is therefore a collection of objects which are similar between themselves and are dissimilar to the objects belonging to the other clusters.

- Classification: Given a training set $(x_1, y_1), \ldots, (x_N, y_N)$, considered as a sample of pairs of random variables (X, Y) with $x_k \in \mathbb{R}^d$ and y_k in a finite set $\mathcal{G}$ of classes, find a function $\hat{f}_n : \mathbb{R}^d \to \mathcal{G}$ which is the best prediction of the true class.

- Regression: Given a training set $(x_1, y_1), \ldots, (x_N, y_N)$, considered as a sample of pairs of random variables (X, Y) with $x_k \in \mathbb{R}^d$ and $y_k \in \mathbb{R}^q$, find a function $\hat{f}_n : \mathbb{R}^d \to \mathbb{R}^q$ to approximate $E(Y|X)$.

We can easily understand the importance of machine learning in many computer–based real world applications. In the sequel we give a survey of the above mentioned classes of problems.

1.1 Machine Learning

Learning is the process of estimating an unknown function or structure of a system using a limited number of observations. There are two major settings of learning, the first one called *supervised learning* which is deriving the required output from a set of inputs. Curve–fitting is a simple example of supervised learning of a function, the examples of input/output functionality are referred to as the *training data* [18, 25, 35]. The second type is called *unsupervised learning*, which is the case when having a training set of vectors without function (output) values for them, the learning task is to gain some understanding of the process that generated the data. This type of learning includes density estimation, clustering, learning the support of a distribution, and so on [8, 14, 18].

There are several ways in which the training set can be used to produce a hypothesis function. The *batch learning* when the training set is available and used all at once to compute the hypothesis function. The entire training set is possibly used to modify a current hypothesis iteratively until an acceptable hypothesis is obtained. The *online learning* uses only one example at a time, and updates the current hypothesis depending on the response to each new example [11].

The learning problem is divided into two parts: specification and estimation. *Specification* consists in determining the parametric form of the unknown distributions, while *estimation* is the process of determining parameters that characterize the specified distributions. The two inductive principles that are most commonly used in the learning process are the *Empirical Risk Minimization* **(ERM)** and the *Maximum Likelihood* **(ML)**.

Machine learning is not just a data base management problem; it is also a part of artificial intelligence. To be intelligent, a system that is in a changing environment should have the ability to learn. If the

system can learn and adapt to such changes, the system designer need not predict and provide solutions for all possible situations. Machine learning requires design of computer programs to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize such parameters using the training data or past experience. The model may be *predictive* to make predictions in the future, or *descriptive* to gain knowledge from data, or both. Machine learning also helps us to find solutions to many problems in vision, speech recognition, and robotics [35].

1.1.1 Input and Output Functions

The situation sketched in Figure 1.1, is when there exists a function f, and the learner job is to ***guess what it is***. We denote by h the hypothesis of the function to be learned. Both f and h are functions of a vector–valued input $\mathbf{X} = (x_1, x_2, \ldots, x_i, \ldots, x_d)$ which has d components. Function h is being implemented by a device that has $\mathbf{X}$ as input and $h(\mathbf{X})$ as output. Both f and h themselves may be vector–valued.

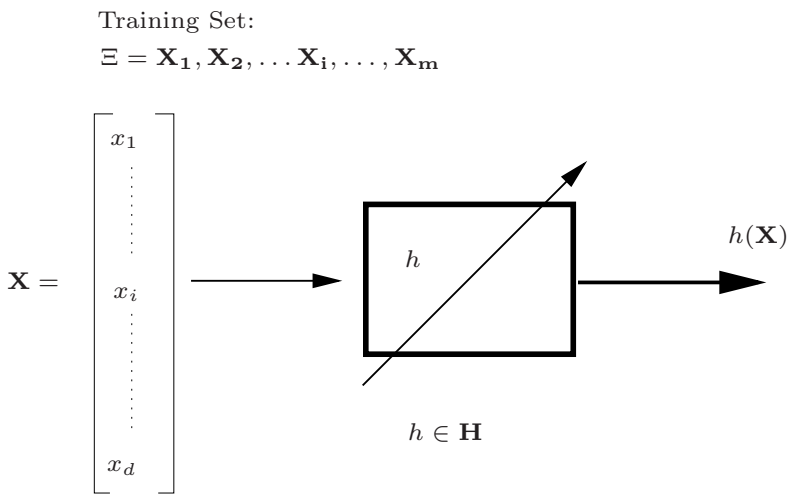

Figure 1.1: An input–output function

We assume *a priori* that the hypothesized function h is selected from a class of functions $\mathcal{H}$. Sometimes we know that f also belongs to this class or to a subset of this class. We select h based on a training set, Ξ, of m input vector examples.

The input vector is called by a variety of names, some of these are *input vector, pattern vector, feature vector, sample, example,* and *instance.* The components, x_i, of the input vector are variously called *features, attributes, input variables,* and *components.* The output may be a real number, in which case the process embodying the function, h, is called a *function estimator*, and the output is called an *output value* or *estimate.* Alternatively, the output may be a categorical value, in which case the process embodying h is variously called a *classifier,*a *recognizer,* or a *categorizer*, and the output itself is called a *label,* a *class,* a *category,* or a *decision.*

1.2 Density Estimation

A classical unsupervised learning task is density estimation. We assume that $f(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \mathbf{\Omega}$ is a set of densities where $\mathbf{w}$ is an **M-**dimensional vector. Let us assume that the unknown density $f(\mathbf{x}, \mathbf{w_0})$ belongs to this class. Assuming that the unlabeled observations (training data) $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]$ were generated independently and identically distributed (i.i.d.) according to some unknown distribution, the task of density estimation is to learn the definition of this probability density function [36]. The *likelihood function* is the probability of seeing $\mathbf{X}$ as a function of $\mathbf{w}$

$$\mathcal{P}(\mathbf{X}|\mathbf{w}) = \prod_{i=1}^{n} f(\mathbf{x}_i, \mathbf{w}), \tag{1.1}$$

the maximum likelihood inductive principle states that we should choose the parameters $\mathbf{w}$ which maximize the likelihood function [8, 35]. Maximizing the *log likelihood function* makes the problem more tractable. This is equivalent to minimizing the **ML** risk functional

$$R_{ML}(\mathbf{w}) = -\sum_{i=1}^{n} \ln f(\mathbf{x}_i, \mathbf{w}), \tag{1.2}$$

empirically estimating the risk function by using the training data depending on the empirical risk minimization inductive principle which is the *average* risk for the training data. This estimate, called the *empirical risk*, is then minimized by choosing the appropriate parameters [14]. The expected risk for density estimation is

$$R(\mathbf{w}) = \int L(f(\mathbf{x}, \mathbf{w}))\, p(\mathbf{x})\, d\mathbf{x}, \tag{1.3}$$

where $L(f(\mathbf{x}, \mathbf{w}))$ is the loss function. Taking an average of the risk over the training data:

$$R_{emp}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} L(\ f(\mathbf{x}_i, \mathbf{w})\ ) \tag{1.4}$$

minimizing the empirical risk (1.4) with respect to $\mathbf{w}$ we can find the optimum parameter values $\mathbf{w}^*$.

Empirical risk minimization (**ERM**) does not specify the particular form of the loss function, therefore; it is a more general inductive principle than maximum likelihood (**ML**) principle [14]. If the loss function is

$$L(f(\mathbf{x}, \mathbf{w})) = -\ln f(\mathbf{x}, \mathbf{w}). \tag{1.5}$$

This mean the **ERM** inductive principle is equivalent to the **ML** inductive principle for density estimation.

1.2.1 Nonparametric Density Estimation

Defining the density by solving the integral equation is the general principle behind nonparametric density estimation

$$\int_{-\infty}^{x} p(u)\ du = F(x), \tag{1.6}$$

where $F(x)$ is the cumulative distribution function (*cdf*). $F(x)$ is approximated by the empirical *cdf* estimated from the training data, because $F(x)$ is unknown, therefore;

$$F_n(x) = \sum_{i=1}^{n} I(x \geq x_i) \tag{1.7}$$

where $I(\ )$ is the indicator function that takes the value **1** if its argument is true and **0** otherwise. As the number of samples tends to infinity, the empirical *cdf* uniformly converges to the true *cdf* . All nonparametric density estimators depend on this asymptotic assumption to give an estimate, since they solve the integral, equation (1.6), using the empirical *cdf*. One of the major drawbacks of nonparametric estimators for density is their poor scaling properties for high–dimensional data [8].

The most widely used method of nonparametric density estimation is the K–nearest neighbors (**KNN**), this method is a simple algorithm, often performs very well and is an important benchmark method. But one drawback of **KNN** is that all the training data must be stored, and a large amount of processing is needed to evaluate the density for a new input pattern [3, 24, 25, 35].

1.3 Clustering

Clustering is an unsupervised learning process, searching for spatial relationships or similarities among data samples, which might be hard to distinguish in high–dimensional feature space [18]. The three basic steps of clustering process are:

1. Defining a dissimilarity measure between examples: typically the Euclidean distance is used.

2. Defining a clustering criterion to be optimized, typically based on within–and between–cluster structure (e.g., elongated, compact or topologically–ordered clusters).

3. Defining a search algorithm to find a "good" assignment of examples to clusters.

Clustering or unsupervised classification is a very difficult problem because data could form clusters with different shapes and sizes as shown in Figure 1.2. Cluster analysis is a very important and useful technique. The speed, reliability, and consistency of a clustering algorithm applied to a large amounts of data constitute strong reasons to use it in applications

such as data mining, information retrieval, image segmentation, signal compression and coding, and machine learning [12].

Figure 1.2: Different Shapes and Sizes of Clusters

There are hundreds of clustering algorithms and new clustering algorithms continue to appear. Most of these algorithms are based on the following two popular clustering techniques: *iterative square-error partitional clustering* and *agglomerative hierarchical clustering* [25]. *Hierarchical techniques* organize data in a nested sequence of groups which can be displayed in the form of a tree. *Square-error partitional algorithms* try to find the partition which either minimizes the within–cluster scatter or maximizes the between–cluster scatter [27].

1.4 Classification

The task of classification is finding the best divider (separator) between two or more different classes, without or with minimum number of errors. In the simplest case there are only two different classes. Estimating a function $f : \mathbb{R}^d \rightarrow \{0, 1\}$, is one possible formalization of this task, using

input-output training data pairs which are assumed to be generated independently and identically distributed (i.i.d.), according to an unknown probability distribution $P(\mathbf{x}, y)$, where

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times Y, \;\; Y = \{0, 1\}.$$

The objective is to define f which will correctly classify unseen examples $(\mathbf{x}, y)$. An example is assigned to the class **1** if $f(\mathbf{x}) \geq 0$ and to the class **0** otherwise. The test examples are assumed to be generated from the same probability distribution $P(\mathbf{x}, y)$ as the training data. A commonly used loss function measures the classification error is [8]

$$L(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } \; y = f(\mathbf{x}) \\ 1 & \text{if } \; y \neq f(\mathbf{x}) \end{cases} \tag{1.8}$$

The best function f is the one that minimizing the expected error (risk)

$$R(f) = \int L(y, f(\mathbf{x})) \; P(\mathbf{x}, y) \; d\mathbf{x} \; dy. \tag{1.9}$$

Learning then becomes the problem of finding the indicator function $f(\mathbf{x})$ (**classifier**) which minimizes the probability of misclassification, equation (1.9), using only the training data. While the underlying probability distribution $P(\mathbf{x}, y)$ is unknown, the risk cannot be minimized directly. So, users try to estimate a function that is close to the optimal one based on the available information.

The conditional densities for each class $p(\mathbf{x}|y = 0)$ and $p(\mathbf{x}|y = 1)$ are estimated via parametric density estimation and the **ML** inductive principle. These estimates will be denoted as $p_0(\mathbf{x}, \alpha^*)$ and $p_1(\mathbf{x}, \beta^*)$, respectively, to indicate that they are parametric functions with parameters chosen via **ML**. *Prior probabilities* are the probability of occurrence of each class, $P(y = 0)$ and $P(y = 1)$ are assumed to be known or estimated. Using **Bayes** theorem, it is possible to determine for a given observation $\mathbf{x}$ the probability that the observation belongs to each class, which is called *posterior probabilities*, that can be used to construct a discriminant rule. This rule chooses the output class which has the maximum

posterior probability [14, 25]. Calculating the posterior probabilities for each class by using **Bayes** rule [8, 35]:

$$\begin{aligned} P(y=0|\mathbf{x}) &= \frac{p_0(\mathbf{x},\alpha^*)P(y=0)}{p(\mathbf{x})} \\ P(y=1|\mathbf{x}) &= \frac{p_1(\mathbf{x},\beta^*)P(y=1)}{p(\mathbf{x})} \end{aligned} \tag{1.10}$$

After calculating the posterior probabilities, $\mathbf{x}$ can be classified by using the following rule:

$$f(\mathbf{x}) = \begin{cases} \mathbf{0} & \text{if} \quad p_0(\mathbf{x},\alpha^*)P(y=0) > p_1(\mathbf{x},\beta^*)P(y=1) \\ \\ \mathbf{1} & \text{otherwise} \end{cases} \tag{1.11}$$

Equivalently, the rule can be written as

$$f(\mathbf{x}) = \mathbf{I}\left(\ln p_1(\mathbf{x},\beta^*) - \ln p_0(\mathbf{x},\alpha^*) + \ln\frac{P(y=1)}{P(y=0)} > 0\right), \tag{1.12}$$

where $\mathbf{I}(\ )$ is the indicator function that takes the value $\mathbf{1}$ if its argument is true and $\mathbf{0}$ otherwise. The class labels are denoted by $\{0\ ,\ 1\}$. Sometimes, for notational convenience, the class labels $\{-1\ ,\ +1\}$ are used.

1.4.1 Multi–Class Classification

Multi–class classification is a central problem in machine learning. In this case a discrimination among several classes is required. The multi–class classification problem refers to assign each of the observations into one of k classes [11].

The most common approach to multi–class classification, the "One versus All" (OvA) approach, makes direct use of "standard" binary classifiers to train the output labels. The OvA scheme assumes that for each class there exists a single (simple) separator between this class and all the other classes [1]. Another common approach, "All versus All" (AvA), that assumes the existence of a separator between any two classes.

"One versus All" classifiers are usually implemented using a Winner-Take-All (WTA) strategy that associates a real–valued function with each class in order to determine class membership. Specifically, an example belongs to the class which assigns it the highest value (i.e., the "winner") among all classes. While it is known that WTA is an expressive classifier, it has limited expressivity when trained using the OvA assumption since OvA assumes that each class can be easily separated from the rest [18, 33]. An alternative interpretation of WTA is that each example provides an order for the classes (sorted in descending order by the assigned values), where the "winner" is the first class in this ordering. Therefore it's natural to specify the ordering of the classes for an example directly, instead of implicitly through WTA.

1.5 Regression

Regression is the process of guessing or estimating a function from some example input–output pairs with little or no knowledge about the form of the function [11, 20, 25]. This means finding the best prediction of a random variable $Y \in \mathbb{R}^q$ (the output) by another random variable $X \in \mathbb{R}^d$ (the input). Assume $q = 1$, that will simplify the notation. In this framework, a predictor is a function $f : \mathbb{R}^d \to \mathbb{R}$.

A common loss function for regression is the squared error (L_2),

$$L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^{\mathbf{2}}. \tag{1.13}$$

Learning then becomes the problem of finding the function $f(\mathbf{x})$ (*regressor*) that minimizes the risk function

$$R = \int (y - f(\mathbf{x}))^2 \, p(\mathbf{x}, y) \, d\mathbf{x} \, dy, \tag{1.14}$$

using only the training data. This risk functional measures the accuracy of the learning machine's *predictions* of the system output [8, 11].

1.5.1 Linear Least Squares Regression

Linear least squares regression is so far the most widely used modeling method. The terms "regression", "linear regression" or "least squares" are synonymous of linear least square regression. For the input vector $x = (x_1, x_2, \ldots, x_p)$ we want to predict a real–valued output Y.

Linear least squares regression can be used to fit the data with any function of the form

$$f(x) = w_0 + \sum_{j=1}^{p} w_j x_j, \tag{1.15}$$

in which:

1. Each explanatory variable (input value x_i) in the function is multiplied by an unknown parameter (w_i).

2. There is at most one unknown parameter with no corresponding explanatory variable.

3. All the individual terms are summed to produce the final function value.

Since the unknown parameters (w_i) are considered to be variables and the explanatory variables (x_i) are considered to be known coefficients corresponding to those "variables", then the problem becomes a system (usually overdetermined) of linear equations that can be solved in the least squares sense for the values of the unknown parameters [12, 20]. Linear least squares regression also takes its name from the way the estimates of the unknown parameters are computed, The word "linear" here describes the linearity of the model in terms of the w_i not in terms of the explanatory variables x_i. The "*method of least squares*" that is used to obtain parameter estimates was independently developed in the late 1700's and the early 1800's by the mathematicians Karl Friedrich Gauss and Adrien Marie Legendre.

In the least squares method the unknown parameters are estimated by minimizing the sum of the squared deviations between the data and the model. If we have a set of training data $(x_1, y_1) \ldots (x_n, y_n)$ and we want

to estimate the parameters w. Each $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ is a vector of feature measurements for the ith case. *Least squares* estimation method picks the coefficients $w = (w_0, w_1, \ldots, w_p)^T$ that minimize the residual sum of squares [25].

$$\begin{aligned} \mathrm{RSS}(w) &= \sum_{i=1}^{n}(y_i - f(x_i))^2 \\ &= \sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j)^2. \end{aligned} \tag{1.16}$$

For minimizing equation (1.16), we denote by $\mathbf{X}$ the $n \times (p+1)$ matrix with each row an input vector (with a 1 in the first position), and similarly let $\mathbf{y}$ be the n–vector of outputs in the training set. Now we can rewrite the residual sum–of–squares as

$$\mathrm{RSS}(w) = (\mathbf{y} - \mathbf{X}w)^T(\mathbf{y} - \mathbf{X}w). \tag{1.17}$$

This is a quadratic function in the $p+1$ parameters. Differentiating with respect to w we obtain

$$\begin{aligned} \frac{\partial \mathrm{RSS}}{\partial w} &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w) \\ \frac{\partial^2 \mathrm{RSS}}{\partial w \partial w^T} &= -2\mathbf{X}^T\mathbf{X}. \end{aligned} \tag{1.18}$$

Assuming that $\mathbf{X}$ is nonsingular and hence $\mathbf{X^T X}$ is positive definite, we set the first derivative to zero

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}w) = 0, \tag{1.19}$$

to obtain the unique solution

$$\hat{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{1.20}$$

The fitted values at the training inputs are

$$\hat{\mathbf{y}} = \mathbf{X}\hat{w} = \mathbf{X}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{1.21}$$

where $\hat{y}_i = \hat{f}(x_i)$. The matrix $\mathbf{H} = \mathbf{X}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}$ appearing in equation (1.21) is sometimes called the "hat" matrix because it puts the hat on y.

1.5.2 Multiple Linear Regression

The linear model, equation (1.15), with $p > 1$ inputs is called the *multiple linear regression model*. Multiple linear regression attempts to model the relationship between two or more explanatory variables ($\mathbf{X}$) and a response variable by fitting a linear equation to observed data [25]. Each value of the independent variable $\mathbf{X}$ is associated with a value of the dependent variable y. Instead of fitting a line to data, we are now fitting a plane (for 2 independent variables), a space (for 3 independent variables), etc.

1.6 Novelty Detection

An important ability of any signal classification scheme is detecting novel events. Several applications require the classifier to act as a detector rather than to be used as a classifier, that is, the requirement is to detect whether an input is part of the data that the classifier was trained on or it is in fact unknown. There are several important issues related to novelty detection; we can summarize them in terms of the following principles [34].

- *Principle of robustness and trade–off*: A novelty detection method must be capable of robust performance on test data that maximizes the exclusion of novel samples while minimizing the exclusion of known samples. This trade-off should be predictable and under experimental control.
- *Principle of uniform data scaling:* In order to assist novelty detection, it should be possible that all test data and training data after normalization lie within the same range.
- *Principle of parameter minimization*: A novelty detection method should aim to minimize the number of parameters that are in user set.
- *Principle of generalization*: The system should be able to generalize without confusing the generalized information as novel.

- *Principle of adaptability*: A system that recognizes novel samples during test should be able to use this information for retraining.
- *Principle of computational complexity*: A number of novelty detection applications are online and therefore the computational complexity of a novelty detection mechanism should be as low as possible.

Statistical approaches are mostly based on modeling data on the basis of their statistical properties and using these information to estimate whether a test sample come from the same distribution or not. The simplest approach can be based on constructing a density function for data of a known class, and then assuming that data is computing the probability of a test sample belongs to that class. Another simple model is to find the distance of the sample from a class mean and threshold on the basis of how many standard deviations away the sample is. The distance measure itself can be Mahalanobis or some other probabilistic distance [12].

Two main approaches exist in the estimation of the probability density function, *parametric* and *non-parametric methods*. The *parametric approach* assumes that the data comes from a family of known distributions, such as the normal distribution and certain parameters are calculated to fit this distribution. In *non-parametric methods* the overall form of the density function is derived from the data as well as the parameters of the model. As a result non-parametric methods give greater flexibility in general systems. Parametric methods for estimating the probability density function have sometimes limited use because they require extensive a *priori* knowledge of the problem. Non–parametric statistical approaches make no assumption on the form of data distribution and therefore they are more flexible (though more computationally expensive) [20].

There are many applications where novelty detection is very important including signal processing, computer vision, pattern recognition, data mining, and robotics.

Chapter 2

Support Vector Machine and Kernel Methods

In this chapter we introduce the Support Vector Machine (SVM) classification technique, and show how it leads to the formulation of a Quadratic Programming (QP) problem in a number of variables that is equal to the number of data points. We will start by reviewing the classical *Empirical Risk Minimization* (ERM) approach, and by showing how it naturally leads, through the theory of VC bounds, to the idea of *Structural Risk Minimization* (SRM), which is a better induction principle, and how SRM is implemented by SVM. Also we discuss the principles of *kernel* transformations, which provide the main building blocks of Support Vector Machine.

2.1 Support Vector Machines

Support vector machines (SVM) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. This family of classifiers has both abilities: to ***minimize*** the empirical classification error and to ***maximize*** the geometric margin. Hence it is also known as *maximum margin classifier approach* [1].

An important feature of the SVM approach is that the related op-

timization problems are convex because of *Mercer's conditions* on the kernels [12]. Consequently, they haven't local minima. The reduced number of non–zero parameters gives the ability to distinguish between these system and other pattern recognition algorithms, such as neural networks [11].

2.1.1 VC–Dimension

The Vapnik–Chernovenkis (VC) dimension is a scalar value that measures the capacity of a hypothesis space. *Capacity* is a measure of complexity and the expressive power, richness or flexibility of a set of functions.

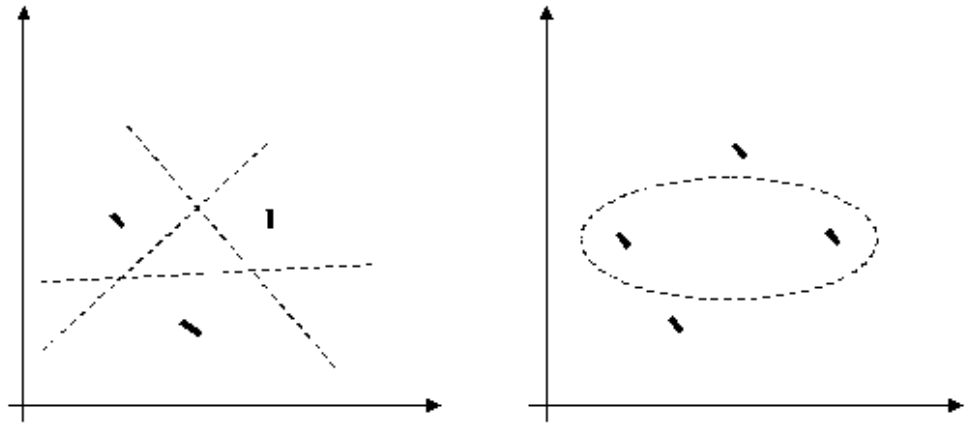

Figure 2.1: VC–Dimension Illustration

To give a simple VC–dimension example as shown in Figure 2.1, there are $2^3 = 8$ ways of assigning 3 points to two classes. For the displayed points in R^2, all 8 possibilities can be realized using one separating hyperplane, in other words, the function class can shatter 3 points, where ***shattering*** is "if n samples can be separated by a set of indicator functions in all 2^n possible ways, then this set of samples is said to be *shattered*

by the set of functions". This would not work if we were given 4 points, no matter how we placed them. Therefore, the VC–dimension of the class of separating hyperplanes in R^2 is 3. In general, the set of linear indicator functions in n dimensional space has a VC–dimension equal to $n+1$ [18, 25, 35].

2.1.2 Empirical Risk Minimization

The task of learning from examples can be formulated in the following way:
Given a set of decision functions

$$\{f_\lambda(\mathbf{x}) : \lambda \in \Lambda\}, \qquad f_\lambda : \mathbb{R}^d \to \{-1, +1\}$$

where Λ is a set of abstract parameters, and a set of examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \qquad \mathbf{x}_i \in \mathbb{R}^d,\ y_i \in \{-1, +1\}$$

drawn from an unknown distribution $P(\mathbf{x}, y)$, we want to find a function f_{λ^*} which provides the smallest possible value for the expected risk:

$$R(\lambda) = \int |f_\lambda(\mathbf{x}) - y|\, P(\mathbf{x}, y)\, d\mathbf{x}\, dy \tag{2.1}$$

the function f_λ are usually called *hypothesis*, and the set $\{f_\lambda(\mathbf{x}) : \lambda \in \Lambda\}$ is called *hypothesis space* and denoted by $\mathcal{H}$. Therefore the measure of how **"good"** the hypothesis in predicting the correct label y for a point $\mathbf{x}$ is called or known as the *expected risk*.

It is not possible in general to compute and then to minimize the expected risk $R(\lambda)$, because the probability distribution $P(\mathbf{x}, y)$ is unknown. However, since $P(\mathbf{x}, y)$ is sampled, it's possible to compute the stochastic approximation of $R(\lambda)$, and this is called *empirical risk*:

$$R_{emp}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} |f_\lambda(\mathbf{x}_i) - y_i| \tag{2.2}$$

The *empirical risk minimization principle* is "if R_{emp} converges to R, the minimum of R_{emp} may converge to the minimum of R". If convergence of the minimum of R_{emp} to the minimum of R does not hold, the

Empirical Risk Minimization principle does not allow us to make any inference based on the data set, and it is therefore said to be not consistent [11].

A typical uniform Vapnik and Chervonenkis bound, which holds with probability $1 - \eta$ has the following form:

$$R(\lambda) \leq R_{emp}(\lambda) + \sqrt{\frac{h(\ln \frac{2n}{h} + 1) - \ln \frac{\eta}{4}}{n}} \qquad \forall \lambda \in \Lambda \tag{2.3}$$

where h is the VC–dimension of f_λ. From this bound it is clear that, in order to achieve small expected risk, that is good generalization performances, both the empirical risk and the ratio between the VC–dimension and the number of data points has to be small. Since the empirical risk is usually a decreasing function of h, it turns out that, for a given number of data points, there is an optimal value of the VC–dimension. The bound of Vapnik and Chervonenkis (equation 2.3) suggests that the Empirical Risk Minimization principle can be replaced by a better induction principle.

2.1.3 Structural Risk Minimization

The technique of Structural Risk Minimization (SRM) has been developed by Vapnik to overcome the problem of choosing an appropriate VC–dimension. It is clear from equation (2.3) that a small value of the empirical risk does not necessarily gives a small value of the expected risk [49]. The principle of *Structural Risk Minimization* is based on the observation that, in order to make the expected risk small, both sides in equation (2.3) should be small, therefore; both the VC–dimension and the empirical risk should be minimized at the same time [2, 25, 46].

In order to implement the SRM principle we need a nested structure of hypothesis spaces as shown in Figure 2.2

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \ldots \subset \mathcal{H}_n \subset \ldots$$

with the property that $h(l) \leq h(l+1)$ where $h(l)$ is the VC–dimension of the set $\mathcal{H}_l$ [5].

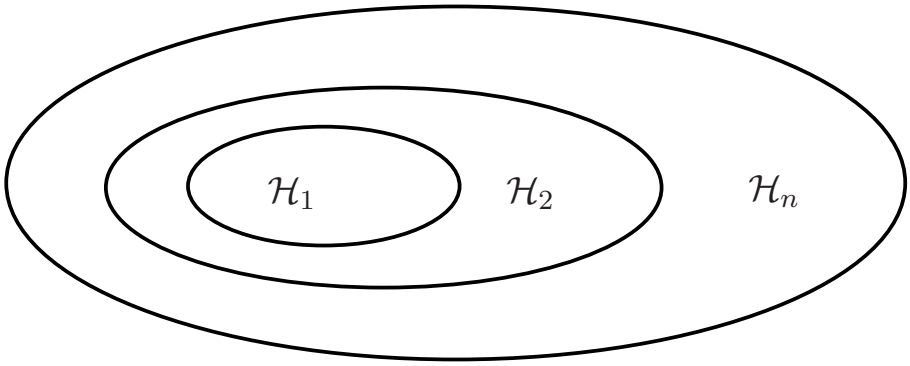

Figure 2.2: Structure of a nested Hypothesis spaces

The SRM principle is clearly well founded mathematically, but it can be difficult to implement for the following two reasons [8]:

1. It is often difficult to find the VC–dimension for a hypothesis space $\mathcal{H}_l$, since not for all models and machines it is known how to calculate this.

2. Even if it is possible to compute h_l (or a bound on it) of $\mathcal{H}_l$ it is not trivial to solve the optimization problem that is given by equation (2.3).

In most cases SRM has to be done by simply training a series of machines, one for each subset, and then choosing $\mathcal{H}_l$ that gives the lowest risk bound. Therefore the implementation of this principle is not easy, because it is important to control the VC–dimension of a learning technique during the training phase. The SVM algorithm achieves this goal, minimizing a bound on the VC–dimension and the number of training errors at the same time [2].

2.1.4 The Optimal Separating Hyperplane

A separating hyperplane is a linear function that has the ability of separating the training data without error as shown in Figure 2.3. Suppose that the training data consists of n samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x} \in \mathbb{R}^d$, $y \in \{+1, -1\}$ that can be separated by a hyperplane decision function

$$D(\mathbf{x}) = \langle \mathbf{w} \,.\, \mathbf{x} \rangle + b, \tag{2.4}$$

with appropriate coefficients $\mathbf{w}$ and b [8, 11, 25, 50]. Notice that the problem is *ill-posed* because the solution may be not unique and then some constraint has to be imposed to the solution to make the problem *well-posed* [13].

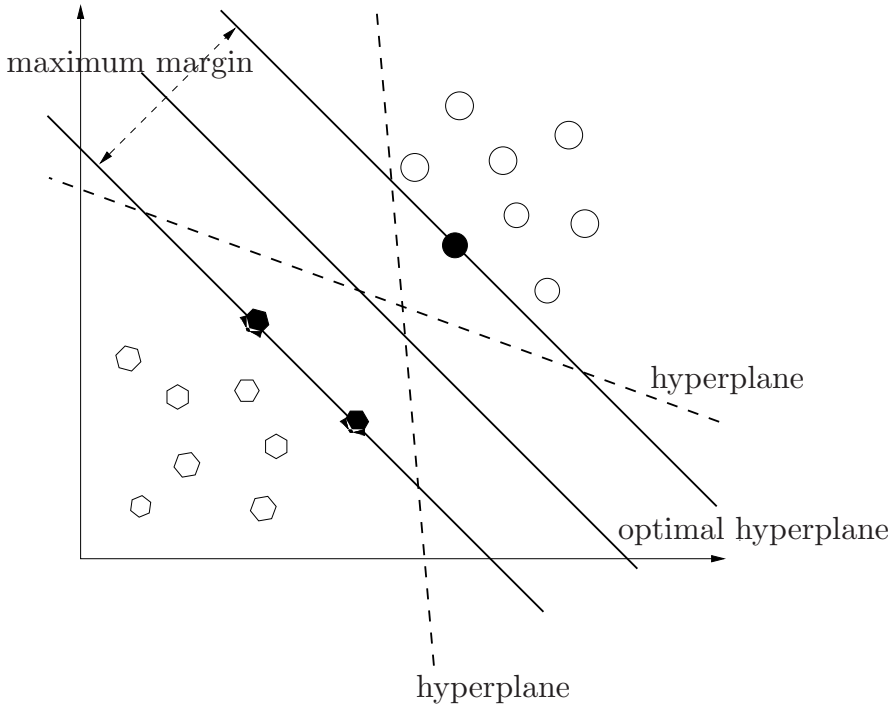

Figure 2.3: Separating hyperplanes in a two–dimensional space. An optimal hyperplane with a maximum margin. The dashed lines are not optimal hyperplanes

A separating hyperplane satisfies the constraints that define the separation of the data samples:

$$\begin{aligned} \langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b &\geq +1 \quad \text{if} \quad y_i = +1 \\ \langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b &\leq -1 \quad \text{if} \quad y_i = -1, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{2.5}$$

Or in more compact form (notation)

$$y_i[\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b] \geq 1, \qquad i = 1, 2, \ldots, n. \tag{2.6}$$

For a given separable training data set, all possible separating hyperplanes can be represented in the form (2.6). The formulation of the separating hyperplanes allows us to solve the classification problem directly. It does not require estimation of density as an intermediate step [1]. When $D(\mathbf{x})$ is equal to $\mathbf{0}$, this hyperplane is called *separating hyperplane* as shown in Figure 2.4.

Let d_i be the signed distance of the point $\mathbf{x}_i$ from the separating hyperplane

$$\mathrm{d}_i = \frac{\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} \tag{2.7}$$

where the symbol $\|\mathbf{w}\|$ denotes the **norm** of $\mathbf{w}$. From this equation follows that

$$\mathrm{d}_i \|\mathbf{w}\| = \langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b$$

and using the constraints (2.6), we have

$$y_i \mathrm{d}_i \|\mathbf{w}\| \geq 1.$$

So for all $\mathbf{x}_i$ the following inequality holds:

$$\frac{1}{\|\mathbf{w}\|} \leq y_i \mathrm{d}_i. \tag{2.8}$$

Notice that $y_i \mathrm{d}_i$ is always positive quantity. Moreover, $1/\|\mathbf{w}\|$ is the lower bound on the distance between the points $\mathbf{x}_i$ and the separating hyperplane $(\mathbf{w}, b)$. The purpose of the "1" in the righthand side of inequality (2.6) for establishing a *one–to–one* correspondence between separating

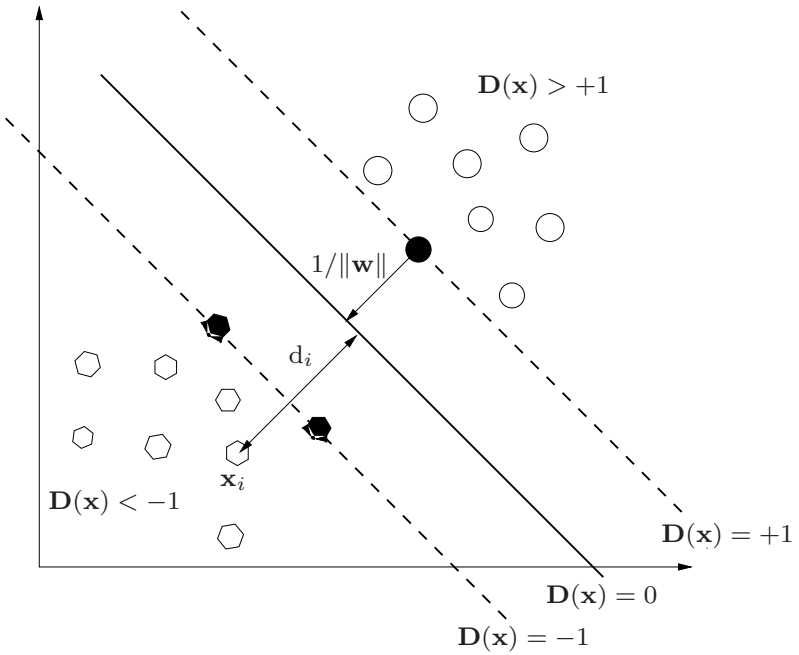

Figure 2.4: Optimal separating hyperplane in a two–dimensional space and the distance between it and any sample

hyperplanes and their parametric representation. This is done through the notion of canonical representation of a separating hyperplane [44].

The optimal hyperplane is given by maximizing the margin, γ, subject to the constraints (2.6). The *margin* is given by [11, 51],

$$\begin{aligned}
\gamma(\mathbf{w}, b) &= \min_{i:y_i=-1} \mathrm{d}_i + \min_{i:y_i=+1} \mathrm{d}_i \\
&= \min_{i:y_i=-1} \frac{\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} + \min_{i:y_i=+1} \frac{\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} \\
&= \frac{1}{\|\mathbf{w}\|} \left(\min_{i:y_i=-1} (\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b) + \min_{i:y_i=+1} (\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b) \right) \\
&= \frac{2}{\|\mathbf{w}\|}.
\end{aligned} \tag{2.9}$$

Thus the optimal hyperplane is the one that minimizes

$$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2. \tag{2.10}$$

Because $\Phi(\mathbf{w})$ is independent of b, changing b moves it in the normal direction to itself, and hence the margin remains unchanged but the hyperplane is no longer optimal in that it will be nearer to one class than the other.

Now we discuss why minimizing (2.10) is equivalent to implementing the SRM principle. Suppose that the following bound holds,

$$\|\mathbf{w}\| \leq A, \tag{2.11}$$

then from (2.8)

$$\mathrm{d} \geq \frac{1}{A}, \tag{2.12}$$

this is equivalent to placing spheres of radius $\frac{1}{A}$ around each data points and consider only the hyperplanes that do not intersect any of the spheres, and intuitively it can be seen in Figure 2.5 how this reduces the possible hyperplanes, and hence the capacity [37].

The VC–dimension, h, of the set of canonical hyperplanes in d–dimensional space is bounded by,

$$h \;\leq\; \min\{\lceil R^2 A^2 \rceil, d\} + 1, \tag{2.13}$$

where R is the radius of the hypersphere enclosing all the data points. Hence minimizing (2.10) is equivalent to minimizing an upper bound on the VC–dimension [8, 44, 50].

2.2 Support Vector Classification

In this section we describe the mathematical derivation of the *Support Vector Machine* (SVM) developed by Vapnik. The Support Vector Machine (SVM) implements the idea of mapping the input vectors $\mathbf{x}$ into the high–dimensional *feature space* F through some nonlinear mapping,

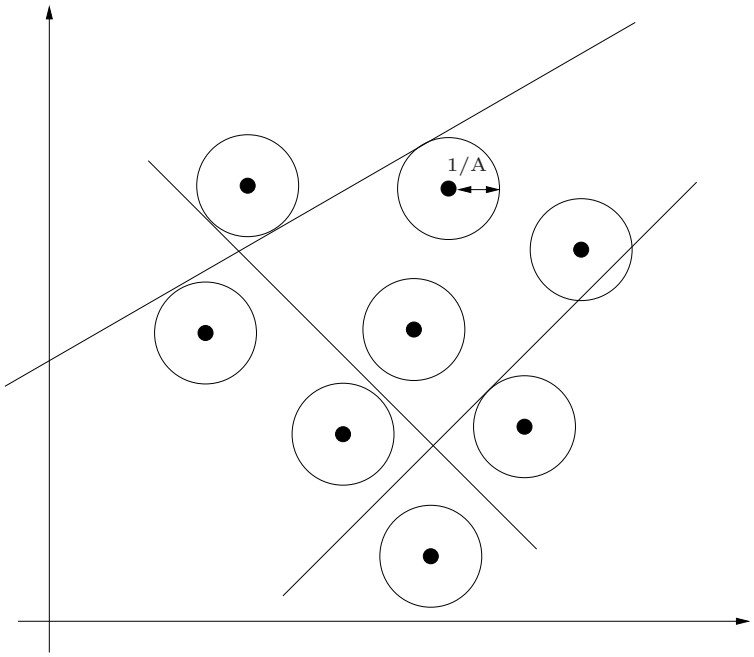

Figure 2.5: Constraining the Canonical Hyperplanes

chosen a priori. In this space, an optimal separating hyperplane is constructed [50].

The technique is introduced by steps: at first consider the simplest case, i.e. a linear classifier and a linearly separable problem; then a linear classifier and a nonseparable problem, and the last one which is the most interesting and useful, nonlinear classifier and nonseparable problem.

2.2.1 Linear Classifier and Linearly Separable Problem

In this section we consider the case where the data set is *linearly separable*, and we want to find the "best" hyperplane that separates the data. In other words finding the pair $(\mathbf{w}, b)$ satisfying the constraints in equation (2.6). The hypothesis space in this case is the set of functions

given by

$$f_{\mathbf{w},b} = \text{sgn}(\langle \mathbf{w} \,.\, \mathbf{x} \rangle + b). \tag{2.14}$$

Because the set of examples are linearly separable, the goal of the SVM is to find, among the *Canonical Hyperplanes* that correctly classify the data, the one at minimum norm. Minimizing $\|\mathbf{w}\|^2$ (in this case of linear separability) is equivalent to finding the separating hyperplane that maximizes the distance between the two convex hulls (of the two classes of training data), measured along the line perpendicular to the hyperplane, as shown in Figure 2.6.

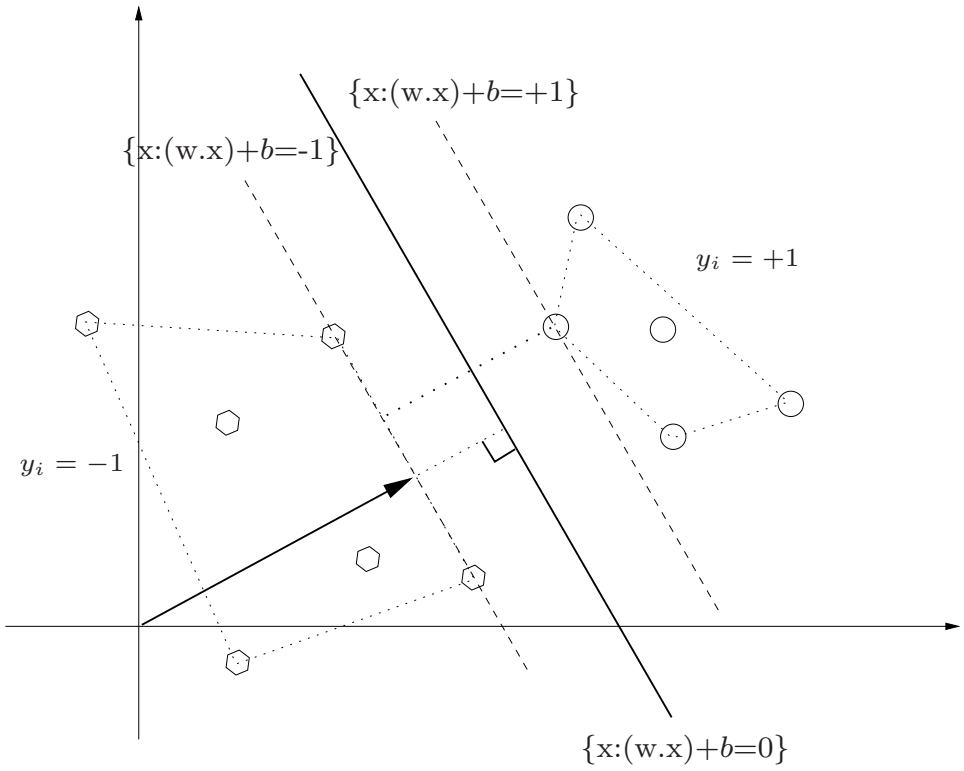

Figure 2.6: The optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted)

The solution to the optimization problem of equation (2.10) under the constraints of equation (2.6) consists of $(d+1)$ parameters. For data of moderate dimension d, this problem can be solved using Quadratic Programming (QP) [1].

For very high–dimensional spaces it is not practical to solve the problem in the present form

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \\ &y_i[\langle \mathbf{w} \ . \ \mathbf{x}_i \rangle + b] \geq 1, \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{2.15}$$

However, this problem can be translated into a dual form that can be better tackled. In case of convexity (as in our problem) solving the dual problem is equivalent to solving the original. For the optimal hyperplane problem it turns out that the size of dual optimization problem scales with the number of samples n and not the dimensionality d. The solution to this problem is given by the saddle point of the Lagrange function (*Lagrangian*),

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \{ \ y_i[\langle \mathbf{w} \ . \ \mathbf{x}_i \rangle + b] - 1 \ \}, \tag{2.16}$$

where α_i are the Lagrange multipliers. The function should be *minimized* with respect to $\mathbf{w}$, b and *maximized* with respect to $\alpha_i \geq 0$. Classical Lagrangian duality enables the primal problem, equation (2.16), to be transformed to its dual problem, which is easier to solve. The dual problem is given by,

$$\max_{\alpha} \ \mathbf{W}(\alpha) = \max_{\alpha} \left(\min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \alpha) \right). \tag{2.17}$$

The Karush–Kuhn–Tucker (KKT) conditions play a central role in both the theory and practice of constrained optimization. For the primal

problem above, the KKT conditions may be stated [11, 15, 49, 50]:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial b} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i, \end{aligned} \tag{2.18}$$

which are satisfied at the minimum of Lagrangian $\mathcal{L}$. The KKT conditions are satisfied at the solution of any constrained optimization problem (convex or not), with any kind of constraints, provided that the intersection of the set of feasible directions with the set of descent directions coincides with the intersection of the set of feasible directions for linearized constraints with the set of descent directions. Furthermore, the problem for SVM is convex (a convex objective function, with constraints which give a convex feasible region), the KKT conditions are *necessary* and *sufficient* for $\mathbf{w}$, b, and α to be a solution. Thus solving the SVM problem is equivalent to finding a solution to the KKT conditions [1, 5, 25]. Hence from (2.16), (2.17) and (2.18), the dual problem is

$$\max_{\alpha} \ \mathbf{W}(\alpha) = \max_{\alpha} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \ . \ \mathbf{x}_j \rangle \right), \tag{2.19}$$

and hence the solution to the problem is given by,

$$\alpha^* = \arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \ . \ \mathbf{x}_j \rangle - \sum_{i=1}^{n} \alpha_i, \tag{2.20}$$

subject to the constraints

$$\begin{aligned} &\sum_{i=1}^{n} y_i \alpha_i = 0, \\ &\alpha_i \geq 0, \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{2.21}$$

Solving (2.20) with the constraints of (2.21) determines the Lagrange

multipliers, and the optimal separating hyperplane is given by,

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{2.22}$$

$$b^* = -\frac{1}{2}\langle \mathbf{w}^* \,.\, (\mathbf{x}_r + \mathbf{x}_s)\rangle, \tag{2.23}$$

where $\mathbf{x}_r$ and $\mathbf{x}_s$ are any *support vector* (points have $\alpha > 0$) from each class satisfying,

$$\alpha_r,\ \alpha_s > 0, \quad y_r = -1, \quad y_s = +1. \tag{2.24}$$

By linearity of the dot product and (2.21), the decision function (hard classifier), as shown in (2.14) can then be written as:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i^* \langle \mathbf{x} \,.\, \mathbf{x}_i\rangle + b^*\right). \tag{2.25}$$

2.2.2 The Soft Margin Hyperplane: Linearly Non-separable Problem (C–SVM)

So far the discussion has been restricted to the case of linearly separable training data. But for the data that can not be separated without error, it would be better to separate the data with a minimal number of errors. This means finding an optimal hyperplane (i.e., with maximal margin) for the training data points that are accurately separated, while it is not possible to satisfy all the constraints in problem (equation 2.15). This corresponds to some data points that fall within the margin or on the wrong side of the decision boundary [8]. Note that the definition of *nonseparable* differs from *misclassification*, which occurs when a data point only falls on the wrong side of the decision boundary as shown in Figure 2.7.

Non negative slack variables $\xi_i,\ i = 1, \ldots, n$, can be introduced to quantify the nonseparable data in the defining condition of the hyperplane [25]. The optimization problem is now posed so as to minimize the classification error as well as minimizing the bound on the VC–dimension

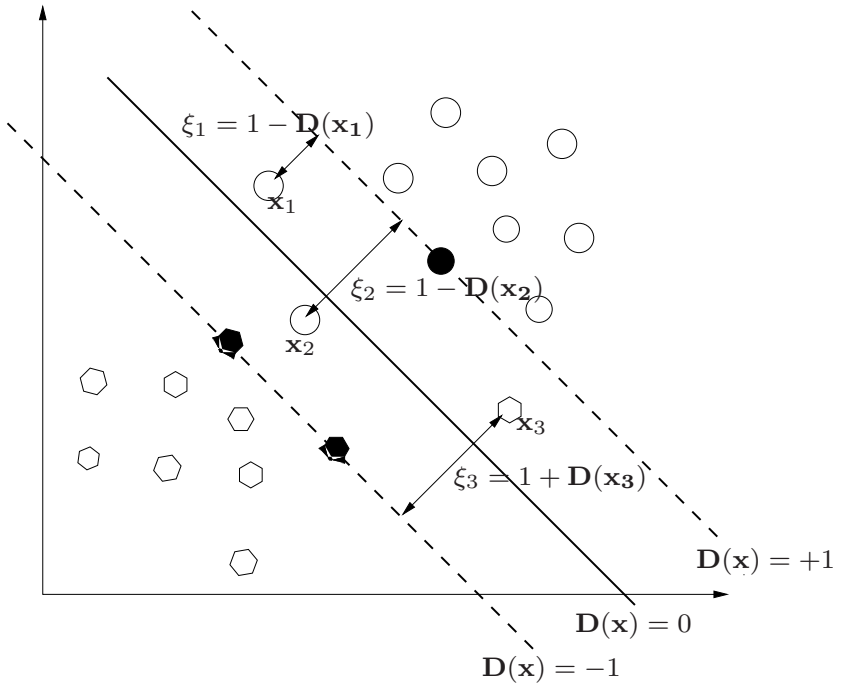

Figure 2.7: Nonseparable case, slack variables are defined that correspond to the deviation from the margin borders.

of the classifier. The constraints of equation (2.6) are modified for the nonseparable case as follows,

$$y_i[\langle \mathbf{w} \,.\, \mathbf{x}_i \rangle + b] \geq 1 - \xi_i, \quad i = 1, 2, \ldots, n. \tag{2.26}$$

For training sample $\mathbf{x}_i$, the slack variable ξ_i is the deviation from the margin border corresponding to the class of y_i (i.e., the margin border defined by $\mathbf{D}(\mathbf{x}_i) = y_i$); as shown in Figure 2.7. According to the definition, slack variables greater than **zero** correspond to *nonseparable* points, while slack variables greater than **one** correspond to *misclassified* samples [41]. Minimizing the number of nonseparated points is a difficult combinatorial optimization problem. Hence we resort to the minimiza-

tion of

$$Q(\xi) = \sum_{i=1}^{n} \xi_i, \tag{2.27}$$

The function (2.27) only approximates the number of nonseparable samples. The hyperplane that minimizes (2.27) subject to the constraint (2.26) and using the structure

$$S_k = \{\langle \mathbf{w} \,.\, \mathbf{x}\rangle + b : \|\mathbf{w}\|^2 \leq c_k\}, \tag{2.28}$$

is called the *soft margin* hyperplane. The optimization problem that finds the soft margin hyperplane is convex, this means that each (local) minimum is also a global minimum [36].

The generalized optimal separating hyperplane for the case of linearly nonseparable classes can be seen as the solution of the following problem:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \quad \Phi(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} \\ &y_i[\langle \mathbf{w} \,.\, \mathbf{x}_i\rangle + b] \geq 1 - \xi_i, \qquad\qquad i = 1,2,\ldots,n \\ &\xi_i \geq 0, \qquad\qquad i = 1,2,\ldots,n \end{aligned} \tag{2.29}$$

where C is a positive constant number which can be regarded as a *regularization parameter*, which determines the trade off between accuracy on the training set (i.e. small $\sum_i \xi_i$) and margin width (i.e. small $\|\mathbf{w}\|^2$). Increasing C means giving more importance to the errors on the training set when the optimal hyperplane is determined. The solution to the optimization problem (2.26) is given by the saddle point (the point that minimizes the functional with respect to $\mathbf{w}$ and b and maximizes it with respect to α) of the Lagrangian

$$\begin{aligned} \mathcal{L}(\mathbf{w},b,\alpha,\xi,\beta) = \; & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\{\, y_i[\langle \mathbf{w} \,.\, \mathbf{x}_i\rangle + b] \\ & -1 \; + \xi_i\} - \sum_{i=1}^{n}\beta_i\xi_i, \end{aligned} \tag{2.30}$$

where α and β are the Lagrange multipliers. The Lagrangian has to be minimized with respect to $\mathbf{w}$, b, and ξ_i, and maximized with respect to $\alpha_i \geq 0$ and $\beta_i \geq 0$.

Classical Lagrangian duality enables the primal problem (equation 2.30) to be transformed into its dual form. The dual problem is given by [1, 11, 25, 50],

$$\max_{\alpha} \mathbf{W}(\alpha, \beta) = \max_{\alpha, \beta} \left(\min_{\mathbf{w}, b, \xi} \mathcal{L}(\mathbf{w}, b, \alpha, \xi, \beta) \right), \tag{2.31}$$

the minimum with respect to $\mathbf{w}$, b and ξ of the Lagrangian, $\mathcal{L}$, is given by,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial b} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\ \frac{\partial \mathcal{L}}{\partial \xi} = 0 \quad &\Rightarrow \quad \alpha_i + \beta_i = C. \end{aligned} \tag{2.32}$$

Hence from equations (2.30), (2.31) and (2.32), the dual problem is,

$$\max_{\alpha} \mathbf{W}(\alpha) = \max_{\alpha} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \, . \, \mathbf{x}_j \rangle \right), \tag{2.33}$$

and hence the solution to the problem is given by,

$$\alpha^* = \arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \, . \, \mathbf{x}_j \rangle - \sum_{i=1}^{n} \alpha_i, \tag{2.34}$$

subject to the constraints

$$\begin{aligned} &0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n \\ &\sum_{i=1}^{n} y_i \alpha_i = 0. \end{aligned} \tag{2.35}$$

This optimization problem differs from the optimization problem for the *separable* case only with the inclusion of a bound C in the constraint (2.35). This parameter introduces additional capacity control within the classifier. C can be directly related to a regularization parameter, but ultimately C must be chosen to reflect the knowledge of the noise on the data [21].

Similarly to the separable case, the points $\mathbf{x}_i$ for which $\alpha_i > 0$ are termed *support vectors*. The main difference is that we must distinguish between the support vectors for which $\alpha_i < C$ and those for which $\alpha_i = C$. In the first case, when $\xi_i = 0$ the support vectors lie at a distance $1/\|\mathbf{w}\|$ from the Optimal Separating Hyperplane (OSH). These support vectors are termed *margin vectors* as we know before. The support vectors for which $\alpha_i = C$, instead, are misclassified points (if $\xi_i > 1$), points correctly classified but closer than $1/\|\mathbf{w}\|$ from the OSH (if $0 < \xi_i \leq 1$), or, in some degenerate cases, even points lying on the margin (if $\xi_i = 0$) [8, 44].

An example of generalized OSH with the relative margin vectors and errors is shown in Figure 2.7. All the points that are not support vectors are correctly classified and lie outside the margin strip.

2.3 Kernel Functions and Nonlinear SVM

Kernel representations offer an alternative solution by projecting the data into a high dimensional feature space to increase the computational power of the linear learning machines. The use of linear machines in the dual representation makes it possible to perform this step explicitly. The advantage of using the machines in the dual representation derives from the fact that in this representation the number of tunable parameters does not depend on the number of attributes being used. By replacing the inner product with an appropriately chosen 'kernel' function, one can implicitly perform a nonlinear mapping to a high dimensional feature space without increasing the number of tunable parameters, provided the kernel computes the inner product of the feature vectors corresponding to the two inputs [11].

2.3.1 Kernel Feature Space

The quantities introduced to describe the data are usually called *attributes*. The task of choosing the most suitable representation is known as *feature selection*. The space X is referred to as the input space, while $F = \{\phi(\mathbf{x}) : \mathbf{x} \in \mathrm{X}\}$ is called the *feature space* [12]. Feature mapping from a two dimensional input space to a two dimensional feature space, is shown in Figure 2.8, where the data cannot be separated by a linear function in the input space, but can be in the feature space [35].

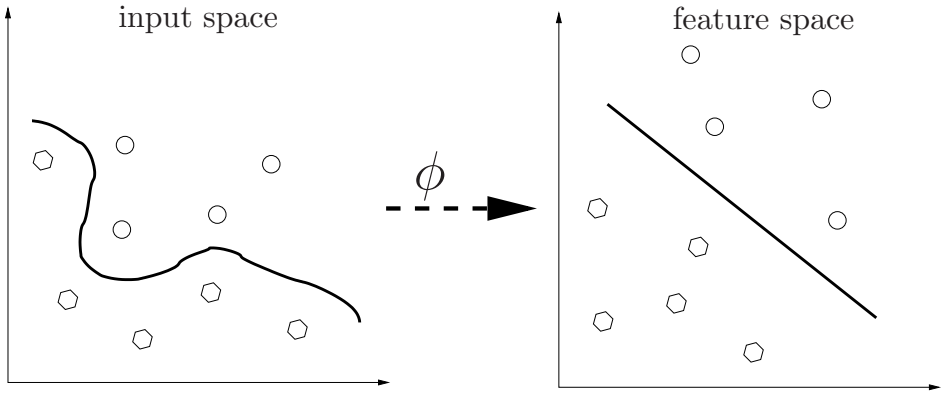

Figure 2.8: A feature map can simplify the classification task.

There are different approaches for feature selection, one of them tries to identify the smallest set of features that still has the important information contained in the original attributes. This is known as *dimensionality reduction*,

$$\mathbf{x} = (x_1, \ldots, x_n) \mapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})), \quad d < n,$$

and can be very beneficial as both computational and generalization performance can degrade as the number of features grows, a phenomenon sometimes referred to as the *curse of dimensionality*. Dimensionality reduction can sometimes be performed by simply removing features corresponding to directions in which the data have low variance, though there

is no guarantee that these features are not important for performing the target classification [11].

Feature selection should be viewed as a part of the learning process itself, and should be automated as much as possible. Therefore for learning nonlinear relations with a linear machine, it's very important to select a set of nonlinear features and rewrite the data in the new representation. This is equivalent to apply a fixed nonlinear mapping of the data to a feature space, in which the linear machine can be used. Hence, the decision function (2.14) becomes

$$f(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{w}_i \phi_i(\mathbf{x}) + b,$$

where $\phi : \mathrm{X} \rightarrow F$ is a nonlinear map from the input space to some feature space, and the number of terms in the summation (m) depends on the dimensionality of the feature space. This mean building non–linear machines consist of two steps:

- **STEP 1:** Transforming the data into a feature space F by using a fixed nonlinear mapping.
- **STEP 2:** Using a linear machine to classify the data in the feature space.

The important property of linear learning machines is that they can be expressed in a dual form. Therefore the hypothesis can be expressed as a linear combination of the training points, so that the decision rule can be evaluated using just inner products between the test point and the training points:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(\mathbf{x}_i) \ . \ \phi(\mathbf{x}) \rangle + b. \tag{2.36}$$

If there is a way to compute the inner product $\alpha_i y_i \langle \phi(\mathbf{x}_i) \ . \ \phi(\mathbf{x}) \rangle$ in feature space directly as a function of the original input points, it becomes possible to combine the two steps needed to build a nonlinear learning machine, and this direct computation method is called *kernel* function

[12]. Through this kernel technique, the value of the kernel function can be computed over a sample set instead of the dot product in a high–dimensional feature space [47]. The *kernel* function will take the symbol K, such that $\mathrm{K}(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}).\phi(\mathbf{z}) \rangle$, and hence, equation (2.36) can be rewritten in this form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \mathrm{K}(\mathbf{x}_i, \mathbf{x}) + b. \tag{2.37}$$

The expansion of the inner product (2.36) in the dual representation allows the construction of decision functions that are nonlinear in the input space. It also makes computationally possible the creation of very high–dimensional feature space, since no direct manipulation is required.

Common classes of basis functions used for learning machines correspond to different choices of kernel functions for computing the inner product. Below are several common classes of multivariate approximating functions and their inner product Kernels:

Polynomials of degree q have inner product kernel

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x} \,.\, \mathbf{z} \rangle + 1)^q, \tag{2.38}$$

which is a popular mapping method for nonlinear modeling

Gaussian Radial Basis Functions of the form,

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = exp(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}), \tag{2.39}$$

where σ defines the width.

Exponential Radial Basis Function of the form

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = exp(-\frac{\|\mathbf{x} - \mathbf{z}\|}{2\sigma^2}), \tag{2.40}$$

produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

Multi–Layer Perceptron (Sigmoid) has a kernel representation

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = \tanh\ (v\langle \mathbf{x} \,.\, \mathbf{z}\rangle + a), \tag{2.41}$$

for parameter values scale, v, and offset, a, selected so that the kernel satisfies Mercer's conditions. Here the Support Vectors (SV) correspond to the first layer and the Lagrange multipliers to the weights.

2.3.2 Non Linear Classifier and Non Separable Problem (C–SVM)

In the case where a linear boundary is unsuitable, the SVM can map the input vector, $\mathbf{x}$, into a high dimensional feature space, $\mathbf{z}$. By choosing a nonlinear mapping a priori, the SVM constructs an optimal separating hyperplane in this higher dimensional space, Figure 2.9 shows the mapping process.

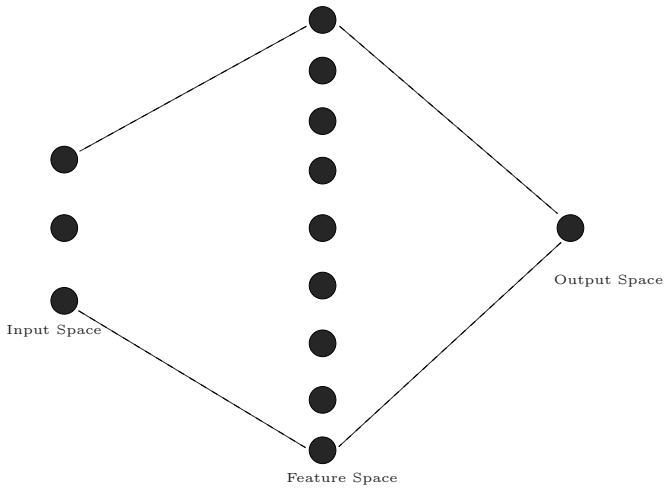

Figure 2.9: Mapping the input space into a high dimensional feature space.

There are some restrictions on the nonlinear mapping that can be employed, but it turns out, surprisingly, that most commonly employed

functions are acceptable. Among acceptable mappings are polynomials, radial basis functions and certain sigmoid functions as explained in the last section. The optimization problem of equation (2.20) becomes,

$$\alpha^* = \arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathrm{K}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{n} \alpha_i, \tag{2.42}$$

where $\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function performing the nonlinear mapping into feature space , and the constraints are unchanged,

$$\begin{gathered} \sum_{i=1}^{n} y_i \alpha_i = 0. \\ 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n. \end{gathered} \tag{2.43}$$

Solving equation (2.42) with constraints equation (2.43) determines the Lagrange multipliers, and a hard classifier implementing the optimal separating hyperplane in the feature space is given by,

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i \in SV_s} y_i \alpha_i \mathrm{K}(\mathbf{x}_i, \mathbf{x}) + b \right), \tag{2.44}$$

where

$$(\mathbf{w}^*.\, \mathbf{x}) = \sum_{i=1}^{n} y_i \alpha_i \mathrm{K}(\mathbf{x}_i, \mathbf{x}).$$

In addition, to compute the threshold b, we consider that for the support vectors $\{\mathbf{x}_i, i \in SV\}$, where SV is the set of index of support vectors, the corresponding $\{\xi_i, i \in SV\}$ are all zeros due to the Kuhn–Tucker dual condition in a dual optimization problem. As a result, we have

$$\sum_{i \in SV} y_i \alpha_i \mathrm{K}(\mathbf{x}_i, \mathbf{x}_j) + b = y_j, \quad j \in SV. \tag{2.45}$$

In order to have robust performance and reduce the effect of computational errors and noises, we can sum (2.45) over all support vectors and

solve for the threshold

$$b = \frac{1}{N_{SV}}[\sum_{j \in SV} y_j - \sum_{i,j \in SV} y_i \alpha_i \mathrm{K}(\mathbf{x}_i, \mathbf{x}_j)], \tag{2.46}$$

where N_{SV} is the number of support vectors. All the results on the linear case can also be directly applied to nonlinear cases through using an appropriate kernel K in place of the Euclidean dot product in feature space. By use of this technique, we can easily compute the dot product in the feature space from the input space [47].

2.4 Support Vector Regression

The SVM approach can be fruitfully applied to regression problems. In this case we suppose that a training set of couples $(y_i, \mathbf{x}_i), i = 1, 2, \ldots, n,\ \mathbf{x}_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}$ is given, corresponding to an unknown function $f : \mathbb{R}^d \to \mathbb{R}$. i.e. $y_i = f(\mathbf{x}_i)$.

Then the SVM approach consists basically in solving

$$\begin{aligned}
&\min \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} (\xi_i^2 + \hat{\xi}_i^2) \\
&\text{subject to} \\
&(\langle \mathbf{w}.\mathbf{x}_i \rangle + b) - y_i \leq \varepsilon + \xi_i, \quad i = 1, 2, \ldots, n, \\
&y_i - (\langle \mathbf{w}.\mathbf{x}_i \rangle + b) \leq \varepsilon + \hat{\xi}_i, \quad i = 1, 2, \ldots, n, \\
&\xi_i, \hat{\xi}_i \geq 0, \qquad\qquad\qquad\quad i = 1, 2, \ldots, n
\end{aligned} \tag{2.47}$$

where we have introduced two slack variables, one for exceeding the target value by more than ε, and the other for being more than ε below the target.

Figure 2.10 shows an example of a one dimensional linear regression function with an ε–insensitive band. The variables ξ measure the cost of the errors on the training points, which have the value equal to zero for all the points inside the band. Figure 2.11 shows a similar situation for a non–linear regression function [11].

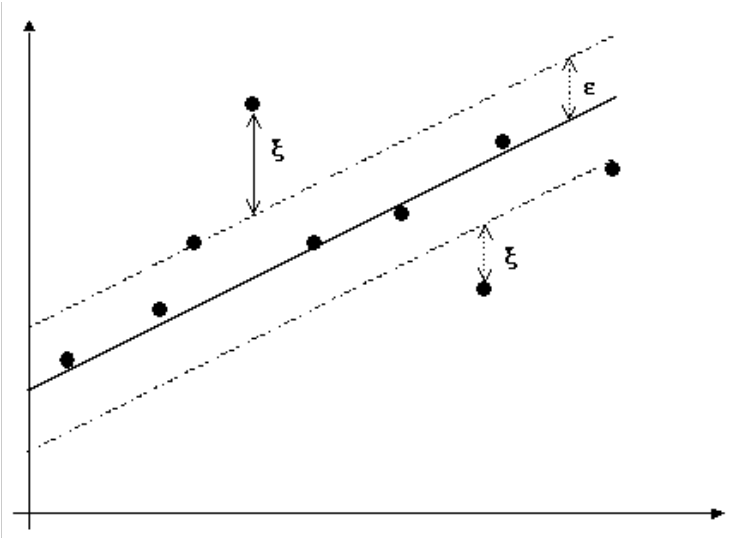

Figure 2.10: The insensitive band for a one dimensional linear regression problem

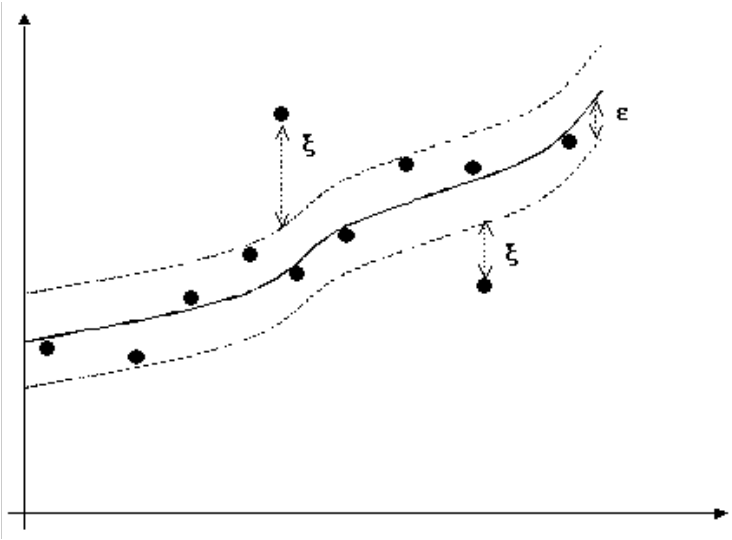

Figure 2.11: The insensitive band for a one dimensional non–linear regression problem

2.5 New Support Vector Machines Algorithms

2.5.1 v–SVM

A new class of support vector algorithms for regression and classification was proposed by Schölkopf. In this class, a parameter v lets one effectively control the number of support vectors. Schölkopf *et al.* [45] proposed v-SVM by incorporating a change from C in the original SVM algorithm with v, where the parameter $v \in [0, 1]$ is an *upper bound* on the fraction of training errors and a *lower bound* of the fraction of support vectors [9, 23].

The optimization problem is:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \quad \Phi(\mathbf{w}, \xi, \rho) = \frac{1}{2}\|\mathbf{w}\|^2 - v\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ &\text{subject to} \\ &y_i[\langle \mathbf{w} \,.\, \mathbf{x}_i\rangle + b] \geq \rho - \xi_i, \qquad i = 1, \ldots, n \\ &\xi_i \geq 0, \qquad i = 1, \ldots, n \\ &\rho \geq 0. \end{aligned} \tag{2.48}$$

This changes the width of the margin from $2/\|\mathbf{w}\|$ as in C–SVM to $2\rho/\|\mathbf{w}\|$, which is to be maximized while minimizing the margin errors, and ρ is the position of the margin.

The primal Lagrangian formulation is:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \alpha, \xi, \beta, \delta, \rho) \quad &= \frac{1}{2}\|\mathbf{w}\|^2 - v\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ &- \sum_{i=1}^{n}\alpha_i\{\, y_i[\langle \mathbf{w} \,.\, \mathbf{x}_i\rangle + b] - \rho \; + \xi_i\} \\ &- \sum_{i=1}^{n}\beta_i\xi_i - \delta\rho, \end{aligned} \tag{2.49}$$

with Lagrange multipliers $\alpha_i, \beta_i, \delta \geq 0$. This function has to be minimized with respect to the primal variables $\mathbf{w}, \xi, b, \rho$ and maximized with

respect to the dual variables $\alpha, \beta,$ and δ. At the optimal solution, one has the following saddle point equations,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i. \tag{2.50}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \;\Rightarrow\; \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.51}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \;\Rightarrow\; \frac{1}{n} - \alpha_i - \beta_i = 0. \tag{2.52}$$

$$\frac{\partial \mathcal{L}}{\partial \rho} = 0 \;\Rightarrow\; \sum_{i=1}^{n} \alpha_i - \delta = v. \tag{2.53}$$

The following dual optimal problem is obtained by substituting equations (2.50 – 2.53) into equation (2.49), using $\alpha_i, \beta_i, \delta \geq 0$, and incorporating *Kernels* (K(.,.)) for dot products leaves us with the following quadratic optimization problem:

$$\max \mathbf{W}(\alpha) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) \tag{2.54}$$

subject to

$$0 \leq \alpha_i \leq 1/n. \tag{2.55}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.56}$$

$$\sum_{i=1}^{n} \alpha_i \geq v. \tag{2.57}$$

With a trained **SVM**, support vectors (SVs) are data vectors with $\alpha_i > 0$, and bounded support vectors (BSVs) are support vectors with $\alpha_i = 1/n$ and $\xi_i > 0$ [9]. The resulting decision function is

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \mathrm{K}(\mathrm{x}_i, \mathrm{x}) + b \right). \tag{2.58}$$

Compared to the original dual of Vapnik [50], there are two differences. **First,** there is an additional constraint, equation (2.57). **Second,** the linear term $\sum_{i=1}^{n} \alpha_i$ no longer appears in the objective function (2.54), this has an interesting consequence equation (2.54) is now quadratically homogeneous in α.

2.5.2 SVM^{light}

SVM^{light} is an implementation of an SVM learner which addresses the problem of large datasets. The decomposition technique breaks the large Quadratic Program (QP) problem into smaller QP sub-problems. Osuna *et al.* [38] published an improved algorithm for training SVM classifiers. The problem, they state, is that previous algorithms assumed small numbers of support vectors.

They present a decomposition algorithm that guarantees global optimality, and can be used to train SVMs over very large data sets. The main idea behind the decomposition is the iterative solution of sub–problems and the evaluation of optimality conditions which are used both to generate improved iterative values, and also establish the stopping criteria for the algorithm [38]. By defining

$$\mathbf{W}(\alpha) = \left(-\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathrm{K}(\mathbf{x}_i \, , \, \mathbf{x}_j) \right), \tag{2.59}$$

the problem may be written in matrix notation, incorporating non–negativity of $\boldsymbol{\alpha}$ and constraint equation (2.42), as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathbf{W}(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}.\mathbf{1} + \frac{1}{2}\, \boldsymbol{\alpha}^T D \boldsymbol{\alpha} \\ &\text{subject to} \\ &\boldsymbol{\alpha} \,.\, \mathbf{y} = 0 \qquad (\mu) \\ &\boldsymbol{\alpha} - C\mathbf{1} \leq \mathbf{0} \qquad (\boldsymbol{\Upsilon}) \\ &-\boldsymbol{\alpha} \leq \mathbf{0} \qquad (\boldsymbol{\Pi}) \end{aligned} \tag{2.60}$$

where $\mathbf{y} = (y_1, \ldots, y_n)$, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, and D is a symmetric, positive semi–definite $n \times n$ matrix with elements $D_{ij} = y_i y_j \mathrm{K}(\mathbf{x}_i\,, \mathbf{x}_j)$, while $\mu, \boldsymbol{\Upsilon} = (v_1, \ldots, v_n)$, and $\boldsymbol{\Pi} = (\pi_1, \ldots, \pi_n)$ are the associated Kuhn–Tuker multipliers. Since D is a positive semi–definite matrix and the constraints (2.60) are linear, the Kuhn–Tuker conditions are necessary and sufficient for optimality, and they are:

$$\begin{array}{ll} \nabla \mathbf{W}(\boldsymbol{\alpha}) + \boldsymbol{\Upsilon} - \boldsymbol{\Pi} + \mu \mathbf{y} & = \mathbf{0} \\ \boldsymbol{\Upsilon}.(\boldsymbol{\alpha} - C\mathbf{1}) & = 0 \\ \boldsymbol{\Pi}\ .\ \boldsymbol{\alpha} & = 0 \\ & \\ \boldsymbol{\Upsilon} & \geq \mathbf{0} \\ \boldsymbol{\Pi} & \geq \mathbf{0} \\ \boldsymbol{\alpha}\ .\ \mathbf{y} & = 0 \\ \boldsymbol{\alpha} - C\mathbf{1} & \leq \mathbf{0} \\ -\boldsymbol{\alpha} & \leq \mathbf{0}. \end{array} \tag{2.61}$$

The Decomposition Algorithm proposed by Osuna *et al.* [38] defined a fixed–size working set called B, such that $|B| \leq n$, and it is big enough to contain all support vectors ($\alpha_i > 0$), but small enough such that the computer can handle it and optimize it using some solver. Then the **decomposition algorithm** can be stated as follows:

1. Arbitrary choose $|B|$ points from the data set.
2. Solve the subproblem defined by the variables in B.
3. If an appropriate stopping condition is not satisfied, exchange one index between B and $B - N$, where $N \triangleq \{1, 2, \ldots, n\}$, and return to step 2.

The main advantage of this decomposition is that it suggests algorithms with memory requirements *linear* in the number of training examples and *linear* in the number of SVs. One potential disadvantage is that these algorithms may need a long training time [28]. Modifications and improvements of the standard SVM^{light} approach are in [39].

Chapter 3

Application of SVM in the Design of Multi–Sensors Systems

The fundamental part of any sensor array system is data analysis and pattern recognition. Several supervised pattern recognition techniques have been explored for the analysis of sensor array data, for example, parametric methods (linear or quadratic), nearest neighbors, and neural networks (multilayer perceptrons and radial basis functions) [24]. In the last decade, a new classification (and regression) technique, called support vector machine (SVM), has been successfully applied to a number of problems ranging from face identification and text categorization to bioinformatics and data mining.

An array of gas sensors constitutes what is called an *Electronic Nose* whose responses establish the odor pattern. A single sensor in the array should not be highly specific in its responses but should respond to a broad range of compounds, such that different patterns are expected to be related to different odors. To achieve higher recognition rate, several sensors with different selectivity patterns are used and pattern recognition techniques must be coupled with the sensor array [19, 31].

In this chapter a survey on the applications of Multi-Sensor arrays using SVM as a classifier or pattern recognition tool will be presented.

3.1 Applications of Electronic Nose

The main idea of electronic nose is to use many sensors of different types and process data in a way that resembles data processing by living brains. Electronic noses, or *ENoses*, are less a sensor or instrument and more a *measurement strategy* [19]. Electronic noses have become popular and combine advanced sensors and sensor array strategies with chemometrics techniques to produce a broad range of intermediate instruments and analyzers [16].

Early *ENoses* tried to duplicate the behavior and capability of human odor sensing. They combined different sensor types to represent the different cell tissues in the nasal cavity and they took the approach of detecting an odor as a collection of individual chemicals. Its functions using an array of broadly tuned chemical sensors, i.e. sensors that interact with a broad range of chemicals with varying strengths. Consequently, an incoming analyte stimulates many of the sensors in the array, and elicits a characteristic response pattern, these patterns are then further analyzed for the benefit of the specific application [6]. The name "odor sensor" is used instead of "gas sensor" whenever its sensitivity approaches that of a human. Odor and fragrance sensors find applications in forensic science, quality assurance in the cosmetic and food industry, environmental control, and so forth [19].

Polikar *et al.* (2001) described a gas sensing system for detecting and identifying volatile organic compounds (VOCs), and they discussed the unique problems associated with the separability of signal patterns obtained by using such a system. Also they presented solutions for enhancing the separability of VOC patterns to enable classification. They introduced a new incremental learning algorithm that allowed new odorants to be learned [43]. ENose systems for detection and identification of volatile organic compounds (VOCs), an important class of chemicals that can readily evaporated at room temperature, have gained considerable attention, since VOCs are encountered in many gas sensing applications. A major problem in VOC identification is the substantial similarity of patterns obtained for different VOCs, a phenomenon attributed to low

selectivity of the sensing system.

They used an array of six 9 MHz QCMs (Quartz Crystal Microbalances) in this study. The QCMs were first coated with chromium/gold, which served as electrodes. Each QCM was then coated with a different polymer to sorb the VOCs of interest. The QCMs were mounted in a sealed test fixture and exposed to VOC vapors. The vapor generation system consisted of calibrated mass flow controllers, conventional gas bubblers containing the VOCs, and a pair of three–way switchable valves leading into the test fixture. The vapor at various concentrations was generated by flowing a carrier gas, typically dry nitrogen, through the bubbler and further diluting the vapor with nitrogen to obtain the desired concentration. The switchable valves were computer controlled to automatically expose the sensor array to various concentrations of VOCs.

The classification technique they used was LEARN++ which is based on generating a number of classifiers using different distributions of the training data, then was combining the outputs of these classifiers using a weighted majority voting scheme. The algorithm kept track of the performance of each classifier on each training instance, and generated a new training subset based on the performances of all previous classifiers. In particular, a weight was given to each instance, and this weight was increased if the instance was misclassified. A new classifier was then trained with the new training set, added to the pool of classifiers generated earlier, and the combined classification performance of all classifiers was then used to determine the next training set. Multiple classifiers were generated for any given database, and as new databases became available, new classifiers were added. The final classification for each instance was then based on the weighted majority voting of all classifiers. In particular, nonlinear cluster translation was introduced for increasing pattern separability, and LEARN++ for incremental learning. They noted that both algorithms were also tested for identification of a larger number of VOCs, individual components of mixtures of VOCs, as well as non gas–sensing applications, and very promising results were obtained [43].

Penza *et al.* (2002) have developed a surface acoustic wave (SAW) multisensor array with five acoustic sensing elements configured as two–port resonator 433.92 MHz oscillators and a reference SAW element to recognize different individual components and determine their concentrations in a binary mixture of volatile organic compounds (VOCs) such as methanol and acetone, in the ranges 15-130 and 50-250 ppm, respectively. They used the relative frequency change as the output signal of the SAW multisensor array with an artificial neural network (ANN), a recognition system has been realized for the identification and quantification of tested VOCs. The features of the SAW multisensor array exposed to a binary component organic mixture of methanol and acetone have been extracted from the output signals of five SAW sensors by pattern recognition (PARC) techniques, such as principal component analysis (PCA). An ANN classifier identification performance of 70 and 80% for methanol and acetone was achieved by using the normalized responses of four selected SAW sensors or the first three principal components scores from the normalized responses of four selected SAW sensors, respectively, and the first two principal components scores from the same set of normalized responses of the four selected SAW sensors. This recognition rate has been obtained because the eigenvectors (principal components) of the original data set are uncorrelated and the latter principal components are found to be noisy in the classification process by the ANN classifier [42].

The prediction of the individual vapor concentrations has been tackled with PCA for features extraction and by using the first two principal components scores as inputs to a feed–forward MLP consisting of a gating network, which decides which of the three specific subnets should be used to determine the output concentration: the first subnet for methanol only, the second subnet for acetone only and the third subnet for methanol and acetone in the binary mixture. They achieved 0.941 and 0.932 correlation coefficients for the predicted versus real concentrations of methanol and acetone, respectively, as individual components in a binary mixture.

Pardo and Sberveglieri (2002) presented the **Pico–1** electronic nose which is based on thin–film semiconductor sensors and an application to the analysis of two groups of seven coffees each. Cups of coffee were

also analyzed by two panels of trained judges who assessed quantitative descriptors and a global index (called *Hedonic Index*, HI) characterizing the sensorial appeal of the coffee.

Two tasks were performed by **Pico–1** [40]. First, for each group, they performed the classification of the seven different coffee types using PCA and *multilayer perceptrons* for the data analysis. Classification rates were above 90%. Secondly, the panel test descriptors were predicted starting from the measurements performed with **Pico–1**, which was an electronic nose consisting of five SnO_2–based thin–films sensors. Two of them were pure SnO_2 sensors; one was catalyzed with gold, one with palladium, and one with platinum. A thin layer of noble metals was deposited as catalyst on three sensors to improve sensitivity and selectivity.

Carmel *et al.* (2003) proposed an algorithm that was able to identify chemicals independently of their concentrations, as well as to quantify their concentrations, even if the particular concentration in question was not present during the training phase. The algorithm was distinct for its intuitive approach, and utilized in a very straightforward way the special properties of a multisensor system. Besides its concentration prediction capabilities, the algorithm was equipped with a reject option, and was thus able to tell when a sample was not one of the chemicals upon which it was trained. The algorithm was inspired by the work of Hopfield, who proposed a similar one to explain how olfactory data analysis was carried out in the brain.

They assumed their ENose consisted of m different sensors, and supposed that it could be exposed to any of n different analytes, where the concentration of the ith analyte c_i could be anywhere in the range $c_i^{min} \leq c_i \leq c_i^{max}$ (where c_i^{min} and c_i^{max} were odor specific predetermined constants). They constructed the functions $f_{ij}(c_j)$ using the data collected in the training data set. Each candidate j was measured in six different concentrations that were denoted by $c_j^1, \ldots, c_j^6$. The corresponding responses of the ith sensor, denoted $r_{ij}^1, \ldots, r_{ij}^6$, were calculated as the average of the repetitions. The six values $(c_j^1, r_{ij}^1), \ldots, (c_j^6, r_{ij}^6)$ were the data from which they had to evaluate $f_{ij}(c_j)$. They did this by using the piecewise cubic spline interpolation of Matlab. In some cases, the

concentration dependency of the response was very close to linear. In such cases, $f_{ij}(c_j)$ could be well approximated by linear regression on the data.

As they claimed, their algorithm was not limited to ENose systems; it could, in fact, be applied to any multisensor system in which the sensors respond monotonically to the stimuli. Their experiments demonstrated that 16 sensors were enough for obtaining good results. As the number of sensors in the system grows, performance was expected to be improved. The robustness of the algorithm was also expected to improve with the increase in the number of sensors [6].

From our point of view, the corresponding responses of the ith sensor were calculated as the average of the repetitions do not give a good result.

Lozano *et al.* (2006) presented an application of an ENose for the identification of typical aromatic compounds in white and red wines. The descriptors of these compounds were fruity, floral, herbaceous, vegetative, spicy, smoky, and microbiological, and they were responsible for the usual aromas in wines. Some of the measured aromas were pear, apple, peach, coconut, rose, geranium, cut green grass, mint, vanilla, clove, almond, toast, wood, and butter. Principal component analysis (PCA) and linear discriminant analysis (LDA) showed that datasets of these groups of compounds were clearly separated, and a comparison among several types of artificial neural networks (ANN) had been also performed. The results confirmed that the system had good performance in the classification of typical red and white wine aromas. The multivariate response of the sensors with broad and partially overlapping selectivities could be utilized as an electronic fingerprint to characterize a wide range of odors or volatile compounds by means of pattern recognition techniques [32].

A total of 16 aromas have been analyzed: eight in white wine and eight in red wine. The measured aromas were the most common ones in white and red wines. The chemical compounds responsible of these aromas were dissolved in the same wine at concentrations from 2-8$\times$ the threshold concentration that humans can smell. An ENose based on a tin oxide-array has been used for measuring wine samples using headspace analysis. The sensor array was prepared by **RF** (radio fre-

quency) sputtering onto alumina substrate. The array was formed by 16 thin film sensors with thicknesses between 200 and 800 nm. Some sensors were doped with chromium and indium either as surface or intermediate layer. The operating temperature of the sensors is controlled at 250 °C with a PID regulator (Proportional-Integral-Derivative regulator, is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly). The array was placed in a stainless–steel cell with a heater and a thermocouple.

The data collected were analyzed by means of pattern recognition techniques using a commercial software package (Matlab 6.1) for linear methods, such as principal component analysis (PCA), which is a signal representation technique that applies a linear transformation to the data and results in a new space of variables called principal components, PCA reduces the dimensionality of feature space by restricting attention to those directions along which the scatter of the cloud data points is greatest [14]. Usually, the first two components carry most of the information of the old variables. Linear discriminant analysis (LDA), is a signal–classification technique that directly maximizes the class separatibility, and generating projections where the examples of each class form a compact clusters [14].

The perceptron network consists of a single layer of nine perceptron neurons connected to the inputs through a set of weights. The perceptron learning rule is capable of training only a single layer. This restriction imposes limitations on the computation a perceptron can perform. The weights of the neurons can be adapted on an iteration–by–iteration basis. For the adaptation, they used an error–correction rule known as the perceptron convergence algorithm. The backpropagation network architecture was formed by three layers: the input layer had 16 neurons corresponding to the 16 sensors, a variable number in the hidden layer, and nine neurons in the output layer, the same number of existing classes. A probabilistic neural network PNN composed of three layers, with radial basis transfer functions in the hidden layer and a competitive one in the

output, was used for classification purposes [24].

Leave–one–out (LOO) cross validation was applied to check the performance of the network. A nonlinear pattern–recognition system based on perceptron, backpropagation, and radial–basis neural networks had been trained for the identification of aromatic compounds added to wine. The best classification was obtained by backpropagation with 100% classification success followed by probabilistic neural networks with 99%, although the time of training was much lower than in that of the probabilistic network. The worse behavior of the perceptron network was due to the limitation of being a one–layer network [32].

Zhou *et al.* (2006) described a modified nonlinear least–squares based algorithm developed to analyze data taken by the ENose, and its performance for the identification and quantification of single gases and binary mixtures of twelve target analytes in clean air. The sensor array in the ENose, which was recently built at JPL (Jet Propulsion Laboratory) and demonstrated aboard NASAs space shuttle flight STS–95, consists of 32 conductometric sensors made from insulating polymer films loaded with carbon. In its current design, it has the capability to detect 10 common contaminants which may be released into the recirculated breathing air of the space shuttle or space station from a spill or a leak. The device could be detected changes in humidity and a marker analyte. The ENose was intended to fill the gap between an alarm which sounds at the presence of chemical compounds but with little or no ability to distinguish among them, and an analytical instrument which can distinguish all compounds present but with no real–time or continuous event monitoring ability [53].

The response pattern requires software analysis to identify the compounds and concentrations causing the response. The primary goal in analysis software development was to identify events of single or mixed gases from the 10 target compounds plus humidity changes and marker analyte with at least 80% accuracy (fewer than 20% false positives and false negatives) in both identification and quantification, where accurate quantification is defined as being $\pm 50\%$ of the known concentration. Correct identification of the compound causing a test event with quantification outside the $\pm 50\%$ range is considered to be a false positive.

Failure to detect an event is a false negative.

They suggested that the linear algebra (LA) based method requires that the sensors follow linearity and additive linearity (superposition) properties, i.e. the response to a mixture of compounds is a linear combination of the response to the individual compounds. They Compared the methods that use concentration–normalized sensor response to identify unknown gas(es) first, by using PCA, for example, and then determine concentration(s) from previously calibrated data by a second analysis, the LA based method effectively has combined the two steps of identification and quantification into one. This combination helped to facilitate the automation of data analysis without human intervention.

A modified nonlinear least–squares based algorithm was developed as part of the JPL ENose program to identify and quantify single gases and mixtures of common air contaminants. The development of the nonlinear least squares (NLS) based method followed their early success with a linear algebra based method and later understanding of its limitation in nonlinear cases. It enabled them to find a good solution instead of an exact one from noisy sensor response patterns. For lab–controlled testing, the algorithm achieved a success rate of about 85% for single gases in air and a moderate 60% success rate for mixtures of two compounds. Improvement of the NLS algorithm analysis speed was also desirable, not only for the purpose of realtime analysis, but also to accommodate the expanded target compound list and polymer set expected for the future generation of the ENose. Since the NLS algorithm is heavy with matrix operations, which largely determines the entire data analysis speed, increased size of the system characteristic matrix will slow analysis speed exponentially. One way to increase speed is to reduce the size of the matrix dynamically by incorporating sensors characteristic response information, such as known negative or no responses of particular sensor to certain gas compound.

From our point of view, suggesting that the response of the sensors to a mixture of compounds is a linear combination of the response to the individual compounds is far away from the real response.

Huang and Leung (2007) developed a polynomial–based optimization method to perform classification and estimation simultaneously to improve the intelligence of an ENose. The proposed method employs a parametric polynomial with user–defined order to describe sensor characteristics. Classification and concentration estimation was formulated as a standard convex optimization problem. They solved the convex optimization either by a typical gradient descent method (Gauss-Newton optimization) for an unconstrained case or a nonlinear least square (NLS) trust–region method for a constrained case [26].

They proposed a gradient descent method to simultaneously detect and estimate the vapor concentrations, and could reach a well balance between computational complexity and accuracy. Therefore, they considered that it practically improves the intelligence of the ENose since continuous monitoring with high accuracy becomes feasible. They formulated the detection and estimation problem as a convex optimization problem. It should be noted that the solution to this convex optimization problem will assure the global minimum. The main advantages of their proposed method are the flexibility and significant reduced computation cost as well as simple implementation; moreover, the global minimum of the optimization is readily achieved.

3.2 Electronic Nose Using SVM as Classification Tool

Over the past few years, the artificial neural networks (ANNs), one of the branches in artificial intelligence technology, have been used to signals recognition of electronic nose and some encouraging results have been achieved [40]. Multilayer perceptrons (MLPs; back propagation ANNs) and radial basis function (RBF) artificial neural networks are most commonly used in signals recognition of electronic nose. Although the MLPs are developed in signals recognition of electronic nose, some inherent drawbacks, e.g., the multiple local minima problem, the choice of the number of hidden units and the danger of over fitting, etc., would make it difficult to put the networks into practice. In the RBF ANNs, the

choice of appropriate set of RBF centers and of the number of hidden units for effective learning, still remains as a problem.

Recently, a new tool from the artificial intelligence field called Support Vector Machine [50] has gained popularity in the machine learning community. It has been applied successfully to classification tasks such as pattern recognition, optical character recognition (OCR) and more recently also to regression and time series prediction. Mathematically, SVMs are a range of classification and regression algorithms that have been formulated from the principles of statistical learning theory. Compared with traditional ANNs, training in SVMs is very robust due to their quadratic objective functions. It is useful to explore this new technology in gas identification area, with the hope that it could overcome some of the problems in ANN and may perform much better than the traditional linear models.

Distante *et al.* (2003) have adopted the SVM classifier for odor recognition with complex patterns. The objective of this classifier was to find optimal hyperplanes for separating clusters in the non–linearly separable context. The leave–one–out procedure has been used for all classifiers, in order to find the near–optimal SVM parameter and both to reduce the generalization error and to avoid outliers.

They used an SVM having a second degree polynomial kernel function. Moreover, the optimal regularization parameter C of the SVM was found out experimentally by minimizing the leave–one–out error over the training set, which provided an estimate of the generalization performances of the final classifier. Each machine had been trained to solve a two–class problem, in order to find experimentally the regularization parameter C. The multi–class problem for a two–class trained machine (SVM) was carried out with leave–one–out procedure and they got an evaluation of the winning machine for each test data point [13].

As a comparison, they made two further feed–forward neural networks had been trained with error back propagation and radial basis function. Comparing the prediction error, they found that back propagation method gave poorer results with more than 40% error, followed by the RBF network with 15%, then SVM with 4.5%.

Uluyol *et al.* (2003) presented a new smellprint derived from Cyranose 320 electronic nose. The new smellprint was based on the initial reactions of the chemoresistance rather than the bulk relative resistance change. Various combinations of the two smellprints including their projections to a small number of principal components were analyzed. Cyranose 320 uses polymer composite sensors that swell or contract when exposed to a vapor–phase analyte. The response from the chemoresistance was then measured as a bulk relative resistance change (Rmax/Rbaseline), and was used to form a smellprint. This 32–element vector was then used for detecting chemicals. The chemical signatures formed based on the initial sensor responses were unique and consistent. The results showed that the classification performance achieved using only the initial response signatures was comparable to or better than the one obtained with the original smell print. When the two sets are combined, significant improvements in classification performance were noted. For classification, the method of support vector machine was employed. This kernel–based technique, was shown to be more powerful and robust than the ones included with the Cyranose 320 system [48].

The binary Support Vector Machine classification results were altered through two different mechanisms: set threshold on the total vote, and winner–take–all method. Because of the small number of exposures are used (8 for each of the 5 compounds), the classification accuracy was determined through the leave–one–out procedure. The best results were obtained when the binary support vector machine method was applied on a small set of features obtained through principal component analysis and the outcome was determined through the winner–take–all approach. From our point of view the results will not be good when the set of features are big.

Wang *et al.* (2005) used the SVM as a classifier to recognize the gas category. The method could classify the complicated patterns and achieved higher recognition rate at reasonably small size of training sample set and could overcome the disadvantages of the artificial neural networks (ANN). In their implementation three different gases were considered, ethanol, gasoline and acetone, three sensors and two SVMs were

used for classification. First, one class was ethanol and the other class is not ethanol (gasoline or acetone). Then, when the gas was known not to be ethanol a second SVM was used to differentiate between gasoline and acetone. For the SVM, there are two parameters while using RBF kernels: γ and C. Good parameter selection is crucial to the SVM success, but the parameter settings are difficult to select. It is not known beforehand which γ and C are the best for one problem [52]. Consequently it is common practice to estimate a range of potential settings and to try several parameter values. Also, selecting parameters can also be regarded in a similar way as choosing the number of hidden nodes in the ANNs, which use cross–validation over the training set to find the best. The results indicated that the method could classify complicated patterns and achieved higher recognition rate at reasonably small size of training sample set. This means the proposed method was effective for signals recognition of electronic nose.

Our comment is, if we have more than three gas types how will work the system, and how many SVMs we must use?

Pardo and Sberveglieri (2005) applied the SVM to the classification of ENose data. They analyzed the test error of SVM as a function of (**a**) the number of principal components (on which the data are projected), (**b**) the kernel parameter value, for both the polynomial and the RBF kernel, and (**c**) the regularization parameter. This permitted to explore the insurgence of underfitting and overfitting effects, which are the principal limitations of non–parametric learning techniques. In particular, they found out that the regularization parameter C, often set a priori to $C = 1$, strongly influences SVM performance. SVM were trained on two electronic nose dataset of different hardness, collected with the **Pico** electronic nose which were developed at the Brescia University [41].

They studied two binary classification problems of different hardness with SVM, also investigated the performance of SVM with regard to:

a- The number of principal components, which were given as inputs to the SVM. These range from two to five, five being the number of sensors used in the ENose. By varying the number of PC they wanted to see if also PCs with very small variance ($\sim 1\%$) had an

influence on the classification performance as in the MLP [40].

b- The type of kernel, they tried both polynomial and RBF kernels. For the polynomial, they scanned the polynomial order from 1 (linear machine) to 8, for the RBF they scanned the standard deviation parameter from 0.1 to 10 (at unequal steps, concentrating on small values).

c- The regularization parameter C, which controls the trade off between low training error and large margin [8, 11, 50].

They also showed that, for the easy classification problem, principal components carrying small variance have an impact on SVM performance, just as they observed for multilayer perceptrons in the past [40]. Further, the parameter controlling the trade–off between big margin and low error (in the criterion function) plays a major role in the test error. In their case a value of $C \approx 300$ gave optimal result for the difficult classification problem [41].

Brudzewski *et al.* (2006) used the electronic nose measurement system in cooperation with the Support Vector Machine (SVM) to the classification of the gasoline with the supplement of bio–products, such as ethanol, methyl tertiary butyl ether (MTBE), ethyl tertiary butyl ether (ETBE), tertiary amyl methyl (TAME) and benzene. The array of semiconductor sensors forming the heart of the electronic nose, responds with a signal pattern characteristic for each gasoline blend type. The sensor array used in all numerical experiments of gasoline recognition was composed of seven tin oxide–based gas sensors (TGS815, TGS821, TGS822, TGS825, TGS824, TGS842, TGS822–modificated) from Figaro Engineering Inc., mounted into an optimized test chamber. The SVM network working in the classification mode, processed these signals and associates them with an appropriate class [4].

Two kinds of experiments had been performed. In the first case they have aimed at discovering the kind of supplement without taking into account its concentration. So they have distinguished only three classes (ethanol, MTBE+ETBE and benzene supplements). In the second experiments they have tried to recognize not only the supplement but also

its concentration (12 classes of blends). The problem of 12–class recognition (recognition of the supplement and its concentration) was more difficult. The main task was to find the SVM network structures of the smallest possible number of support vectors at application of different kernel functions in order to achieve 100% accuracy of classification for the testing data. The experiments applying the validation set had been performed to discover the optimal value of the regularization coefficient C and constant γ corresponding to different kernel functions, the linear one, the polynomial and Gaussian. In all cases they have obtained 100% accuracy of the recognition of the samples. Linear kernel SVM networks were sufficient to obtain such good recognition rate.

Chapter 4

The SVM Electronic Nose

We adopted a multi-sensor scheme and useful information is gathered by combining the outputs of the different sensors, because the use of just one sensor does not allow in general to identify the gas. In fact the same sensor output may correspond to different concentrations of many different gases. On the other hand by combining the information coming from several sensors of diverse types we identify the gas and estimate its concentration.

In this chapter we present a detailed description of our system, including block diagram as shown in Figure 4.1, design process, electronic parts, the software used, the training and testing phases. We show also the results of classifying the different types of gases used in our experiments and their concentration estimation.

Our system consists of three major parts, the gas test BOX that contains the array of sensors, the interfacing part which is from the National Instrument company (NI DAQPad–6015), and the personal computer (PC) that saves the data and makes all the training and testing phases including the classification and estimation jobs.

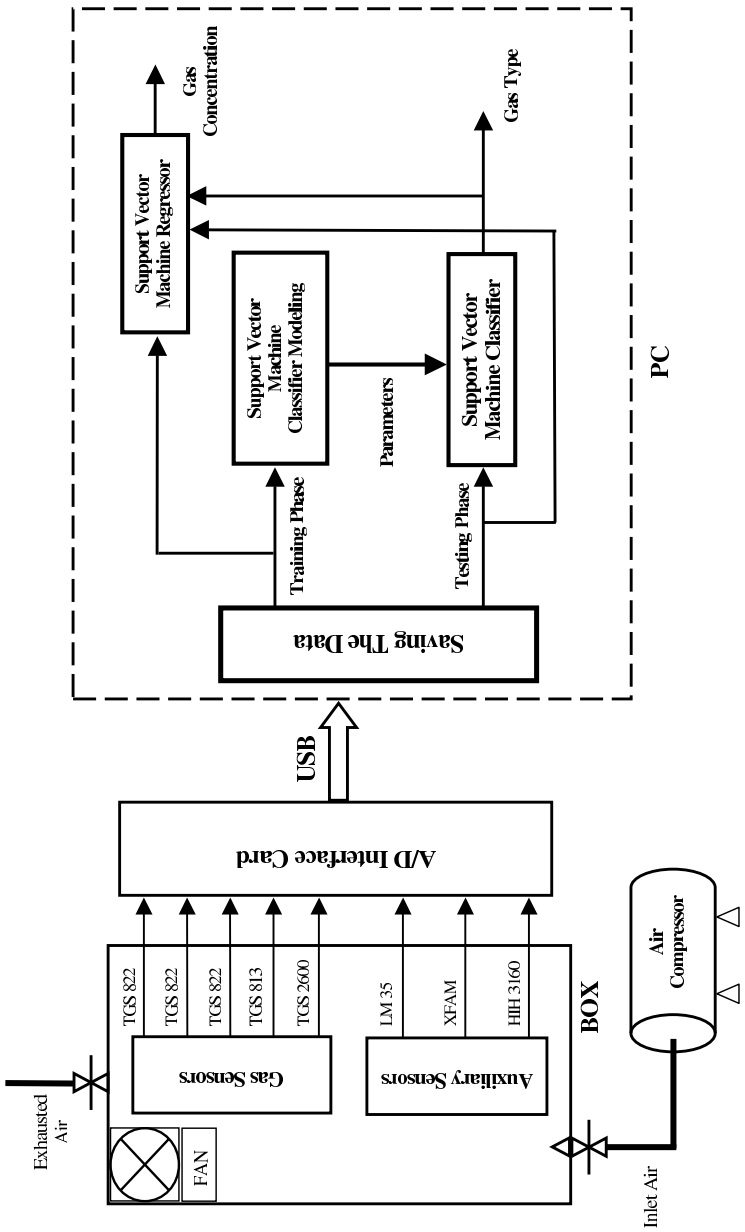

Figure 4.1: Block diagram of the system

4.1 The Gas Test Box

This BOX contains the PCB (Printed Circuit Board) where we fixed two different types of sensors i.e. gas sensors and auxiliary sensors. It also contains a fan for circulating the gas inside during the test. There exists one input for inlet compressed air, coming from an air compressor. It has been used to clean the BOX and the gas sensors after each test. One output is used for the exhaust air. The outer dimensions for the gas test BOX are 22cm length, 14.5cm width, and 10cm height, the effective volume is 3000cc.

The amount of volatile compounds needed to create the desired concentration in the sensor chamber (our BOX) was introduced in the liquid phase using high–precision liquid chromatography syringe. Since temperature, pressure and volume were known, the liquid needed to create the desired concentration of volatile species inside the test chamber (BOX) could be calculated using the ideal gas theory, as we explain below [22]. The gas concentration versus gas volume injected is shown in Table 4.1:

Table 4.1: Gas concentration vs. gas volume

Gas Concentration (ppm)	Volume of Pure Gas (cc)
10	0.03
50	0.15
100	0.3
200	0.6
400	1.2
800	2.4
1000	3.0
2000	6.0

A syringe of $10\mu l$ is used for injecting the test volatile compounds. In the experiments we used four different types of volatile compounds, methanol, ethanol, acetone and benzene. We take methanol as an example for calculating the ppm (parts–per–million) for each compound. Methanol has molecular weight $\mathbf{MW} = 32.04$g/mol and density $\rho =$

0.7918g/cm^3. The volume of the BOX is 3000cm^3; therefore, for example, to get 100ppm inside the BOX, from Table 4.1, we used 0.3cm^3 of methanol, or equivalently $0.3ml$.
The density of methanol is

$$\mathfrak{d} = \frac{\mathbf{P} \times \mathbf{MW}}{R \times T}, \tag{4.1}$$

where:
$\mathfrak{d}$ = the density of the gas of methanol in g/l,
P = The Standard Atmospheric Pressure (in atm) is used as a reference for gas densities and volumes (equal 1 atm),
MW = Molecular Weight in g/mol,
R = universal gas constant in atm/mol.K (equal 0.0821 atm/mol.K),
T = temperature in Kelvin ($T_K = T_C$+273.15).
As a result we get $\mathfrak{d}$ = 1.33 g/l.

Table 4.2: Methanol concentration vs. methanol quantity

Methanol Concentration (ppm)	Methanol quantity (μl)
40	0.2
100	0.5
200	1.0
400	2.0
800	4.0
1000	5.0
1400	7.0
2000	10.0

Methanol has the same mass however if it is gas or liquid:

$$\text{mass} = \text{v}_{gas} \times \mathfrak{d} = \text{v}_{liq} \times \rho. \tag{4.2}$$

where v_{gas} is the volume occupied by the gas of methanol which is equal to $0.3 * 10^{-3} l$, $\mathfrak{d}$ is the density of the gas of methanol as calculated before,

ρ is the constant density of methanol, therefore;

$$\mathrm{v}_{liq} = \frac{\mathrm{v}_{gas} \times \mathfrak{d}}{\rho} \Rightarrow \mathrm{v}_{liq} = \frac{0.3 * 10^{-3} \times 1.33}{0.7918} \tag{4.3}$$

the volume (v_{liq}) is $0.503 * 10^{-6} l$ which provides 100ppm of methanol. This means that if we want to get 100ppm of methanol we must put $0.503 \mu l$ of methanol as liquid in the BOX by using the syringe. Table 4.2 shows different concentrations of methanol (in ppm) versus its quantities (in μl).

Following the same procedures we can calculate the concentration versus quantities, for ethanol, acetone, and benzene. The results are shown in Table 4.3.

Table 4.3: Concentrations vs. Ethanol, Acetone, and Benzene quantities

Concentration (ppm)	Ethanol quantity (μl)	Benzene quantity (μl)	Acetone quantity (μl)
25	0.181	0.278	0.228
100	0.727	1.114	0.915
200	1.454	2.228	1.83
400	2.908	4.456	3.66
800	5.816	8.912	7.32
1000	7.27	11.14	9.15
1400	10.178	15.596	12.81
2000	14.54	22.28	18.3

4.1.1 Gas Sensors

In this section we give the technical information for each type of gas sensors used in our system.

TGS 813

This type of sensors from FIGARO are considered as a general purpose sensor with a good sensitivity to a wide range of combustible gases like

methane, propane, and butane. Other important properties of this type of sensor are long life, low cost and simplicity of the electric circuit. TGS 813 has a wide variety of applications, like, domestic gas leak detectors and alarms, and portable gas detectors.

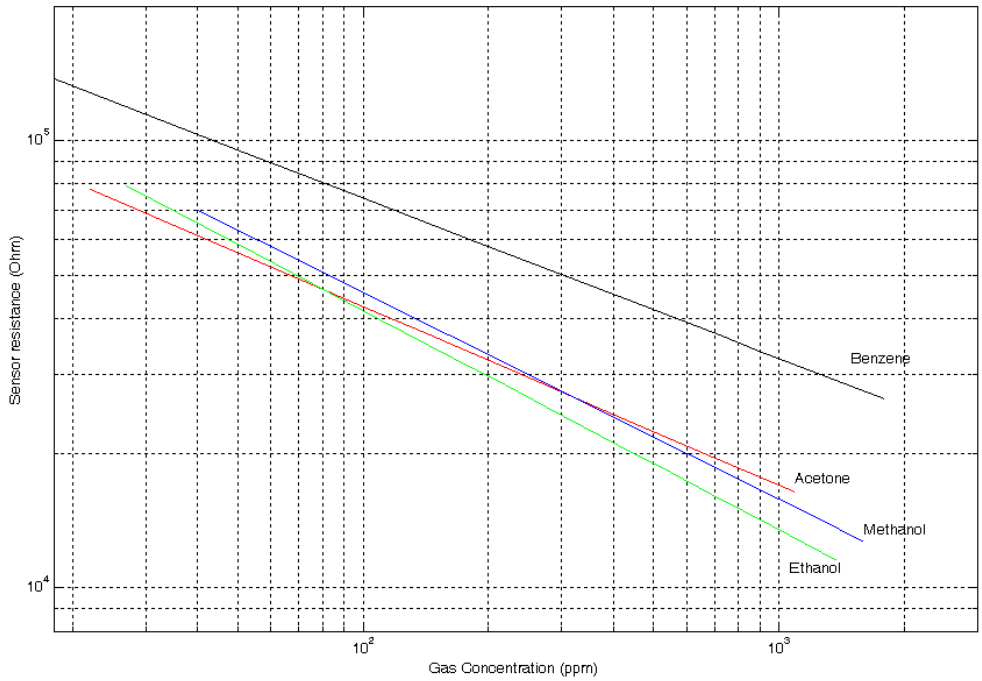

Figure 4.2: Sensitivity characteristic for the sensor type TGS 813

The sensing element of TGS 813 is a tin dioxide (SnO_2) semiconductor which has low conductivity in clean air. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air as shown in Figure 4.2. When a metal oxide crystal (such as SnO_2) is heated at a certain high temperature in air, oxygen is adsorbed on the crystal surface with a negative charge. Then donor electrons in the crystal surface are transferred to the adsorbed oxygen, resulting in leaving positive charges in a space charge layer. Thus, surface potential is formed to serve as a potential barrier against electron

flow.

A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration. The sensor resistance (Rs) is calculated by the following formula:

$$Rs = (\frac{Vc}{V_{RL}} - 1) \times R_L \tag{4.4}$$

where Vc is the source voltage (5 volts), V_{RL} is the output voltage, and R_L is the load resistance (5000 Ohm in our system).

TGS 822

The sensing element of FIGARO gas sensors is also a tin dioxide (SnO_2) semiconductor which has low conductivity in clean air. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air, the sensor concentrations versus resistances variations are shown in Figure 4.3 for the gases used in the tests.

A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration. This type has high sensitivity to the vapors of organic solvents as well as other volatile vapors. It also has sensitivity to a variety of combustible gases such as carbon monoxide, making it a good general purpose sensor. TGS 822 has many good features like, high stability and reliability over a long period, as well as long life and low cost.

This type of sensor has a wide range of applications in breath alcohol detectors, gas leak detectors/alarms, and solvent detectors for factories, dry cleaners, and semiconductor industries. The sensor resistance can be calculated by using equation 4.4

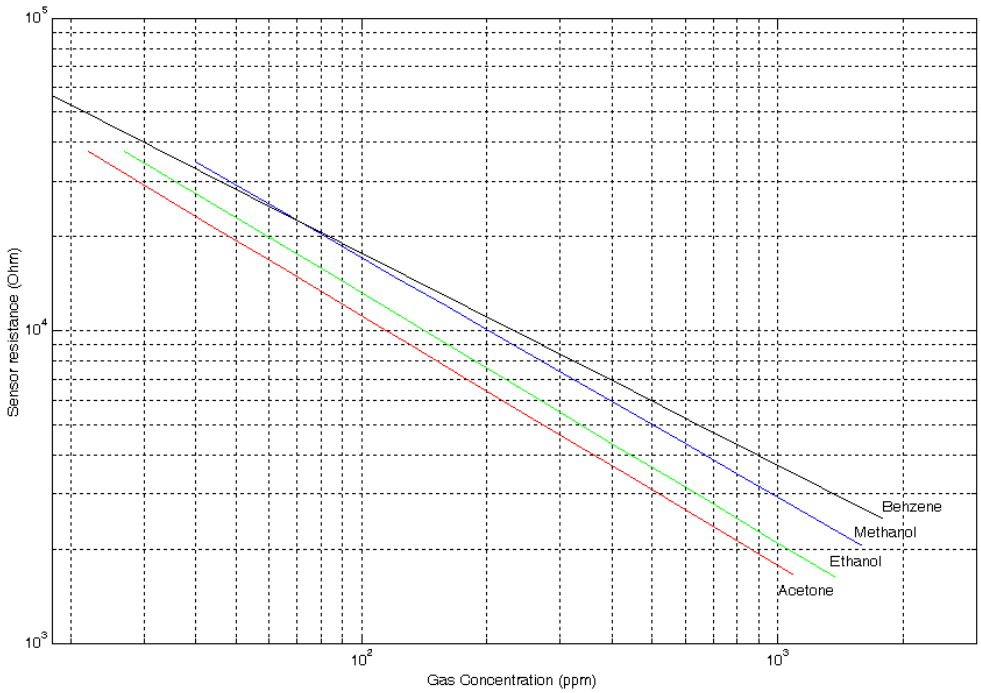

Figure 4.3: Sensitivity characteristic for the sensor type TGS 822

TGS 2600

The sensing element is comprised of a metal oxide semiconductor layer formed on an alumina substrate of a sensing chip together with an integrated heater. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air as shown in Figure 4.4.

A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration. The TGS 2600 has high sensitivity to low concentrations of gaseous air contaminants such as hydrogen and carbon monoxide which exist in cigarette smoke. The sensor can detect hydrogen at a level of several ppm. This type has many fields of application in air cleaners, ventilation control, and air quality monitors.

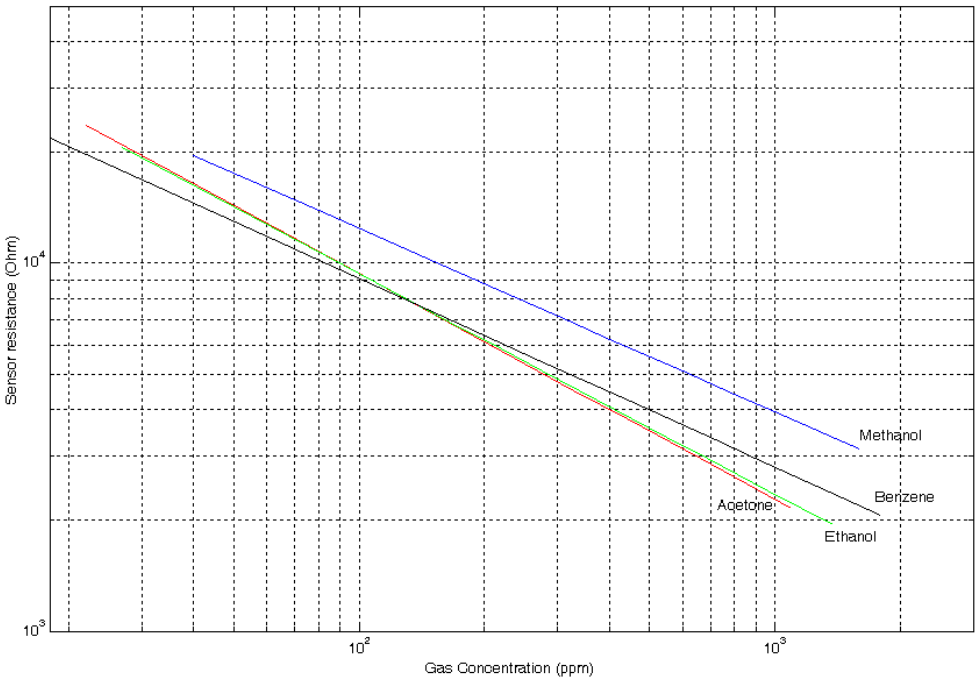

Figure 4.4: Sensitivity characteristic for the sensor type TGS 2600

4.1.2 Auxiliary Sensors

In this section we describe the Auxiliary sensors which are used in our system. They are humidity sensor (HIH–3610 from Honeywell), pressure sensor (XFAM from Fujikura Ltd.), and temperature sensor (LM35 from National Semiconductor Corporation).

Humidity Sensor HIH–3610

The HIH–3610 Series humidity sensor is designed specifically for high volume OEM (Original Equipment Manufacturer) users. Direct input to a controller or other device is made possible by this sensors linear voltage output. With a typical current draw of only 200μA, the HIH–3610 Series is ideally suited for low drain, battery operated systems.

Tight sensor interchangeability reduces or eliminates OEM production calibration costs.

The HIH–3610 Series delivers instrumentation–quality RH (Relative Humidity) sensing performance in a low cost, solderable SIP (Single In–line Package). Available in two lead spacing configurations, the RH sensor is a laser trimmed thermoset polymer capacitive sensing element with on–chip integrated signal conditioning. The sensing element's multilayer construction provides excellent resistance to application hazards such as wetting, dust, dirt, oils, and common environmental chemicals.

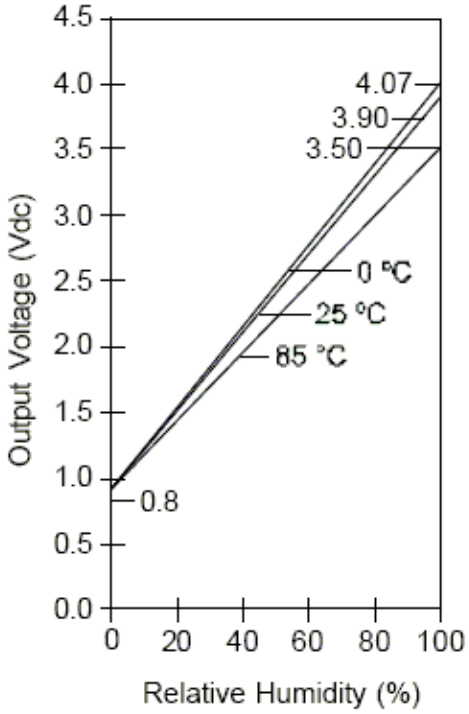

Figure 4.5: Output voltage vs. relative humidity at 0 °C, 25 °C, 85 °C

HIH–3610 sensor has many good features like it's linear voltage output versus relative humidity as shown in Figure 4.5, laser trimmed interchangeability, low power design, high accuracy, fast response time,

stable low drift performance, and chemically resistant. Because of these features it has many typical applications, for example, refrigeration, drying, metrology, battery–powered systems, and OEM assemblies.

Pressure Sensor XFAM

Fujikura silicon piezoresistive pressure sensors are designed and manufactured incorporating the best semiconductor technology available to meet today's demanding requirements of high accuracy, low cost and high reliability. In order to meet the customer's diverse needs, Fujikura Ltd. provides sensors at many performance levels from low cost, unamplified pressure sensors to fully integrated transducers with all circuitry on–chip. Based on the piezoresistance effect, Fujikura's silicon pressure sensors are piezoresistive sensors. A diffusion strain gauge on the surface of a silicon wafer is formed using the same manufacturing technology used in the integrated circuit industry. The silicon itself is then thinned by an etching process and formed into diaphragms which differ in size and thickness depending on the intended pressure range.

This sensor has many applications areas, in industrial instrumentation, medical device, barometer, altimeter, and altitude compensation. The transfer function for XFAM is

$$\begin{aligned} V_{out} = \quad & V_s \times (P \times \mu + \lambda) \pm (\text{Pressure Error} \\ & \times \text{Temperature Error Multiplier} \times \mu \times V_s), \end{aligned} \tag{4.5}$$

where $V_s = 5.0$volts, P the input pressure (KPa), $\mu = 0.009$, $\lambda = -0.095$, Pressure Error=2.5KPa, and the pressure range 15∼115KPa. The Temperature Error Multiplier can be calculated from the curve shown in Figure 4.6.

Temperature Sensor LM35

The LM35 series are precision integrated–circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade

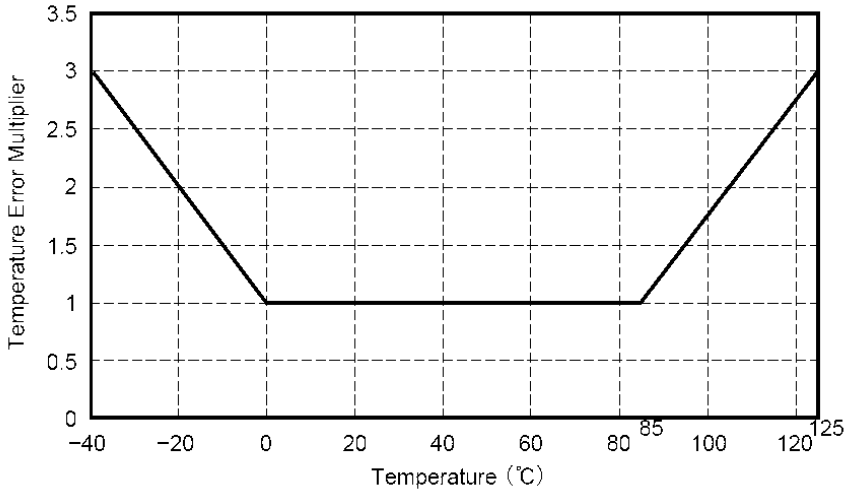

Figure 4.6: Temperature vs. Temperature Error Multiplier

scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1/4°C at room temperature and ±3/4°C over a full -55° to +150°C temperature range.

Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 μA from its supply, it has very low self–heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to +150°C temperature range.

4.2 Interfacing Card

Multifunction NI DAQPad–6015 for USB is used in our system as an interfacing between the BOX and the PC, the National Instruments DAQPad-6015 multifunction data acquisition (DAQ) device provide plug–and–play connectivity via USB for acquiring, generating, and logging data. With DAQPad–6015 mass termination, we can cable to external

accessories and signal conditioning devices such as NI SCC devices. It give 16–bit accuracy at up to 200 kS/s, also has 16 analog inputs, 8 digital I/O, 2 analog outputs, and 2 counter/timers.

NI DAQPad–6015 includes NI–DAQmx measurement services software with which we can quickly configure and begin taking measurements with our DAQ device. NI–DAQmx provides an interface to our LabWindows/CVI that we used inside the PC.

4.3 The Software

We used the LabWindows package as a programming tool running on a PC Pentium 4 type. The integrated LabWindows/CVI environment features code generation tools and prototyping utilities for fast and easy C code development. It offers a unique, interactive ANSI C approach that delivers access to the full power of C with the ease of use of Visual Basic. Because LabWindows/CVI is a programming environment for developing measurement applications, it includes a large set of run–time libraries for instrument control, data acquisition, analysis, and user interface. LabWindows/CVI also contains many features that make developing measurement applications much easier than in traditional C language environments.

For support vector machine(SVM) training and testing in both multi–class classification and regression we used LIBSVM–2.82 package [7]. LIBSVM–2.82 uses the one–against–one approach [29] in which k(k – 1)/2 binary classifiers are constructed and each one trains data from two different classes. The first use of this strategy on SVM was in [17, 30]. In classification it uses a voting strategy: each binary classification is considered to be a voting where votes can be cast for all data points; in the end a point is designated to be in the class with maximum number of votes. In case two classes have identical votes, though it may not be a good strategy, it simply select the one with the smallest index.

LIBSVM provides a parameter selection tool using the RBF kernel: cross validation via parallel grid search. It can be easily modified for other kernels such as *linear* and *polynomial*. For median–sized problems,

cross validation might be the most reliable way for parameter selection. First, the training data is separated to several folds. Sequentially a fold is considered as the validation set and the rest are for training. The average of accuracy on predicting the validation sets is the cross validation accuracy [18].

For measuring the regression quality we used *Correlation Coefficient*, which is a concept from statistics measures of how well trends in the predicted values follow trends in past actual values. It is a measure of how well the predicted values from a forecast model "fit" with the real–life data [42].

The correlation coefficient is a number between 0.0 and 1.0. If there is no relationship between the predicted values and the actual values the correlation coefficient is 0.0 or very low (the predicted values are no better than random numbers). As the strength of the relationship between the predicted values and actual values increases so does the correlation coefficient. A perfect fit gives a coefficient of 1.0. Thus the higher correlation coefficient (near to 1.0) the better is the regressor [10]. Correlation coefficient is calculated as follows:

$$r = \frac{\sum_{i=1}^{N} X_i \hat{X}_i - \frac{\sum_{i=1}^{N} X_i \sum_{i=1}^{N} \hat{X}_i}{N}}{\sqrt{(\sum_{i=1}^{N} X_i^2 - \frac{(\sum_{i=1}^{N} X_i)^2}{N})(\sum_{i=1}^{N} \hat{X}_i{}^2 - \frac{(\sum_{i=1}^{N} \hat{X}_i)^2}{N})}}, \tag{4.6}$$

where r is the correlation coefficient, X are the actual values, $\hat{X}$ are the predicted values, and N is the number of data points.

4.4 Experiments and Results

In our experiments we used four different types of gases, they are methanol, ethanol, acetone, and benzene, with different concentrations range from low concentration (up to 18ppm) to high concentration (up to 2000ppm), We used 26 gas samples for methanol, 26 gas samples for ethanol, 26 gas samples for acetone, and 18 gas samples for benzene. Each experiment was repeated twice with each concentration. The data set for these solvents (gases) is made up of samples in $\mathbb{R}^8$ space where each sample correspond to the outputs of the sensors for a given couple (gas, concentration) as shown in Figure 4.7.

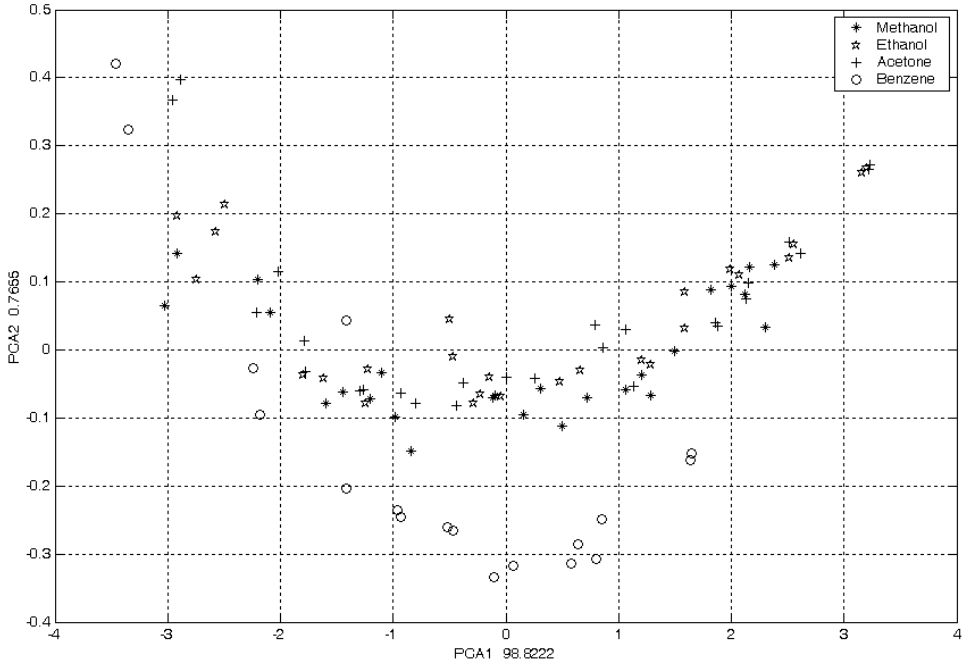

Figure 4.7: First two principal components for the experimental data set

We put the desired quantity of solvent (in ppm) previously calculated inside the BOX, switching ON the fan which is inside the box to let the

solvent drops evaporate easily. The program starts reading the data that are coming from the eight sensors which form our system as shown in Figure 4.8.

After few seconds, when the signals start to be stable, we switch OFF the fan and then we save the data in a file that indicates also the class label of the current sample. After that we must clear the BOX and the sensors by supplying a compressed air coming from an Air Compressor. We repeat twice this procedure for each gas type and for each concentration.

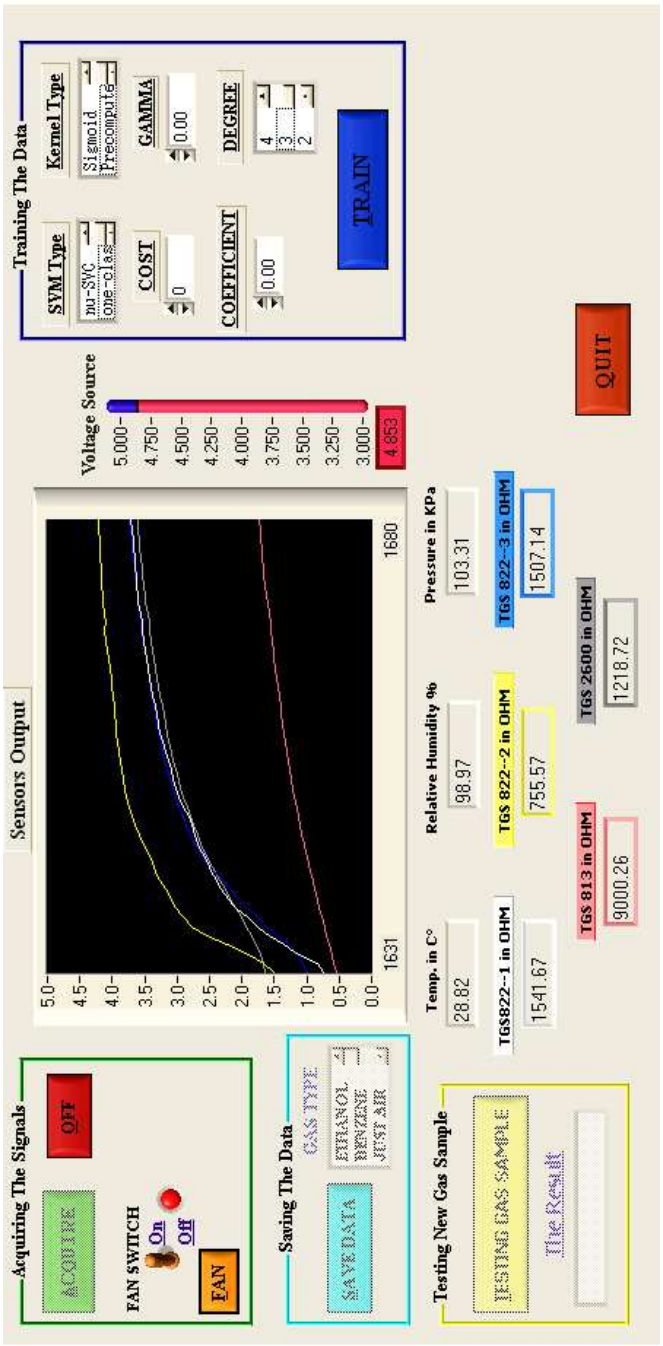

Figure 4.8: Signals are coming from the system

4.4.1 Classification Process

When all the data are saved in the file, which are 96 samples (26 samples for methanol, 26 samples for ethanol, 26 sample for acetone, and 18 samples for benzene), we start the *training phase* by applying the SVM approach to find the best classifier that divides the multiple classes of data with minimum number of misclassified point and maximum margin. This means finding the parameters that give the highest classification rate. In particular for each configuration of the parameters we have run 96 times our SVM program, in a leave–one–out cross–validation fashion obtaining the classification rate which is the ratio between the number of successful classifications of the left out point, and 96, the number of runs. These parameters are kernel type (linear, polynomial, radial basis function (RBF), degree of kernel polynomial function), the value of sigma (σ) in the cases of polynomial, or radial basis function type, the tolerance of termination, and the regularization parameter value (cost C). We used the linear kernel function with many values of C and measured the classification rates as shown in Table 4.4, we got the best classification rate 94.74% when $C = 400$.

Table 4.4: Multiple C values vs. classification rate with linear kernel

C value	Classification rate %
1	56.84
10	80.00
100	90.53
200	91.58
300	92.63
400	**94.74**
500	93.68
700	92.63
900	93.68
1000	93.68

Table 4.5: Multiple C values vs. classification rate for different values of sigma with 3rd degree polynomial kernel

C value	Classification rate%				
	$\sigma = 0.5$	$\sigma = 0.1$	$\sigma = 0.01$	$\sigma = 0.007$	$\sigma = 0.001$
10	95.79	94.74	92.63	90.53	80.00
50	95.79	94.74	94.74	94.74	88.42
80	95.79	94.74	94.74	93.68	90.53
100	95.79	94.74	94.74	93.68	90.53
500	95.79	94.74	94.74	93.68	93.68
1000	95.79	94.74	94.74	93.68	94.74
5000	95.79	94.74	94.74	93.68	93.68
10000	95.79	94.74	94.74	93.68	93.68

We used also the polynomial kernel of 3rd degree with different values of σ and different values of C. The results in Table 4.5 show that the best classification rate was 95.79%, when $\sigma = 0.5$, and C with any value between 10 and 10000.

Also we applied the RBF kernel type with many different σ values and costs (C), as shown in Table 4.6. We got the best classification rate 96.84%, when $\sigma = 0.007$ and $C = 10000$.

Table 4.6: Multiple C values vs. classification rate for different values of σ with RBF kernel

C value	Classification rate%				
	$\sigma = 0.5$	$\sigma = 0.1$	$\sigma = 0.01$	$\sigma = 0.007$	$\sigma = 0.001$
10	70.53	55.79	0.00	0.00	0.00
50	89.47	87.37	42.10	35.79	0.00
80	90.53	88.42	67.37	48.42	0.00
100	91.58	90.53	73.68	62.10	0.00
500	93.68	92.63	89.47	86.31	52.63
1000	93.68	93.68	92.63	90.52	69.47
5000	93.68	94.74	95.79	94.74	85.26
10000	93.68	94.74	95.79	**96.84**	88.42

4.4.2 Concentration Estimation Process

At the end of the training phase (for classification), our system is able to classify new gas samples. Then we started the training phase for regression (concentration estimation). We applied also the linear, polynomial, and radial basis function (RBF) kernels, with many parameter values which are the cost value (regularization parameter C), the degree of the polynomial, σ value, and the tolerance of termination.

Table 4.7: Multiple C values in the case of linear kernel

C value	Mean Squared Error	Squared Correlation Coefficient
10	52529.9	0.755
50	38145.8	0.711
100	32899.2	0.741
500	26397.0	0.813
2000	18855.4	0.870
3000	**9390.5**	**0.929**
5000	9707.5	0.924
10000	17469.1	0.860

As the measure of regression quality we have used the cross–validation *Squared Correlation Coefficient* as a regression quality measurement. Regression is good when the value of cross-validation Squared Correlation Coefficient are nearer to the value 1.0.

We show the results of regression of acetone type. When we applied linear kernel with different values of regularization parameter (C), we got the results shown in Table 4.7. When we used a polynomial kernel of 3rd degree with different values of C and σ, we got the result which are shown in Table 4.8.

Table 4.9 shows the regression results when RBF kernel with different values of C and σ are applied.

At the end of classification and estimation process our system has the ability to recognize type of gas and its concentration. The highest

Table 4.8: Multiple C and σ values with polynomial kernel of 3rd degree

C value	Correlation Coefficient		
	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 0.1$
0.1	0.967	0.955	0.632
0.5	0.982	0.963	0.946
1.0	**0.985**	0.970	0.939
5.0	0.974	0.982	0.967
10.0	0.967	**0.985**	0.955
100.0	0.922	0.963	**0.967**

classification rate was 96.84% when RBF kernel applied with $C = 1000$ and $\sigma = 0.007$ as shown in Table 4.6, the best regression result with respect to highest correlation coefficient (0.985) when we used kernel function of 3rd degree polynomial with $C = 10.0$ and $\sigma = 0.5$ as shown in Table 4.8.

Table 4.9: Multiple C and σ values with RBF kernel

C value	Correlation Coefficient		
	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 0.1$
100	0.736	0.741	0.797
200	0.868	0.842	0.780
500	0.869	0.917	0.874
1000	0.909	0.955	0.909
5000	**0.953**	0.968	0.971
10000	0.953	**0.979**	**0.978**

4.5 Conclusions

We have implemented our electronic nose based on the use of different types of sensors, combined with a recognition and estimation system based on SVM. The results appear encouraging in terms of classification and estimation quality.

Further research would be devoted to test the effectiveness of more complex sensor networks as well as of new classification techniques such as semi–supervised learning (which falls between supervised learning and unsupervised learning) that make use of both labeled and unlabeled data for training a typically small amount of labeled data with a large amount of unlabeled data. The problem of testing the method on other types of gases, as well as that of analyzing mixtures of gases definitely deserve future attention.

Bibliography

[1] S. Abe, *Support Vector Machines for Pattern Classification*. Springer–Verlag, 2005.

[2] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2004.

[3] S. Brahim-Belhouari, A. Bermak, M. Shi, and C. Chan, "Fast and robust gas identification system using an integrated gas sensor technology and gaussian mixture models," *IEEE Sensors Journal*, vol. 5, no. 6, pp. 1433–1444, December 2005.

[4] K. Brudzewski, S. Osowski, T. Markiewicz, and J. Ulaczyk, "Classification of gasoline with supplement of bio–products by meansof an electronic nose and svm neural network," *Sensors and Actuators B: Chemical*, vol. 113, no. 1, pp. 135–141, 17 January 2006.

[5] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[6] L. Carmel, N. Sever, D. Lancet, and D. Harel, "An enose algorithm for identifying chemicals and determining their concentration," *Sensors and Actuators B: Chemical*, vol. 93, no. 1–3, pp. 77–83, 1 August 2003.

[7] C. C. Chang and C. J. Lin, "Libsvm: A library for support vector machines," *[Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2001.

[8] V. Cherkassky and F. Mulier, *Learning From Data, Concepts, Theory, and Methods.* John Wiley and Sons, 1998.

[9] H. Chew, R. Bogner, and C. Lim, "Dual v–support vector machine with error rate and training size biasing," *Proceedings of 2001 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Salt Lake City, USA*, vol. 2, 7–11 May 2001.

[10] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*, 3rd ed. Hillsdale, NJ: Lawrence Erlbaum Associates, 2003.

[11] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel–based learning methods.* Cambridge University Press, 2000.

[12] ——, *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[13] C. Distante, N. Ancona, and P. Siciliano, "Support vector machines for olfactory signals recognition," *Sensors and Actuators B: Chemical*, vol. 88, no. 1, pp. 30–39, 1 January 2003.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. NewYork: Wiley, 2000.

[15] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. John Wiley and Sons Inc., 1987.

[16] J. Fraden, *Handbook of modern sensors: physics, design, and applications.* Springer, 3rd edition 2003.

[17] J. Friedman, "Another approach to polychotomous classification," Department of Statistics, Stanford University, Technical report, 1996.

[18] S. I. Gallant, *Neural Network Learning and Expert Systems.* MIT press, 1993.

[19] J. W. Gardner and P. N. Bartlett, *Electronic Noses: Principles and Applications*. Oxford University Press, 1999.

[20] J. E. Gentle, W. Härdle, and Y. Mori, *Handbook of Computational Statistics:Concepts and Methods*. Springer, 2004.

[21] F. Girosi, "An equivalence between sparse approximation and support vector machines," *Neural Computation*, vol. 10, no. 6, pp. 1455–1480, August 1998.

[22] O. Gualdrón, E. Llobet, J. Brezmes, X. Vilanova, and X. Correig, "Coupling fast variable selection methods to neural network–based classifiers: Application to multisensor system." *Sensors and Actuators B: Chemical*, vol. 114, no. 1, pp. 522–529, 30 March 2006.

[23] L. Guangli and P. Bo, "v–support vector classification with uncertainty based on expert advices," *Proceedings of 2005 IEEE Int. Conf. on Granular Computing*, vol. 2, pp. 451–453, 25–27 July 2005.

[24] R. Gutierrez-Osuna, "Pattern analysis for machine olfaction: A review," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 189–202, June 2002.

[25] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Springer–Verlag New York, 2001.

[26] D. Huang and H. Leung, "Simultaneous classification and concentration estimation for electronic nose," *IEEE Sensors Journal*, vol. 7, no. 5, pp. 825–834, May 2007.

[27] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, January 2000.

[28] T. Joachims, "Making large–scale svm learning practical," in *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. MIT Press, 1999, pp. 169–184.

[29] S. Knerr, L. Personnaz, and G. Dreyfus, "Single–layer learning revisited: a stepwise procedure for building and training a neural network," in *Neurocomputing: Algorithms, Architectures and Applications*, J. Fogelman, Ed. Springer–Verlag, 1990, pp. 169–184.

[30] U. Kreßel, "Making large–scale svm learning practical," in *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. MIT Press, 1999, pp. 255–268.

[31] W. Land, O. Sadik, J. Embrechts, D. Leibensperger, A. Wanekayab, L. Wong, and M. Uematsu, "New results using multi array sensors and support vector machines for the detection and classification of organophosphate nerve agents," *IEEE International Conference on Systems, Man. and Cybernetics*, vol. 3, pp. 2883–2888, 5–8 October 2003.

[32] J. Lozano, J. Santos, M. Aleixandre, I. Sayago, J. Gutirrez, and M. Horrillo, "Identification of typical wine aromas by means of an electronic nose," *IEEE Sensors Journal*, vol. 6, no. 1, pp. 173–178, February 2006.

[33] W. Maass, "On the computational power of winner–take–all," *Neural Computation*, vol. 11, no. 12, pp. 2519–2536, November 2000.

[34] M. Markou and S. Singh, "Novelty detection: A review–part 1: Statistical approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, December 2003.

[35] T. Mitchell, *Machine Learning*. McGraw–Hill, 1997.

[36] K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. on Neural Networks*, vol. 12, no. 2, pp. 181–201, March 2001.

[37] E. Osuna, R. Freund, and F. Girosi, "Support vector machines: Training and applications," Massachusetts Institute of Technology, Cambridge, MA, USA, Technical Report 144, March 1997.

[38] ——, "Training support vector machines: An application to face detection," *Proceedings of 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

[39] L. Palagi and M. Sciandrone, "On the convergence of a modified version of svm^{light} algorithm," *Optimization Methods and Software*, vol. 20, pp. 311–328, 2005.

[40] M. Pardo and G. Sberveglieri, "Coffee analysis with an electronic nose," *IEEE Trans. on Instrumentation and Measurement*, vol. 51, no. 6, pp. 1334–1339, December 2002.

[41] ——, "Classification of electronic nose data with support vector machines," *Sensors and Actuators B: Chemical*, vol. 107, no. 2, pp. 730–737, 29 June 2005.

[42] M. Penza, G. Cassano, and F. Tortorella, "Identification and quantification of individual volatile organic compounds in a binary mixture by saw multisensor array and pattern recognition analysis," *Measurement Science and Technology*, vol. 13, pp. 846–858, 2002.

[43] R. Polikar, R. Shinar, V. Honavar, L. Udpa, and M. Porter, "Detection and identification of odorants using an electronic nose," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3137–3140, 7–11 May 2001.

[44] M. Pontil and A. Verri, "Properties of support vector machines," *Neural Computation*, vol. 10, no. 4, pp. 955–974, May 1998.

[45] B. Schölkopf, A. J. Smola, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[46] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony, "Structural risk minimization over data-dependent hierarchies," *IEEE Trans. on Information Theory*, vol. 44, no. 5, pp. 1926–1940, September 1998.

[47] Y. Tan and J. Wang, "A support vector machine with a hybrid kernel and minimal vapnik–chervonenkis dimension," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 385–395, April 2004.

[48] O. Uluyol, A. Wood, M. Kaiser, and K. Arnold, "Improved e–nose detection using initial reaction smellprint and advanced classifiers," *Sensors, Proceedings of IEEE*, vol. 2, pp. 1214–1218, 22–24 October 2003.

[49] V. Vapnik, *Estimation of Dependences Based on Empirical Data.* Springer–Verlag, 1982.

[50] ——, *The Nature of Statistical Learning Theory.* Springer–Verlag, New York, 1996.

[51] V. Vapnik and O. Chapelle, "Bounds on error expectation for support vector machines," *Neural Computation*, vol. 12, no. 9, pp. 2013–2036, September 2000.

[52] X. Wang, H. Zhang, and C. Zhang, "Signals recognition of electronic nose based on support vector machines," *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou*, pp. 3394–3398, 18–21 August 2005.

[53] H. Zhou, M. L. Homer, A. V. Shevade, and M. A. Ryan, "Nonlinear least–squares based method for identifying and quantifying single and mixed contaminants in air with an electronic nose," *Sensors*, vol. 6, no. 1, pp. 1–18, January 2006.