UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ROBUST OPTIMIZATION IN SUPPORT VECTOR MACHINES AND APPLICATIONS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

In partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

SAMIR A. ALWAZZI

Norman, Oklahoma

2002

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

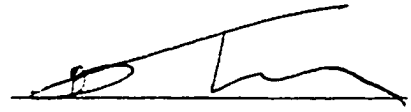
ROBUST OPTIMIZATION IN SUPPORT VECTOR MACHINES AND APPLICATIONS

A Dissertation

APPROVED FOR THE

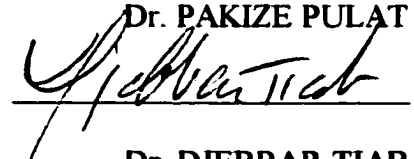
SCHOOL OF INDUSTRIAL ENGINEERING

BY

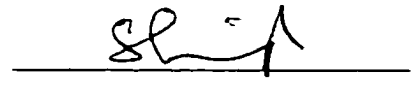


Dr. THEODORE TRAFALIS

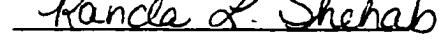
Dr. PAKIZE PULAT



Dr. DJEBBAR TIAB



Dr. SHIVAKUMAR RAMAN



Dr. RANDA SHEHAB

Acknowledgements

In the name of God, Most Gracious, Most Merciful. " Read, in the name of thy Lord cherisher, who created. Created man from a (leech-like) clot. Read, and thy Lord is most bountiful. He who taught (the use of) the pen. Taught man that which we knew not. Nay, but man doth transgress all bounds. To thy Lord is the return (of all)- (Holy Qur'an Surah No.96).

O mankind! Lo! We have created you from a single pair male and female, and have made you nations and tribes that ye may know one another and not to despise each other. Lo! The noblest of you, in the sight of God, is the best in conduct. Lo! God is all Knower, all Aware-(Holy Qur'an Surah No.49).

I would like to express my gratitude to the members of my committee, each and every one of them Dr.P.S. Pulat, Dr.S.Raman, Dr.D.Tiab and Dr.R.Shehab for the time and assistance they gave throughout the development of this work. It is not possible for me to fully express my appreciation and sincere gratitude to my chair advisor, Dr. Theodore Trafalis, for his generous and unlimited assistance, fruitful guidance, understanding and support throughout the entire program. His expertise and guidance were of great importance for this research study. Also, I should not forget Mrs.Trafalis for her encouragement and support.

A word of thanks is also due to all my friends who helped me in one-way or another: Dr.A.Alrefaii, Mr. M. Alramahi, Mr. N. Algiyash, Mr.A.Shaher, Dr. I. Alsaleh, Mr. K. Habib, Mr.S.Hasanjee, Dr. H. Khattab, Dr.M. Hawari, Mr. M.

Alhamly, Dr. M. Issa, Mr. L. Ali, Mr. M. Amgad, Dr.H.Saber, Mr. S. Budi, Mr. I. Hussein, Mr. A. Malyscheff and many others. A special debt of gratitude and affection is extended to my father and my mother for their love, understanding, support and encouragement. Special thanks are also due to my brothers and sister, parents-in-law, and brothers-in-law Mr. N. Alkhaznadar for their great help and Du'ua.

Finally, it is not possible to fully express my thankfulness to my dear wife Nala Alkhaznadar and children (Salsabil, abdullah, Ismaeel, and Issa) for their patience and love during these long and stressful years of this program.

I also acknowledge the support by the National Science Foundation under NSF Grant ECS-9978813.

Contents

List of Tables

List of Figures

Figures for classification problems

Figures for regression problems

Figure 10- Comparison among training and testing points with the original data points for the lynx regression application with uncertainty equals to 0.0001.

Abstract

In the theory of support vector machines learning, it is assumed that the data are precise. However, real world data have uncertainties both in their inputs and outputs. Robust optimization techniques recently have attracted a lot of researchers who are interested in finding solutions to problems dealing with uncertainty, erroneous, or incomplete data. In this dissertation, robust optimization techniques are investigated in the support vector machine approach, with uncertainties in the data in feature space.

The main objective of this work is to investigate the robustness and stability of the behavior of the solutions of the Support Vector Machines model under bounded perturbations of the input data in the feature space. The resulting optimization model is equivalent to a second order cone-programming problem. Specifically those techniques are used both for pattern classification and regression analysis.

Computational results are provided both for synthetic problems such as the AND function and the XOR and real world problems such as the echocardiogram and the breast cancer in the classification case. Also in the regression analysis case, a synthetic problem was created to illustrate the stability of the resulting model. Real world problems (e.g lynx data) were examined. In both classification and regression cases, the obtained results were consistent with the goal of this dissertation.

Chapter 1

INTRODUCTION

1.1 Uncertainty

The decision of managers, engineers and other decision-makers are affected widely by the reality of uncertainty whatever the field you are working in is. Uncertainty in future cash flows makes investment decisions in long term projects difficult, uncertainty in price, labor and other production costs as well as in the availability of needed raw material supplies, complicates the task of the production manager in planning the mix of products to be produced.

Uncertainty is not temporary deviation from well thought out long term plans. It is a basic structural character of the technological and business fields or environment. So how can it be dealt with this fact, which affects the solution of the real-world problems (Kouvelis and Grang 1997).

The researchers have understood this reality, and thought that, the best way to handle the uncertainty, and to be able to make decisions under uncertainty is to accept it, understand it, and then make some good efforts to build a model or structure a system, which can deal easily and properly with this problem, that is the problem of incomplete, noisy and uncertainty data.

This uncertainty in input data sometimes is controlled by some uncontrollable parameters or factors. So in mathematical modeling, the coefficients of the objective

function and the constraints are supplied as input data to the model and hence, the value of the objective function will depend on their values.

In practice, these coefficients are seldom known with absolute certainty; as a consequence, the solution will not be accurate under these circumstances. Each variation in the values of the data may affect the optimum solution found in an earlier stage. So one has to study how it changes with the changes in the input data coefficients (Reklaitis et al 1983). Assume e.g., that linear programming (LP) is a model of technological process in chemical industry. The process is comprised of a number of decomposition-recombination stages. In a meaningful production plan, the balance constraints must be satisfied (the amount of every material to be used at a particular stage can not exceed the amount of the same material yielded by the preceding stages); at the same time, these balance constraints involve coefficients affected by uncertainty of the raw material, parameters of the devices, e.t.c.

Since dealing with the real-world problems is our concern, researchers in recent years have made some attempts to get closer to this concern and find the best available solution for this problem. Thus, their attempts were successful by developing an attractable approach known as the robust approach, which can deal properly with the problem of uncertainty. It is important to shade some light about the lack of knowledge of the data that might be in reality. The crude knowledge of a data, is best explained by giving examples from the real-world:

A) The data element ζ, is unknown at the time the values of the variables x should be determined but will be known in the future. For example in a production

problem, components of the vector x are the production variables and ζ is the future demand.

In construction engineering e.g., building a bridge, the vector x could be the cross-section of bars and the data element ζ represents the future load that the bridge will have to carry.

B) At the time, the vector x is determined, the data element ζ becomes realizable but can not be measured; e.g material properties like the Young's modulus, temperature and pressure (Ben-Tal and Nemirovski 1998).

To understand more about the properties of the uncertainty data, one should locate the area of study where incomplete, noisy or uncertain data can be found. In fact, they can be located in many different areas such as:

1) In social sciences, for example, data are incomplete most of the time like when partially census surveys are carried out, instead of a complete census of the population. Also in economic modeling, problems will arise because of these incomplete data.

2) In business applications/management science, researchers have to deal with noisy data since those data have some probabilistic distribution nature.

3) In engineering and physical sciences, the data are usually subject to some measurement errors, such as in the case of models of image restoration from remote sensing experiments (Mulvey et al.1995).

Some other models are assumed to be deterministic. These can be found in the world of mathematical programming models. These models are usually formulated by

"best guessing" uncertain values, or by solving "worst-case" or mean-value problems, which will lead to solutions either conservative and potentially expensive, or to an inadequate one such as the large bounds arising in an attempt to solve the mean-value problem (Birge, 1982).

Thus, it can be seen that there are some contradictions between the real-world data, and the world of mathematical programming. So to reconcile the relationship of these two worlds, different approaches have been used to discover the influence or the impact of the data uncertainties on the models' recommendations, such as the sensitivity analysis and the stochastic linear programming (Beal, 1955 and Dantzig, 1955).

Even with these two approaches, their solutions were reactive rather than being proactive.

The robust optimization approach leads to solutions that are less sensitive to the model data, and therefore more robust over other methods. Despite the fact that the robust approach is more applicable than others, it has some limitations sometimes. This limitation can be visualized in the diet example, where the solution according to the robust approach is less sensitive to uncertainty than the linear programming one, but with a little higher cost. But even then, the solution will be more stable (Zenios et al.1995).

1.2 Machine Learning

The main idea in machine learning can be described as in the following; assume, an experiment is conducted under specific rules and conditions, which can be represented as a set of parameters. Usually, the experiment is repeated under different conditions, whenever is possible, so that a more reliable result can be observed. These rules are called attributes or set of attributes. If one can present the outcome of these experiments by a mathematical expression, so that in future, predicting and approximating the outcome becomes possible, then a great deal of achievement will one be proud of. Each of these attributes can be represented by a variable known as the input parameter. Let us name the result of each set of the input parameters by the output parameter. Now after n experiments are conducted, a specific rule should be obtained, which will lead to the correct value (with minimum number of errors), of the output parameter or result. This process is known as training, and the set of experiments is known as the training set, which is used to find the mathematical connection or expression between the given information and the result, or between the input and the output. Thus, during training, we learn about the behavior or the conduct of the unknown system, which is described by a set of input data, and the output or the result is actually known. Thus, once training is completed and a mathematical function (known as discriminant function) has been calculated, the testing stage can start, where the result or the output, of course, is unknown. This discriminant function plays a vital role by being a guide for the testing experiments, in which committing as few errors as possible is our goal. From a mathematical point

of view, let x_j represent the input parameters, and y_j correspond to the output parameters. In other words, the training set can then be described by (x_j , y_j), $j = 1,\ldots, n$. The problem is to find a decision function f (x), which predicts accurately y of any example x that may or may not belong to the training set. The discriminant function f(x) may be parameterized with some parameter vector $w = (w_1,w_2,\ldots,w_n)$, that is determined from the training examples by means of a learning algorithm. To be explicit we should write f (x, w).

1.3 Classification and Regression

Support vector machine (SVM) is a tool for machine learning applications. Its name comes from those points in the input space, which support or "touch" the convex hull of the classes of points we want to classify (Vapnik, 1995). The abbreviation SVM is typically used to describe classification with support vector methods, and support vector regression (SVR) is used to describe regression with support vector methods. From now on, the term "SVM" will refer to both classification and regression methods.

In the pattern classification problem, we restrict our consideration to a two-class problem without loss of generality. The output parameter y_j indicates to which class the input parameters x_j belong. So label $y_j = +1$ refers to class A and label $y_j = -1$ refers to class B. So mathematically a label can be expressed as $y_j \in \{+1,-1\}$, and $x_j \in \Re^d$ is an input vector, $j=1,\ldots,n$,where n is the number of experiments.

The main idea in pattern classification is to find a separating hypersurface that separates two or more classes of points. In the case of linear separable data, there are many possible linear classifiers that can separate the data points. But indeed, there is only one, which maximizes the distance (margin) between it and the closest data points of each class. This linear classifier is known as the optimal separating hyperplane. It should be pointed out here, that there are cases where data sets cannot be linearly separable. Optical character recognition (Vapnik, 1995), breast cancer diagnosis (Mangasarian et al., 1995), can be taken as application examples for the pattern classification problem. In the regression problem, the purpose is to estimate a function which approximates the input and output relationship of (x_j,y_j), $j=1,\ldots,n$. Regression analysis and prediction of the value of financial securities is an example for the regression problem. Note that, most applications consider "one-dimensional" input parameters (Mangasarian and Wolberg, 1995).

1.4 The Kernel Method

Many machine learning algorithms cannot handle non-linear cases. The Kernel method allows making the necessary transformation for the problem from the input space to the feature space (higher dimensional) through the kernel mapping. Hence, linear classification methods can be applied in feature space to solve non-linear problems in the input space. Thus, the idea of the Kernel function is to perform the operations in the input space rather than in the potentially high dimensional feature

space. Thus, the inner product does not need to be evaluated in the feature space. An inner product in feature space can be expressed as:

$$K(x_1, x_2) = \varphi(x_1).\ \varphi(x_2),$$

where φ is a map, which maps the input space to the feature space. Finding φ generally is a very difficult problem.

The mapping of these inner products yields a linear classification problem or a linear regression one in the feature space. Figures 16 illustrate the transformation process of the problem data from input space to feature space in classification problems. Similar procedure for transformation can be used for regression problems as can be seen in figure 17.The most commonly employed functions are satisfying Mercer's condition. Among the acceptable mappings are: 1) polynomials such as $\varphi(x_1, x_2) = ((x_1.x_2) + 1)^d$, where d is the degree of the polynomial, 2) Radial basis functions such as gaussian, $\varphi(x_1, x_2) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$, exponential kernels, Fourier series, linear splines, B_n splines and tensor product splines functions (Gunn, 1997).

Chapter 2

Support Vector Machines

2.1 Introduction

Support vector machine (SVM) is a new technique for machine learning applications and it is gaining popularity due to many attractive features and promising empirical performance (Vapnik, 1995). It can be implemented in two kinds of problems, the classification pattern problem and the regression analysis one. In most applications, SVM performance or the error rates on test sets, either matches or dominates other competing methods. Although SVMs have impressive performance in generalization on unseen data, they can be slow in training (Burges 1996, Osuna and Girosi 1998). The basic idea of the SVM is that, for a given learning task, with a given finite amount of training data, the best generalization performance will be achieved if the right balance is struck between the accuracy attained on that particular training set, and the capacity of the machine; in other words, the ability of the machine to learn any training set without error. In the theory of SVM's, it is assumed that the data are precise. Therefore if we combine the idea of robust optimization and SVM technique to solve pattern recognition problems, then a new approach will be added to the present techniques, and it may well work better than many of them with more stable

or even better results, since uncertainty will be taken into consideration during modeling or constructing the problem.

As it is well known, SVM is dealing with precise data. In this dissertation, we consider uncertainty in the data and develop a new SVM training model by using robust optimization methods. The robust optimization approach leads to solutions that are less sensitive to the model data.

2.2 Mathematical Background

The SVM algorithm developed by (Vapnik and Cortes, 1995) is based on statistical learning theory. The initial formulation of the problem of optimal separation of two classes of data points, consists on finding the hyperplane that separates the two sets in such a way that the distance between it (optimal hyperplane) and the nearest point of each of the data set is maximum. This distance is known as the margin, and since there are many possible linear classifiers (hyperplanes) that can separate the data well, then the hyperplane that divides these two classes with maximum distance is called the optimal separating hyperplane. Let us now write down some formulas to express our thoughts mathematically:

Let us have two parallel hyperplanes in $\Re^d$, H_1: $w .x + b_1 = 0$ and H_2: $w .x + b_2 = 0$. The distance between these two hyperplanes with an L_p norm is the following:

$$d_p(H_1, H_2) = \min_{\substack{x \in H_1 \\ y \in H_2}} ||x - y||_p , \qquad (2.1)$$

where

$$|| x ||_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}} .$$

Equation (2.1) can be rewritten as

$$d_p(H_1, H_2) = \min_{x \in H_1} ||x - y||_p \quad . \tag{2.2}$$

Note, that $y \in H_2$ was taken as an arbitrary point. These two hyperplanes can have different equations if they are shifted parallel, so if we let H_2 passing through the origin and the point y is chosen to be that origin, then the equations of these two hyperplanes will be as the following: H_2': $w.x = 0$ and $H_1' = w.x + (b_1 - b_2) = 0$ and the distance between them will be:

$$d_p(H_1', H_2') = \min_{x \in H_1'} || x ||_p \quad . \tag{2.3}$$

To be able to define the distance between these two hyperplanes, the conjugate L_p and L_q norms are useed. *Hölder's* inequality states the following:

$$|x.w| \leq ||x||_p . ||w||_q \text{ , where } \frac{1}{p} + \frac{1}{q} = 1 \tag{2.4}$$

Since for $x \in H_1'$ $|x.w| = |b_1 - b_2|$ we have

$$\min_{x \in H_1'} || x ||_p || w ||_q = | b_1 - b_2 | . \tag{2.5}$$

Thus, using the L_p norm, the distance between our two hyperplanes becomes:

$$d_p(H_1, H_2) = \min_{x \in H_1'} || x ||_p = \frac{|b_1 - b_2|}{\|w\|_q}, \tag{2.6}$$

where q is the conjugate norm of p (Murata and Pedroso, 1999).

It should be pointed out here, that normalization of constraints is necessary to make sure that the values of w and b, are bounded, so it is appropriate to consider a canonical hyperplane (Vapnik 1995), where the parameters w and b are constrained by:

$$\min_{x_i} |x_i . w + b| = 1 . \tag{2.7}$$

A separating hyperplane in canonical form must satisfy the following constraints,

$$y_i [(w.x_i) + b] \geq 1, \quad i = 1,\ldots,n . \tag{2.8}$$

The distance d(w,b;x) of a point x from the hyperplane (w,b) will be:

$$d\,(w,b;x) = \frac{|w.x + b|}{\|w\|} . \tag{2.9}$$

The optimal hyperplane can be obtained by maximizing the margin ρ(w, b), which is the distance between the hyperplane and the nearest data point of each class, and it is also a function of w and b. This margin is maximized subject to the constraints of equation (2.8), and it can be calculated by the following mathematical equalities:

$$\begin{aligned}
\rho(w,b) &= \min_{\{x_i;y_i=1\}} d(w,b;x_i) + \min_{\{x_j;y_j=-1\}} d(w,b;x_j) \\
&= \min_{\{x_i;y_i=1\}} \frac{|w.x_i + b|}{\|w\|} + \min_{\{x_j;y_j=-1\}} \frac{|w.x_j + b|}{\|w\|} \\
&= \frac{1}{\|w\|}\left(\min_{\{x_i;y_i=1\}} |w.x_i + b| + \min_{\{x_j;y_j=-1\}} |w.x_j + b| \right) \\
&= \frac{2}{\|w\|} .
\end{aligned} \tag{2.10}$$

So the hyperplane that minimizes the function

$$\varphi(w) = \|w\| \tag{2.11}$$

will be the optimal hyperplane which separates the data points optimally. SVM formulations based on a general L_p norm can be found in (Mangasarian, 1999).

2.3 Support Vector Machine Formulation Using L_1 Norm

When the L_1 norm is used for determining the distances between the separating hyperplane and the training points, then a linear optimization problem will be formulated. From section two, equation (2.6) and using p =1, equation (2.6) becomes:

$$d_1(H_1, H_2) = \frac{|b_1 - b_2|}{\|w\|_\infty}. \tag{2.12}$$

2.3.1 Linearly Separable Case

Since $w.x_i + b = \pm 1$ for support vectors, then the support hyperplanes corresponding to each class are expressed respectively as:

H^+: $w.x_i + b = +1$, and H^-: $w.x_i + b = -1$ respectively. The distance between the two hyperplanes H^+ and H^- will be:

$$d_1(H^+, H^-) = \frac{|(b+1) - (b-1)|}{\|w\|_\infty} = \frac{2}{\max_j |w_j|}, \; j = 1,\ldots,d. \tag{2.13}$$

The distance $d_1(H^+,H^-)$ needs to be maximized which is equivalent to minimizing the denominator of (2.13).This can be expressed as:

$$\min_{w,b} \max_{j} |w_j| . \qquad (2.14)$$

By setting $\min_{w,b} \max_{j} |w_j|$ equal to an auxiliary variable α, the SVM learning problem is equivalent to the following linear programming problem,

$$\text{Min } \alpha \qquad (2.15)$$

$$\text{S.t} \quad (w.x_i) + b \geq +1, \quad i \in P$$

$$(w.x_i) + b \leq -1, \quad i \in N$$

$$\alpha \geq w_j \qquad \forall j \in \{1,\ldots,d\}$$

$$\alpha \geq -w_j \qquad \forall j \in \{1,\ldots,d\}$$

where $a,b \in \mathbb{R}$, $w \in \mathbb{R}^d$, P = positive examples and N = negative examples.

Note, that the support vectors are the training data points for which $w.x_i + b = \pm 1$, (the points belong to the active or binding constraints). These points can be determined by using an LP software (e.g.Matlab). In section three, an example will be given to illustrate the above analysis by using Matlab.

2.3.2 Linearly Non-Separable Case

If the problem above has no feasible solution, then some constraints such as the soft constraints can be relaxed and a penalty will be added to the objective function. The constraint for the positive examples becomes: $(w.x_i) + b \geq +1 - z_i$, and for the

negative one will be: $(w.x_i) + b \leq -1 + z_i$, where $z_i \geq 0$. The objective function will be as follows:

$$\min_{w,b,z} \left\{ \max_j \left| w_j \right| + C\left(\sum_i z_i \right) \right\}, \tag{2.16}$$

where C is a positive parameter selected by the user. This parameter is important because it can put the classification errors under control in the training operation. The linear programming formulation is as follows:

$$\min \ \alpha + C \sum_{i=1}^{n} z_i \tag{2.17}$$

S.t. $(w.x_i) + b \geq +1 - z_i, \quad i \in P$

$(w.x_i) + b \leq -1 + z_i \quad i \in N$

$\alpha \geq w_j \quad \forall j \in \{1,\ldots,d\}$

$\alpha \geq -w_j \quad \forall j \in \{1,\ldots,d\},$

where $\alpha, b \in \mathbb{R}$, $w \in \mathbb{R}^d$, P = positive examples and N = negative examples.

2.4 Support Vector Machine Formulation Using L_∞ Norm

Again, if one uses the L_∞ norm to measure the distance between the training points and the separating hyperplane, the problem can then be formulated as a linear optimization problem. From previous section recalling equation (2.6), we see that the

distance between the two parallel hyperplanes H_1: w.x + b_1 = 0 and H_2: w.x +b_2 = 0 is:

$$d_\infty (H_1, H_2) = \frac{|b_1 - b_2|}{\|w\|_1}. \qquad (2.18)$$

2.4.1 Linearly Separable Case

In this case, we may also choose w.x_i +b =± 1 for support vectors as in the L_1 norm case. The distance between the two support parallel hyperplane of each class is then:

$$d_\infty (H^+, H^-) = \frac{|(1-b)-(-1-b)|}{\|w\|_1} = \frac{2}{\sum_j |w_j|}, \qquad j = 1,\ldots,d. \qquad (2.19)$$

Now this distance, d_∞, needs to be maximized; minimizing the denominator can do this and thus we need to solve $\min_{w,b} \sum_j |w_j|$ subject to the constraints of separation. This can be transformed to an LP problem by adding an auxiliary variable $\alpha_j = |w_j|$, and the constraints $\alpha_j \geq w_j$, and $\alpha_j \geq -w_j$ for every j. Hence, the problem is to minimize$\sum_j \alpha_j$ subject to all the constraints of separation. The complete formulation for this LP problem is the following:

$$\min \sum_{j=1}^{d} \alpha_j \qquad (2.20)$$

S.t

w.x_i + b ≥ 1 $\quad \forall\, i \in P$

w.x_i + b ≤ -1 $\quad \forall\, i \in N$

$\alpha_j \geq w_j \qquad \forall j \in \{1,\ldots,d\}$

$\alpha_j \geq -w_j \qquad \forall j \in \{1,\ldots,d\},$

where $b \in \Re$, α_j ,$w \in \Re^d$, P is the index set of positive examples , and N is the index set of negative examples.

2.4.2 Linearly Non- Separable Case

In this case, we can still use the soft-margin constraints as in the L_1 case, with the following objective function:

$$\min_{w,b,\xi} \left\{ \sum_j |w_j| + C\left(\sum_i z_i \right) \right\} . \tag{2.21}$$

Then, the following formulation will give the complete LP problem,

$$\min_{w,b,\alpha} \sum_{j=1}^{d} \alpha_j + C \sum_{i=1}^{n} z_i \tag{2.22}$$

S.t. $(w.x_i) + b \geq +1 - z_i, \qquad \forall i \in P$

$(w.x_i) + b \leq -1 + z_i \qquad \forall i \in N$

$\alpha \geq w_j \qquad \forall j \in \{1,\ldots,d\}$

$\alpha \geq -w_j \qquad \forall j \in \{1,\ldots,d\},$

where $\alpha, b \in \mathbb{R}$, $\alpha, w \in \mathbb{R}^d$, $z \geq 0$,C is a user defined positive constant and P and N are the index set of positive and negative examples respectively. In chapter four we will discuss how the robust optimization methods can be used in SVM learning.

2.5 Kernelization

In section 1.4, we have shaded some lights about Kernels in general terms. In this section, we would like to explore the idea of kernelization in a deeper manner. We can observe that the decision function D(x) = w.x + b can be expressed in the form of dot products, $x_i.x_j$. Specifically we can express $w = \sum_{i=1}^{l} y_i \alpha_i x_i$, with $\alpha_i \geq 0$. Then D(x) $= \sum_{i=1}^{l} y_i \alpha_i \langle x, x_i \rangle + b$. Suppose we map the data to some other higher dimensional space H, using a mapping φ. This mapping can be symbolized as $\varphi: \Re^d \rightarrow H$. D (x)= $\sum_{i=1}^{l} y_i \alpha_i \langle \varphi(x), \varphi(x_i) \rangle + b$, and is called feature map. Hence, the training algorithm would only depend on the data through dot products in H, i.e. on functions of the form $\varphi(x_i).\varphi(x_j)$. Note that if there is a kernel function k that equals to this dot product, $k(x_i,x_j) = \varphi(x_i).\varphi(x_j)$, then it would be easy for us and more practical to use k in the training algorithm, and ultimately, we will never need to use or even to know what φ is. Therefore the algorithm will produce a linear support vector machine in the feature space. Indeed, we are still doing a linear separation, but in a different space. Let us use the L_1 norm to maximize the margin. Then this is equivalent to minimize the 1-norm of w (the sum of absolute values of the components of w), while satisfying the separation constraints. Specifically we can consider the following optimization problem:

$$\min_{w,b,z} \|w\|_1 + C\sum_{i=1}^{l} z_i \qquad (2.23)$$

S.t

$$y_i (x_i.w + b) + z_i \geq 1$$

$$z_i \geq 0 \quad i = 1,\ldots,l,$$

where the parameter C is the tradeoff between minimizing the training set error and maximizing the margin. z_i isdefined as the nonnegative slack or error variable, which is added to each constraint for possible violation of the constraint.

The above problem can be transformed into a LP problem solvable by different methods such as the simplex method or by the interior point algorithms. It should be noticed here that, since we are minimizing the 1-norm of w, the optimal would be very sparse. Now the problem can be formulated directly in the kernel or feature space to create nonlinear discriminants. In the original SVM formulation, we found that the final classification was done as follows:

$$f(x) = \text{sign}\left(\sum_i y_i \alpha_i k(x, x_i) + b \right). \qquad (2.24)$$

where $\text{sign}(z) = \begin{cases} +1 & \text{if} \quad z \geq 0 \\ -1 & \text{otherwise} \end{cases}$

Note that $w = \sum_{i=1}^{l} y_i \alpha_i x_i$.

Now let us substitute w as above into the LP (2.23). This yields the following optimization problem:

$$\min_{\alpha,b,z} \|\alpha\|_1 + C\sum_{i=1}^{l} z_i$$

$$S.t$$

$$y_i\left(\sum_{j=1}^{m} y_j \alpha_j k(x_i, x_j) + b\right) + z_i \geq 1$$

$$z_i \geq 0, \alpha_i \geq 0, i = 1,...,l, j = 1,...,m.$$

Now by minimizing $\| \alpha \|_1 = |\alpha_1| + |\alpha_2| + ... + |\alpha_m| = \sum_{i=1}^{m} \alpha_i$ we obtain a solution which is sparse, and not too many data points will be support vectors (Bennett and Campbell 2000). Next we provide the following definitions to ease our presentation in chapters four and five.

2.5.1 Definitions (Features and Feature Space)

Definition 1

A function $\varphi_i : \mathbb{R}^d$ ----> $\mathbb{R}$ that maps each point x in $\mathbb{R}^d$ to a real value φ_i (x) is called a ***feature*** . Combining n features $\varphi_1, \varphi_2, ..., \varphi_n$ results in a feature map φ: $\mathbb{R}^d$ ---->H and the space H is called a feature space.

In order to avoid an unnecessarily complicated notation, we will abbreviate $\varphi(x)$ by **x** for the rest of the dissertation. The vector **x** `in H` is also called the representation of x in $\mathbb{R}^d$.

Definition 2

Suppose we are given a feature map φ: $\mathbb{R}^d$ ---->H . The kernel is the inner product function

K: $\mathbb{R}^d \times \mathbb{R}^d$ ---->: $\mathbb{R}$ in H, i.e for all x_i , x_j , in $\mathbb{R}^d$,

$$K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle = \langle x_i, x_j \rangle.$$

Definition 3 (Kernel Matrix)

Given a kernel K: $\mathbb{R}^d \times \mathbb{R}^d$ ---->: $\mathbb{R}$ and a set of l points x_1 , $x_2, \ldots, x_l$ in $\mathbb{R}^d$, we call the $l \times l$ matrix with ,

$K_{ij} = \langle \varphi(x_i), \varphi(x_j) \rangle = \langle x_i, x_j \rangle$, the kernel matrix of k at x_1 , $x_2, \ldots, x_l$.

Chapter 3

Literature Review and Basic Concepts of Robust optimization

The robust optimization methodology is relatively a new approach to deal with uncertainty in the data. Researchers' work in this field has varied from one aspect to another. Many of them have concentrated their research on building or structuring a modeling methodology, and then developing the analytical and computational optimization tools to obtain solutions for uncertain, noisy, erroneous, or incomplete data, which are common in operation research applications. In this chapter we provide a literature survey and the basics of the robust optiomization approach. Our exposition in sections 3.1 and 3.2 is based on (Ben-Tal and Nemirovski, 1998).

3.1 Robust Optimization

Ben –Tal & Nemirovski (1998), proposed the foundation of robust convex optimization. The main idea of their study about convex optimization problems with uncertainty is that the data are not accurately specified, and the only knowledge about those is that, they belong to a bounded uncertainty set U.

Any value of this data from that set must satisfy all the constraints. They have shown, that when this set U is an ellipsoidal uncertainty set, then the robust convex program corresponding to some of the most important generic convex problems, such as the linear programming, semi-definite programming and others, is either exactly or approximately, a tractable problem which can be solved by an efficient algorithm, such as polynomial time interior point methods. The convex optimization problem with uncertainties can be formulated as follows,

(P) $\quad$ Min $f(x,\zeta)$ $\quad$ $\forall \zeta \in U \subset R^m$ and k is a convex cone

S.t $\quad$ $F(x, ,\zeta) \in K$

Zhou et al (1995) used this notion of feasibility before, in the sense of robust control.

Their definition of the optimal solution to this uncertain optimization problem is that, it must give the best possible value of the original objective under the given restrictions (constraints) $F(x, \zeta) \in K$, $\forall \zeta \in U$.Specifically,

$$\sup_{\zeta \in U} f(x,\zeta).$$

Such a solution should be an optimal one to the uncertain optimization problem. Mathematically this can be expressed as the following certain optimization problem

(P^*) $\quad$ Min $\{\sup_{\zeta \in U} f(x,\zeta), / F(x, ,\zeta) \in K \quad \forall \zeta \in U\}$.

From now on we call feasible solutions of (P^*) robust feasible solutions of (P) and the optimal value of (P^*) is called robust optimal value of (P). Problem (P^*) will be called the robust counterpart of (P).

Special attention was given to situations where, constraints are hard, thus all constraints must be satisfied, and this is very common in reality, especially in engineering applications, such as the process in the chemical industry or in the construction field. In these applications, a small violation of the physical balance constraints can cause a catastrophe such as an explosion in the chemical industry or a crush in the construction field, like the truss topology design (TTD) problem (Ben-Tal & Nemirovisky 1998). A more detailed qanalysis will be given in section 3.2.

3.1.1 Robust Solutions of Uncertain Linear Programs Via Convex Programming

There have been some studies about the above subject conducted by few researchers, trying to deal with the uncertain data of linear programming problems. In some of their studies, they have focused on the uncertainty associated with the hard constraints, which must be satisfied whatever the data's actual realization is within a known uncertainty set.

The robust counterpart (RC) problem has replaced the uncertain linear problem through a certain convex optimization problem. They were aware of the importance of the geometry of the uncertainty set, and thus they have focused on the ellipsoidal set, because they have noticed that computationally was tractable since it leads to a conic quadratic program, which in turn can be solved in polynomial time (Ben-Tal & Nemiroviski, 1996).

The uncertainty set of the LP problem was restricted to the constraints only. Specifically, to the constraints matrix and the right hand side vector (RHS), which belong to a given uncertainty, set U. This vector is feasible for the uncertain LP problem, as long as, it is feasible for every certain realization.

An important notice worth to mention, is that there are some situations in real-world problems, where feasible solution must be attained for all realizations of the data, and not only for part of it, because even a small violation of the constraints can lead to a severely unstable structure, as in the case for problems of designing engineering structures, such as bridges (Ben-Tal & Nemirovski, 1998).
In their study they refer that all constraints are equivalent to a system of linear constraints but indeed, only for column wise uncertainty.

The general case is the one of row-wise uncertainty that belongs to given convex sets. In this case, the robust counterpart is not an LP problem, but rather a conic quadratic one as long as the uncertainty sets for the rows are ellipsoids.

Therefore, they are trying to differentiate between the column-wise and the row-wise uncertainty This is because in the column-wise case, the coefficients of the constraints could be very bad, that is very large which makes the problem very hard to solve. In the row-wise case, the coefficients of the constraints can not simultaneously be as bad as every one of them could (Ben-Tal & Nemirovski, 1996 & 1998).

In the rest of their paper, some definitions were given for the robust counterpart of uncertain LP problems, and then they show the links between this (RC) and the

usual LP, taking into considerations the case of the worst realization of the data from the uncertainty set U.

The semi-infinite approach was investigated, despite the fact that it could lead theoretically to some efficient methods, but in practice and specifically for large-scale applications, it turned out to be poor performer. This is one of the reasons, why the study of uncertainty's geometry was highly considered, and also because it leads to a nice analytical structure, which permits the (RC) problem to be solved with high performance methods such as the interior point methods. Also, the intersection of finitely many ellipsoids sets were considered, and thus a nice structure of the robust counterpart was formed which turned out to be a conic quadratic problem (Ben-Tal & Nemirovski 1996).

Finally a simple portfolio example was given to illustrate their robust solution and then a comparison was made to the solution obtained by the scenario-based approach for the robust optimization of large-scale systems by Mulvey, Vanderbei and Zenios (1995) (MVZ), known as the robust Mathematical programming. That comparison was made between three candidate investment policies:

1-The first is called the nominal, which says to invest all we have in the most promising shares.

2-The second is the robust counterpart, which says to invest equally in all shares.

3-The third is the robust one done by Mulvey et al. (1995).

In the result of that comparison, one can see that the (RC) policy is about 15 times more stable than the nominal one as far as the standard deviation is concerned.

In term of losses, the (RC) never resulted in losses, but instead it yields at least 11% as profit, where the nominal results in 9% losses with probability 0.5. As for the (MVZ) policy, it was more stable than the nominal one, but less stable than the (RC)-policy.

The (MVZ) policy depends heavily on the number of scenarios. With 16 scenarios, the (MVZ) results in losses of about 7% of the number of scenarios and has standard deviation 7.5 times larger than the one for the (RC)-policy. But as the number of scenarios increases, the (MVZ) stability gets closer to those of (RC), although its standard deviation remains 1.8 times worse than the one for (RC) even for up to 256 scenarios. You are referred to (Ben-Tal & Nemirovski, 1998) to see the result.

3.1.2 Robust Counterpart of an Uncertain Linear Programming

Consider the "LP" problem in the following form:

$$(\mathfrak{I}) \qquad \min \{c^T x \ / \ Ax \geq 0,\ f^T x = 1\}, \qquad (3.1)$$

where c, $f \in R^n$ are fixed data, and the uncertainty is associated only with the m×n matrix A. This matrix A, is a member of the uncertainty set U of m×n matrix real matrices; thus matrix A belongs to U. The robust counterpart form ($\mathfrak{I}_u$) of ($\mathfrak{I}$) is the following:

$$(\mathfrak{I}_u) \qquad \min \{c^T x \ / \ Ax \geq 0,\ \forall\ A \in U;\ f^T x = 1\}. \qquad (3.2)$$

Therefore we define the robust feasible solution, as a solution, which satisfies all realizations of the constraints from the uncertainty set U. Also, the robust optimal

solution is defined, as the robust feasible solution for which the best value of the objective function is obtained. Now for the sake of flexibility, let us make some assumptions:

1-Assume that the uncertainty set U is convex and closed, (since replacing U by its closed convex hull will not change ($\Im_u$)).

2-Fix the certain "LP" data U, c, and f. Note that c, f and some $A \in U$ are given.

3-Denote by ($\Im^*$) to the family of instances or realizations of the uncertain "LP" program ($\Im$), then ($\Im^*$) = {($\Im$).

3.1.3 Scope on The Worst LP In ($\Im$) and In ($\Im_u$)

An important question should be asked is the robust counterpart ($\Im_u$) worse than the worst instance from ($\Im$)?

To answer this question, let us assume that c^* is the finite optimal value to the feasible (RC) problem ($\Im_u$), which is by construction greater than or equal to the optimal value $c^*(\Im)$ of every problem instance $(\Im) \in (\Im^*)$. The question now is:

a) Will there be an infeasible instance $(\Im) \in (\Im^*)$ if ($\Im_u$) is infeasible?

b) Will there exist a problem instance $(\Im) \in (\Im^*)$ with $c^*(\Im) = c^*(\Im_u)$?

The following example will show that the gap between the solvability properties of the instances of an uncertain "LP" program and those of the robust one ($\Im_u$) of the problem may exist. Consider this problem:

$$\text{Min } x_1 + x_2$$

$$\text{s.t} \qquad a_{11}x_1 + x_2 \geq 1$$

$$x_1 + a_{22}x_2 \geq 1$$

$$x_1 + x_2 = 1$$

$$x_i \geq 0, \quad i=1,2$$

a_{11} and a_{22} are the elements of uncertainty set $U = \{ a_{11} + a_{22} = 2, \quad 1/2 \leq a_{11} \leq 3/2\}$, with optimal value equals to one, then every problem instance is solvable. So for example, if $a_{11} \geq 1$, then the corresponding optimal solution is (1,0) or (0,1). The robust counterpart for the above example is:

$$\text{Min } x_1 + x_2$$

$$\text{s.t} \qquad 1/2\,x_1 + x_2 \geq 1$$

$$x_1 + 1/2\,x_2 \geq 1$$

$$x_1 + x_2 = 1$$

$$x_i \geq 0, \quad i=1,2$$

This (RC) is infeasible if we use any of the two optimal solutions above, namely (1,0), (0,1). This is because the existence of the gap between the solvability properties of the two problems (the uncertain "LP" and its certain robust). Now under some specific natural assumptions, this gap could disappear and ($\Im_u$) won't be worse than the worst instance from ($\Im^*$) (Ben-Tal and Nemirovski, 1996).

Next we make the following assumptuions.

Assumptions:

1-Constraint –wise uncertainty

2-Boundedness existence

Let us elaborate about the first part of the above assumptions:

1-Constraint –wise uncertainty

Let U_i be the set of all possible realizations of ith row in the constraint matrix, that is the projection of $U \subset R^{m\times n} = R^r \times ... \times R^n$ onto i-th direct factor of the (RHS). Hence x is robust feasible if and only if:

$$a^T x \geq 0 \quad \forall\, a \in U_i \quad \forall\, i \;;\; f^T x = 1. \qquad (3.3)$$

In a more simpler word, if the initial uncertainty set U is extended to the direct product $\hat{U} = U_1 \times ... \times U_m$, then $(\Im_u)$ "feels" only the possible realizations of the i-th constraint, i=1,...,m and does not "feel" the dependencies (if any) between these constraints in the instances. Next we provide the following definition:

Definition (Ben-Tal and Nemirovski, 1998)

The uncertainty U is called is called constraint-wise, if it coincides with $\hat{U}$ rather than being a proper subset of the later. Thus, any given uncertainty set U can always be extended to a constraint –wise uncertainty set and still obtaining the same RC .

Now let us elaborate about the second assumption:

2-Boundedness Existence

There is a convex compact set $Q \subset R^n$, which for sure contains feasible sets of all problem instances $(\Im) \in (\Im^*)$. This assumption can be ensured; if the constraints of all instances have a common "certain" part, which defines a bounded set in R^n. As a result of the above two assumptions, a proposition can be generated:

If assumption one and two hold, then:

1-($\Im_u$) is infeasible if and only if there exists an infeasible instance ($\Im$) $\in$($\Im^*$).

2-If ($\Im_u$) is feasible and c^* is the optimal value of the problem, then c^* equals to the supremum of the c^* ($\Im$). Therefore

$$c^* = \sup\{ c^* (\Im) : (\Im) \in (\Im^*)\}.$$

For a proof, see Ben-Tal and Nemirovki (1996).

3.2 Robust Counterpart of Uncertain Convex programming

The convex optimization problem with uncertainties can be formulated as follows,

$$\text{Min } f(x,\zeta)$$

$$\text{S.t} \quad F(x, ,\zeta) \in K \subset R^m \qquad (p_\zeta)$$

$$\forall j \in U,$$

where:

ζ is the data element of the problem, which belongs to R^M .

x is the decision vector, which belongs to R^n.

K is a convex cone describing the constraints of the problem.

All of the above symbols along with dimension n, m, M and the mapping f(. , .) , F (. , .) , describe the structural elements of the problem.

It should be noticed here, as part of defining the problem is that, the data element or data vector satisfies the following properties:

It belongs to a given uncertainty set $U \subset R^M$.

It is bounded.

Therefore for every $\zeta \in U$, the constraints $F(x, \zeta) \in K$ must be satisfied no matter what the realization of ζ is.

Thus, for a vector x to be a feasible solution to the uncertainty optimization problem (p_ζ), must satisfy all possible realization of the constraints, specifically,

$$F(x,\zeta) \in K \ , \forall \zeta \in U. \qquad (3.4)$$

Now, a solution to the uncertain optimization problem (P) is called optimal solution, if it gives the best value of the original problem, which is: sup $f(x, ,\zeta) \in K$ $,\forall \zeta \in U$.

Hence we can replace the original uncertain problem (P) with its robust counterpart (P^*):

$$(P^*) \qquad \text{Min } \{\sup f(x, \zeta)\}$$

$$\text{S.t} \qquad F(x, ,\zeta) \in K \ ,\forall \zeta \in U.$$

Thus, the best guaranteed value of the original objective under constraint (3.4) is the optimal solution, which should also be the optimal solution to the robust counterpart (P^*).

So any feasible/optimal solution to the certain problem (P^*) will be also feasible/optimal solution to the uncertain problem (P). It should be pointed out here, that in reality, there are some situations where the constraints can be violated. If these violations are not that crucial, then the robust counterpart method may be conservative meaning that the solution we get may not be desirable or may be less desirable than the one we get by the nominal approach. Many of the arising problems

in some of Ben-Tal's (1995) papers are highly degenerate. In this case, they use the nominal approach that ignores the uncertainty factor. The solution they normally get is a point on the boundary from a massive set of nearly optimal solutions. Now this point (solution) may be very bad solution to the perturbed data problem.

What we are trying to say is that, even under the above situation, it is worth to try the (RC) approach, because it tries to choose from the most inner solution from the massive optimal set, which normally will be more stable with respect to data perturbations than the boundary one. The uncertain convex program (P) is nothing but a set or a family of instances of type (p), where (p) is

$$\text{Min } c^T x$$

$$(p) \qquad \text{S.t} \qquad F(x, ,\zeta) \in K \ , \ x \in X \qquad (3.5)$$

$$(P) = \{(p): \{\min c^T x \ / \ F(x, ,\zeta) \in K, x \in X\}\}_{\zeta \in U}, \qquad (3.6)$$

where:

c and x belong to R^n and $\zeta \in U \subset A$, they are the design certain cost and data vector respectively. We need some assumptions so that we ensure that (p) always be a convex program whenever $\zeta \in A$. Those are as follows:

1) $K \subset R^N$ is a closed convex cone with a non-empty interior.

2) X is a closed convex subset in R^n with a non-empty interior.

3) A is a closed convex subset in R^M with a non-empty interior.

4) F (x, ζ) is a continuous differentiable, mapping on the Cartesian product X x A and K-concave in x, that is: $F(\lambda x'+(1-\lambda)x'',\zeta) \geq_k \lambda F(x',\zeta) +(1-\lambda)F(x'',\zeta)$. Note that $b \geq_k a$ stands for $(b-a)\in K$ (Ben-Tal & Nemirovski, 1998).

The (RC) of the above uncertain program (P) is the following certain optimization problem:

$$(P^*) \qquad \{\min c^T x: F(x, ,\zeta) \in K,\ x \in X\ ,\ \forall \zeta \in U\} \qquad (3.7)$$

Note that in (3.6) the notion $\zeta \in U$ refers to a specific problem, while in (3.7) refers to every problem observing all the above assumptions.

In the case where the uncertain optimization problem (P) has concave uncertainty then, the infimum of the objective value at robust feasible solutions to (P) will be the robust optimal value of (P).

Note that the problem (P) will have concave uncertainty if the underlying mapping

F (x, ζ) is K-concave in the data vector ζ if

$$\forall (x \in X, \zeta', \zeta'' \in A) \text{ and } \forall (\lambda, \zeta' \in [0,1]) \text{ we have}$$

$$F(x, \lambda\zeta' +(1-\lambda)\zeta'' \geq \lambda F(x,\zeta') + (1-\lambda) F(x,\zeta'').$$

So in this case (concave uncertainty), we can assume without loss of generality that, the uncertainty set U is a closed convex set (Ben-Tal and Nemirovski 1998).

3.2.1 Definition And Propositions

Many of the generic convex programs (linear, quadratic...) are affine in the data, so we again define the uncertainty optimization problem (P) as having affine uncertainty if the underlying mapping $F(x, \zeta)$ is affine in ζ, $\forall x \in X$.

Proposition 1 - *Let (P') be the uncertain convex program obtained from (P) by replacing the concave uncertainty set U by its closed convex hull, then the robust counterpart of (P') and (P) are identical to each other. Specifically,*

$$0 \leq_K \Sigma \lambda_i F(x, \zeta) \leq_K F(x, \zeta).$$

For further information about its proof, you are referred to (Ben-Tal & Nemirovski, 1998).

Proposition 2 - *Let the feasible set of the (RC) formulation is a closed convex set. If the (RC) is feasible, then all instances of the uncertain program are feasible, and the robust optimal value of the program is greater or equal to the optimal values of all instances.*

It is very important to notice that, generally there exists a substantial gap between solvability properties of the instances of an uncertain program and those of the RC ones. Ting gap is zero in the case of constraint-wise affine uncertainty, which is a particular case of the uncertain program (P) associated with $K = R^m$.

The constraint –wise uncertainty is exactly the ith component of f depending on x and the ith portion ζ_i. The uncertainty set U is a direct product of a closed convex uncertainty set U_i in the space of ζ_i 's i=1...m. Now assume that the uncertain problem (P) has constraint –wise affine uncertainty and that the set X is compact, then the

(RC) (P^*) of the problem is feasible if and only if all instances are feasible, and the robust optimal value will be the supremum of those instances then:

$$\Sigma^m_{i=1} \lambda_i f_i (x,\zeta_i) < 0, \forall \ x \in X . \qquad (3.8)$$

According to proposition (1), it is only enough to show that all instances of (P) are feasible, then:

i) The RC (P^*) is feasible and ii) The optimal value of (P^*), c^* is the supremum of the optimal values of all instances. i) Can be proven by contradiction technique, and ii) can be shown by adding certain constraint $c^T x \leq c^*$ to all instances of the original problem (P), taking into account that X is bounded. When ii) is applied to the resulting problem, it can be concluded that RC is feasible, hence the optimal value in (P^*) is at most c^* which is finite (Ben-Tal & Nemirovski, 1995).

3.2.2 Sensitivity Analysis For Uncertain Programs

One should analyze the similarity and differences between the optimal value of the robust RC (P^*) and that of its nominal one. In another word, what can be said about the proximity of the optimal value of the nominal instance to that of the RC?

Let the nominal instance of an uncertain convex program (P) be the following:

$$(P^0): \quad \min \{ c^T x : x \in X, F(x,\zeta^{0'}) \in K\}, \qquad (3.9)$$

with uncertainty set: $U = \zeta^0 + V$, where V is the perturbation set. Then:

$$(P) \quad \{\min \{c^T x : x \in X, \ F(x, ,\zeta) \in K \}\}_{\zeta \in \zeta^0 + V.} \qquad (3.10)$$

The (RC) (P^*) to (P) is as follows:

$$(P^*) \quad \{\min \{c^T x : x \in X, \ F(x, ,\zeta) \in K, \zeta \in \zeta^0 + V \} \qquad (3.11)$$

So to answer the question of proximity of the optimal values, let us assume that: (P^0) is solvable, G^0 be the feasible set, x^*_0 be the optimal solution and c^*_0 be the optimal value of the instance. Now let us introduce the quantity λ_V (x) of x with respect to V and x$\in$ G^0, known as the feasibility margin of the nominal instance, which will be our standard to measure the closeness of the nominal and the robust optimal values, where λ_V (x) equals to:

$$\lambda_V (x)=\sup \{\lambda \geq 0: \forall (\delta \in V):[\zeta^0 \pm \lambda\delta \in A] \text{ and } [F(x, \zeta^0 \pm \lambda\delta \in K]\}.$$

Lemma (Ben-Tal and Nemirovski, 1990)

Let $\lambda > 1, \rho \in (0,1)$, $x\perp \in G^0$ and $(1-\rho)\, x\perp + \rho G^0$ satisfies the relation $\lambda_V (x) \geq \lambda$. Let also the mapping be affine in x. The optimal value $c^*_u (P^*)$ satisfies the inequality:

$$c^*_u \leq c^*_0 + 2(1-\rho)/(\rho\lambda - \rho + 2)[c^T x\perp - c^*_0] \qquad (3.12)$$

The Lemma allows us to introduce definitions and propositions, which will help in explaining the idea of feasibility and its effect on the (RC), especially in the case where the nominal instance (P^0) has a bounded strictly feasible set with respect to the perturbation set U. So what will happen if the nominal is not strictly feasible?

To answer this question, let us introduce the following definition:

A pair (λ,ρ) with $\lambda > 1$, $\rho > 0$ is admissible, if there exists x$\perp$ for these λ and ρ, that satisfy the logic or the premise of the lemma. The feasibility margin denoted as K_V (P^0) of the nominal instance (P^0) with respect to the perturbation set V, is nothing but the upper bound of the quantities $\rho\lambda /(1-\rho)$ taken over all admissible pairs. With the

help of the definition in the previous section, a proposition can be implied from the above lemma , which states the following:

Proposition 3

Let (P^0) the nominal instance has a positive feasibility margin with respect to the perturbation set V.

Let the function F (x, ζ) be affine in X.

Let the feasible set G^0 of (P^0) be bounded

Then, the robust counterpart (P^*) of (P) is solvable and the optimal value c^*_v satisfies the inequality:

$$c^*_v \leq c^*_0 + 2/1 + K_v(P^0)\, var_{G0}(c), \qquad (3.13)$$

where $Var_G(c) = max_{x \in G}\, c^T x - min_{x \in G}\, c^T x$.

With the help of the proposition in the previous section, the result can be better explained by the following interpretation to some situations, where the actual perturbations in the data are at most of level π, and then the associated perturbation set is:

$$V_\pi = \pi\, W$$

Where W is a symmetric set with respect to the origin of unit perturbation of the data. So what will happen to the (RC) as $\pi \rightarrow +0$? Now proposition 3, gives the result of one case where the nominal instance (P^0) has bounded feasible set, and strictly feasible with respect to the perturbation set W. Then there exist ($\pi^*$and ρ^*) >0 such that the following set:

$$\{(x / [\zeta^0 + \pi^* \delta \in A] \text{ and } F(x, \zeta^0 + \pi^* \delta) \in K]\ \forall \delta \in W\}$$

contains a ρ^* _ dilatation of the nominal feasible set G^0. Then, the (RC)_ (P*π) of the uncertain version of (P^0) associated with perturbations of the level $\pi < \pi^*$ with the corresponding set $U = \zeta^0 + \pi W$, is solvable and the corresponding robust optimal value c^*_π differs from the nominal one c^*_0 by maximum of $o(\pi)$:

$$c^*_\pi \leq c^*_0 + [2\pi (1-\rho) / \pi(1-\rho^*) + \pi^* \rho^*] \, var_{G0}(c) \qquad (3.14)$$

So if the problem is strictly feasible (i.e. $F(x\perp, \zeta^0) \in \text{int } K$ for some $x\perp \in X$) and the unit perturbation's set W and G^0 are bounded, then for sure the desired pair ($\pi^*$and ρ^*) exists.

In the case, where the nominal program is not strictly feasible, then c^*_π will not be a desirable function because it is a bad function of π. Let us illustrate with a small one-dimensional example, where the nominal problem is:

$$\min x \qquad (*)$$

$$\text{S.t.} \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \geq 0$$

As it can be seen, the optimal value is –1.Now if W is the central symmetric convex set in R^2 with nonempty interior, and if the coefficients of x in (*) are only affected by the unit perturbation, then the (RC) (P^*_π) has the singleton feasible set {0} for every positive π, and thus $c^*_\pi = 0$, (Ben-Tal & Nemirovski, 1998).

3.3 Robust Linear Programming Discrimination of Two Linearly Inseparable Sets

The idea of this approach consists of finding a linear surface (plane), as a separator of two linearly inseparable sets. This linear surface is an optimal separator when we minimize the weighted average sum of the violations of the points lying on the wrong side of the separator. These points can be described as the misclassified points belonging to two disjoint points in n-dimensional real space. When the convex hulls of the two sets are also disjoint, the plane completely separates the two sets. Hence, they have been able to overcome the problem of the null solution, which has plagued previous linear programming approaches by using an appropriate weighted sum. According to them, their linear program might give an edge over other linear programming approaches and thus, it becomes a suitable linear program for a linearly inseparable case (Mangasarian and Bennett, 1992).

3.4 Linear Programming With Interval Coefficient "LPIC"

To solve a linear programming problem, it is assumed that the model coefficients must be fixed at specific values. This assumption leads us to believe, that these coefficients are totally accurate; but in real world problems, this is not true, because they are only estimates; in other words, these coefficients are uncertain and thus, using sensitivity testing or analysis of the model becomes necessary to see the changes in their values. Therefore in order to deal with this type of uncertainty, some

or all of the coefficients of the "LP" problem are labeled as intervals. Our goal becomes to find the best and the worst optimum, along with finding the point settings of the interval coefficients that yields these two extremes. This approach will describe the range of the optimized objective function; also the coefficient settings give some insight into the likelihood of these extremes.

As an example of the above discussion, the profit per unit of production of an item is likely estimated from knowledge of average component costs, labor costs, and wholesale prices over previous or preceding months. The question here is how a manager is going to treat the results of an "LP" solution in the case where coefficients are known to be uncertain. In practice, a manager would like to know the range of optimum solutions that could be returned by the "LP" model with various settings of the uncertain coefficients. So any unknown coefficient can be expressed as an interval, that is, a range of real numbers (lower and upper bounds). The best optimum represents, the highest maximum or the lowest minimum and the worst optimum represents, the lowest maximum or the highest minimum respectively. The solution of this "LPIC" problem provides the manager with the necessary information to make his/her decision. He/she will have the taste of risk involved by knowing the best and the worst optimum solutions.

Similarly, the specific values of the uncertain coefficients can be chosen to reflect the risk-taking strategy (Chinneck and Ramadan, 2000). An example will be furnished applying the LPIC approach to taste the flavor of solving problems with

interval coefficients. Algorithm 3 of Chinneck and Ramadan, (2000) article is the basis for solving the following problem with interval coefficients:

$$\text{Min } z = x_1 + x_2$$

S.t

$$C_1: -x_1 + x_2 \geq [-2, -1]$$

$$C_2: [2,3]\, x_1 + x_2 = [3,4]$$

$$C_3: \quad x_2 \leq 3$$

$$x_1, x_2 \geq 0$$

Step 1.1: Convert the equality constraint C_2 into two inequality constraints:

C_{2a}: $3x_1 + x_2 \geq 3$

C_{2b}: $2x_1 + x_2 \leq 4$

Step 1.2: Solve the best optimum "LP":

$$\text{Min } z = x_1 + x_2$$

S.t

$$C_{11}: -x_1 + x_2 \geq -2$$

$$C_{2a}: 3x_1 + x_2 \geq 3$$

$$C_{2b}: 2x_1 + x_2 \leq 4$$

$$C_3: \quad x_2 \leq 3$$

$$x_1, x_2 \geq 0$$

Step 1.3.1: Matlab software was used to find the best optimum: $z = 1$ at $x = (1,0)$.

Step 1.3.2: The best optimum set up for C_2, is as given in step 1.2

Step 1.3.3: Find the best optimum set up for C_2 by solving the following "LP":

$$\text{Max } a + b$$

S.t

$$1.a + 0 = b$$

$$2 \leq a \leq 3$$

$$3 \leq b \leq 4$$

The solution will be: a = 3 and b = 3. Hence, the specific version of C_2 for the best optimum is: $3x_1 + x_2 = 3$.

Step 2.1: Again, according to algorithm 3, there are two models to enumerate and solve:

Model A: $\min z = x_1 + x_2$

S.t

C_{111}: $-x_1 + x_2 \geq -1$

C_{2a} : $3x_1 + x_2 = 3$

C_3: $x_2 \leq 3$

$x_1, x_2 \geq 0.$

This results in: z =1 at x = (1,0)

Model B: $\min z = x_1 + x_2$

S.t

C_{111}: $-x_1 + x_2 \geq -1$

C_{2a} : $2x_1 + x_2 = 4$

C_3: $x_2 \leq 3$

$x_1, x_2 \geq 0.$

This results in: z = 2.333 at x = (1.667, 0.667).

Step 2.2.1: Model B provides the worst z.

Step 2.2.2: The specific version of the interval constraints that provides the worst optimum is given by model B (Chinneck and Ramadan, 2000).

It is noticed that, there are no equality constraints having interval coefficients after being solved. Hence, the best and the worst optima and the point settings of the interval coefficients that yield these optima's can be obtained by solving only two LPs as it has been done above. Unfortunately, an enumerative approach is required when finding the worst optimum for type 1 and 2 "LPIC" having interval equality constraints. For more details, you are referred to the article written by Chinneck and Ramadan (2000).

3.5 Robust Linear and Support Vector Regression

- A new methodology for solving the Huber M-estimator problem is the main purpose of this method. It reduces the problem to a simple quadratic program. This formulation performs well when compared to other algorithms through some computational results for this new quadratic programming for both linear and nonlinear support vector problems. Then, a modification of this quadratic program is introduced to allow kernel functions to be used. Those kernel functions are used to find a non-linear regression surface, which can yield a better testing performance than a purely linear separating surface.

Their experiments have been conducted on a relatively large set of data points; specifically about 20,000 data points (Mangasarian and Musicant, 2000).

- Another methodology for measuring the goodness of fit has been used. This approach did not use the usual quadratic loss function (the mean square error), but rather a different loss function called the ϵ-Insensitive Loss Function, which has some similarities to other loss functions used in the area of robust statistics. Also it is known that in the quadratic loss function which is commonly used in the classical regularization theory, some assumptions to the noise which affect the data points have been made to justify its use, such as the assumption of Gaussian, additive noise, while in the case of support vector machines regression, it is not clear what noise model underlies the choice of insensitive loss function. So it was necessarily to understand the nature of this noise for two reasons:

  1) It makes it easy for us to decide under which conditions it is appropriate to use the ILF in the SVMR rather than the square error loss used in classical regularization theory;

2) It may give us a better understanding of the role of the parameter ϵ, which can be found in the definition of the ILF. One should also notice that, even in the SVMR approach, the same assumptions were used to justify the use of the ILF, and that is that the noise affecting the data is additive and Gaussian, while the mean does not necessarily be zero, and that its variance and mean are random variables with given probability distributions. It should be mentioned also, the use

of the same Bayesian frame work to derive the result in both SVMR and regularization theory approach. Overall, it can be said that in this study, a comparison was made between SVMR and the classical regularization theory (Pontil et al. 1999).

3.6 Stochastic Linear Programming (Slpm)

In many methods, the stabilization of the solution over a period of times has been given a special attention, such as in the case in Ben-Tal & Nemirovski's (1998) work. The (SLP) method does not employ any penalty factors for violation of the constraints. In general, the (SLP) minimizes the cost or maximizes the profit. The method has a probabilistic nature, with a specific statistical distribution for the data of the problem. There have been some developments to the stochastic programming along many branches, namely:

1-Chance-constrained programming, (Mishina, 1996).

2-Active and passive approaches to stochastic programming,(Mishina, 1996).

3-Stochastic programming, with recourses,(Mishina, 1996).

The stochastic model can be defined and solved in many ways, since stochastic elements or components could enter the system at different stages or levels and, thus each solution is different in concept. Since there are different forms of defining the stochastic programming, the following simple form known, as the stochastic linear programming with recource (SLPWR) will be introduced here. The SLPWR is defined as follows:

$$\text{Min } c^T x$$

s.t $Ax \geq 0, \quad x \geq 0$

Let the resource b be a random vector in R^m, and let y be a recourse vector, which eliminates infeasibility in the constraints. Assume that the domain of interest is a set of linear constraints, and can be stated as:

$$Ax + By = b$$

$$x \geq 0, y \geq 0$$

Let $F = \{x \mid By = -Ax + b, y \geq 0$ are feasible for every possible resource $b\}$, be the non-empty set from which x can be chosen. Let the coefficients, c and d, be the unit costs of choosing x and y, respectively. So for a given x and observed b, the recourse vector y can be chosen to solve the following deterministic problem:

$$\text{Min } \varphi(x, b) = \{d^T y \mid By = -Ax + b, y \geq 0\}.$$

Since b varies with fixed values of x, then the feasibility of the original problem will be violated most of the time. In this case, the expected value of penalties for these violations will be minimized, despite the fact that, it tries not to employ the penalty, but in this case it does. For more details, you are referred to Dantzig (1955) and Lasdon (1979).

There are some important properties of (SLPWR) and can be summarized as follows:

1-A reasonable assumption for source b is that is drawn from a finite set $\{b_{i1},...,b_{ik}\}$ with probabilities $p_1 ,...,p_k$. This observation validates the possibility of scenario decomposition.

2-The size of the problem depends on the number of variables n, the constraints m, and the set of probabilities k. When these terms increase, the size of the problem increases as well. Also the hassle of the computational difficulties will increase too. But with the help of any of the traditional decomposition method, this problem could be solved properly.

Other researchers, Mulvey et al. (1995) have developed a general framework for achieving robustness. They have described the desirable properties of a solution to models, when data are described by a set of scenarios for their values, instead of using point estimates. They have also shown how the robust optimization models would generate robust solutions for several real-world problems, such as the diet problem, power capacity expansion and others as well (Zenios et al., 1994). In this study, the advantages of robust optimization over other traditional approaches (sensitivity analysis and stochastic programming) were discussed. Also some comments about the suitability of parallel and distributed computer architectures for the solution of robust optimization models were made. Next we describe the work of Mulvey et al. (1995) about the robust optimization. The RO problem according to Mulvey et al. (1995), is an extension of the stochastic programming, at least from the active approach view. One feature of (RO) can be raised as the principle of scenario aggregation (Rockafellar and Wets, 1991). Let us use the following stochastic programming problem, which can be solved by using scenario aggregation. Specifically,

(P_1) $\qquad\qquad$ Min $\sum_s p_s z(x,s)$

$$\text{s.t} \qquad x \in \cap_s R_s ,$$

Where uncertainty is structured by a limited number of scenarios $\Omega = \{1,2,...,s\}$; z (x, s) is an error factor, and R_s are the constraints for each scenario.

The scenario aggregation is the main component of a scenario study, where a number of subproblems are derived from an underlying optimization problem. By analyzing the optimal solution for each subproblem, the decision-maker will then be able to choose the most appropriate one to the original problem. Finding the most reasonable weights to the corresponding scenarios could do this. Many probabilistic programming problems can be transformed into the form of (P_1), and this is counted as an advantage for the scenario aggregation. The RO model Mulvey et al. (1995) has two main components:

1-A fixed component that is free of noise in the data, and it is known as the structural component.

2- A component that is subject to noise in its data, and this is known as the control component.

Two types of variables are found in each of these two components:

1-The decision variable, x, which influences the structure of the model and its optimal value is not conditioned on the realization of the uncertain parameters.

2-The control variable, y, which has the role of adjusting the model when disruptions occur in the uncertain situations. Therefore, the linear model in which these two types of variables are used is as follows:

$$(P_2) \qquad \min c^T x + d^T y$$

$$\text{s.t} \quad Ax = b \quad \rightarrow \text{structural constraints}$$

$$Bx + Cy = e \quad \rightarrow \text{control constraints}$$

$$x, y \geq 0 ,$$

where d, B, C and e, are subject to noise. The set of coefficients, $\{d_s, B_s, C_s \text{ and } e_s\}$ of the control constraints associates each scenario $s \in \Omega$ with the probability p_s such that $\sum p_s = 1$. There are two solutions of (P_2):

1-The first is a robust solution in terms of optimality if it stays close to optimal for all scenarios of realizations

2-The second is robust solution in terms of feasibility if it remains almost feasible for all scenarios.

Since it is not possible for most of the times to obtain a solution that is both optimal and feasible for all scenarios, then a tradeoff between them should take place, and it will be as follows:

First include two sets to the (P_2) problem, namely the error vector set and the control variable set, and then we will have the following model:

$$(P_3) \qquad \min \sum p_s (c^T x + d_s^T y_s) + \omega\rho (z_1, \ldots, z_s)$$

$$\text{s.t} \quad Ax = b$$

$$B_s x + C_s y_s + z_s = e_s$$

$$x, y_s \geq 0,$$

Where ω is a goal programming weight, because we are now dealing with a multiobjective perspective, and $\rho(z_1,...,z_s)$ is a feasibility penalty function.

For equality constrained problems, where both positive and negative violations of the control constraints are equally unwanted, then use the penalty function $\rho(z_1,...,z_s) = \sum_{s\in\Omega} p_s z_s^T z_s$.

For inequality control constraints problems, where only positive violations are occurred, then the penalty function $\rho(z_1,...,z_s) = \sum_{s\in\Omega} p_s \max\{0, z_s\}$ (Mulvey, Vanderbei and Zenios, 1995) is used.

Also, other researchers such as Yu (1996) define robustness as the most conservative attitude toward realization of the model against the worst-case scenario. Accordingly, he understood the 0 – 1 knapsack problem as a maximin problem in the following form:

$$\text{Maximize}_x \min_{s\in S} \{\sum v_i^s x_i \mid \sum a_i x_i \le b;\ \ x_i \in \{0,1\},\ i= 1,...,n\}, \qquad (**)$$

where S is a set of scenarios, v_i^s is the value of item i under scenario $s \in S$, a_i is the weight of item i, and b is the capacity of the knapsack. The (**) model can be transformed in another form and the author called it the absolute robustness, which can be expressed as follows:

$\text{Maximize}_x \{\sum v_i x_i \mid \sum a_i x_i \le b;\ \ x_i \in \{0,1\},\ i= 1,...,n\}$, where S is a set of scenarios, v_i^s is the value of item I under scenario $s\in S$, a_i is the weight of item i, and b is the capacity of the knapsack. The (**) model can be transformed to another form and the author called it the absolute robustness, which can be expressed as follows:

$\text{Max}_x \{v_i x_i \mid \sum a_i x_i \le b;\ xi\in \{0,1\},\ i = 1...n\}$.

3.7 Uncertainty According To Elghaoui And Lebret

The least-squares problems with uncertain data were also considered. The coefficient matrices A, b is unknown but bounded. In this study, work was done on minimizing the worst-case residual error using convex second-order cone programming which leads to an algorithm with complexity, similar to that of singular value decomposition of A. This method provides an exact bound on the robustness of solution, and a rigorous way to compute the regularization optimal parameter.

The case where A and b, are rational functions of an unknown but bounded perturbation vector was also considered. Numerical examples were given to illustrate these ideas, and minimizing via Semidefinite Programming upper bounds on the optimal-case residual was shown.

The regularization procedure known as Tikhonov's regularization procedure, was explained as a method for ill-conditioned problems and was interpreted for the unstructured robust least-square problems.

In this unstructured case, both worst-case residual and (unique) robust "L-S" solutions were shown to be continuous. Also similar (weighted) "L-S" interpretations and continuity results were given (Elghaoui and Lebret, 1996).

3.8 Boyd's Approach

The subject of second –order cone programming (SOCP), was shown to be practical to solve some simple robust convex optimization problems, in which, uncertainty in

the data is explicitly accounted for (Boyd 1998). He considers the following optimization problem:

$$\text{Min } c_i^T x$$

$$\text{s.t} \qquad a_i^T x \le b_i , \quad i=1,..., m,$$

in which, there is some uncertainty or variation in the parameters c, a_i , b_i . For simplicity, it is assumed that the coefficients c, b_i are fixed, and that a_i lies in a given ellipsoid, such that:

$$a_i \in U = \{a_i^* + p_i u : \| u \| \le 1 \},$$

where, $P_i = P_i^T \ge 0$. Note that, when P_i is singular, then we will have a flat ellipsoid of dimension equals to the rank of P_i. In the worst –case framework, all constraints must be satisfied for all possible values of the parameter a_i .

That leads to the following robust linear programming:

$$\text{Min } c_i^T x$$

$$\text{s.t} \qquad a_i^T x \le b_i , \quad \forall a_i \in U , \quad i=1,..., m \qquad (3.15)$$

They have shown that, this robust "LP" can be expressed as a SOCP problem in the following:

$$\text{Min } c_i^T x$$

$$\text{s.t} \qquad a_i^{*T} x + \| p_i x \| \le b_i \quad , i= 1,...,m \qquad (3.16)$$

The norm ||.|| acts as a "regularization term", discouraging large x in directions with considerable uncertainty in the parameter a_i. Note that, with $b_i = 0$, the general SOCP, can be interpreted as a robust "LP". From a statistical point of view, the parameters a_i in the robust LP, are the independent gaussian random vectors, with

mean a^*_i and covariance Σ_i. Now for any probability or confidence exceeding $\mu \geq .5$, each constraint must satisfy

$$\text{Prob}\,(a_i^T x \leq b_i) \geq \mu \qquad (3.17)$$

This probability constraint can be expressed as (SOC) constraints. For more detail you are referred to Ben-Tal and Nemirovski (1996) and Oustry et al. (1996). In summary, the problem:

$$\text{Min } c_i^T x$$

$$\text{S.t} \qquad \text{Prob}\,(a_i^T x \leq b_i) \geq \mu, \quad i=1,....,m, \qquad (3.18)$$

can be expressed as the (SOCP):

$$\text{Min } c^T x$$

$$\text{S.t} \qquad a_i^{*T} x + \phi^{-1}(\mu) \,\| \Sigma \delta_i^{1/2} x \,\| \leq b_i, \; i=1...m. \qquad (3.19)$$

This formulation, can be used in applications (e.g. portfolio optimization problem), with another form:

$$\text{Min } p^T x$$

$$\text{S.t} \qquad p^T x + \phi^{-1}(\beta) \,\| \Sigma^{1/2} x \,\| \geq \alpha \qquad (3.20)$$

$$x \geq 0; \quad \sum_{i=1}^{n} x_i = 1,$$

where, α is a given excessive loss or undesired return level, $\beta \leq ½$ is a given maximum probability, and ϕ is the CDF (cumulative distribution function). So (3.20) is to be maximized, that is, to maximize the expected return subject to a bound on the loss risk (Boyd et al., 1998).

Also other researchers have written articles aimed at treating the uncertainty data. They have applied a novel modeling methodology on multi-period asset allocation problem to deal with the issue of uncertainty. They have started their paper with the explanation of the original problem of Dantzig and Infanger (1993), and then they applied to this model the robust counterpart approach. A comparison was made between this resulting model and the one of the portfolio problem. It was concluded that in terms of the risky market, the corresponding standard deviation of the gain in the portfolio value was less than that for the nominal policy by factor of 5 to 8; and by factor of 4.7 to 5 for the stochastic programming policy. In their reported experiments (4 experiments), the RC never resulted in losses, where in the stochastic and in the nominal, the loss was high about 15 to 20%.

As for the expected gain, the stochastic and the nominal policies were the same. In terms of the average gain, the robust policy was almost optimal. The stochastic one was slightly better than the robust policy and far better than the nominal one. So the robust counterpart approach to the portfolio problem from the computational point of view (efforts and time) is incomparably less than the one for multistage stochastic programming (Ben-Tal et al., 1996).

The idea of robust approach to unconstrained optimization problems with discrete variables has been investigated. The orthogonal array based on the Taguchi concept was utilized to arrange the discrete variables. Remember that in Taguchi's method, noise was allowed in the system with some closer tolerance. Through several engineering applications such as the three-bar and the ten-bar truss, the robust design

was performed so that the displacement of a specified node is insensitive to noise. The optimum sectional areas were evaluated by conventional optimization considering constraints; the robust design was applied for the post processing. Then the sensitivity of the robust solution was compared to the conventional optimum one. This step was important because this approach was used as a post processing of constrained problems, although the robust design was used for unconstrained problems.

In their result, the sensitivity of robust design decreased by 14 to 69%. However, the final configuration of the robust design may violate constraints for the constrained problems. This means that, the robust design may be more robust but unacceptable, so an adjustment is needed (Kwon-Hee Lee et al., 1996).

In the late seventies and eighties, the subject of local and global properties of optimization problems was under the microscope of some researchers. They developed an algorithm to help testing these problems and find solutions for them (Quan Zheng, 1985, 1986).

An extension of Zheng's work took place in the early nineties. He first studied the properties of robust sets and robust functions. Minimization problem of a robust function over a robust compact set was considered, using the integral approach. Numerical tests and industrial applications used in earlier studies like the one for (Pans, Wang, Liu and Zheng 1986), showed that the algorithm was effective (Zheng, 1990).

3.9 The Relatioship Between Tractability, Ellipsoids And Ellipsoidal Uncertainties

In this section, we will speak in some detail about the ellipsoidal uncertainty sets, which will cover all cases in which we are interested in, such as:

K-dimensional ellipsoids in R^k,2-Flat ellipsoids and 3- Cylindrical ellipsoids.

The principle of tractability will be also discussed. This is one of the most important parts in constructing the RC problem for some generic programs, such as linear programming, quadratic programming and others as well. One of the reasons why it is important is because the (RC) problem will be computationally tractable (*Grötschel* et al., 1988). So all we need to efficiently solve a minimizing problem with linear objective over a convex set is an efficient separation for the set oracle (*Grötschel* et al., 1988). An efficient separation oracle for U (uncertainty set) implies an efficient inclusion oracle. This in turn implies an efficient separation oracle for G_u, and now this also leads to computational tractability of the robust problem.

Thus, according to the tractability principles, all reasonable closed convex uncertainty sets U, lead to computationally tractable problem. So the set U defined by finitely many convex constraints $g_i(A) \leq 0$, can have an efficient separation oracle by verifying whether $g_i(A) \leq 0$ for all i given A. If this is the case, then $A \in U$, otherwise the sub-gradient of the violated constraint (taken at A) is a separator of A and U.

This argument of tractability is based on the assumption, that the problem is practically solvable, which requires a simple analytical structure for the RC problem, which in turn requires U to be relatively simple.

On the other hand, it should not be forgotten either, that making the geometry of U too simple would cause to lose flexibility in terms of ability to model diverse actual or practical uncertainties. Indeed, when U, is restricted to be an ellipsoidal uncertainty or an intersection of finitely many ellipsoids – sets given by convex quadratic inequalities, then a reasonable solution to conflicting goals can be obtained with the help of that restricted set of uncertainty U (Ben-Tal & Nemirovski 1996).

Some of the arguments, which are in favor of the importance of ellipsoids, are as follows:

1-The robust problem associated with an ellipsoidal U possesses a very well analytical structure, which is nothing but the known "conic quadratic program"; a program with linear objective and constraints of the type: $a^T_i x + \alpha_i \geq \| B^T_i x + b_i \|$, i = 1.....M.

Where α_i are fixed real, a_i and b_i are fixed vectors, B_i are fixed matrices of proper dimensions, and $\| \bullet \|$ represents the Euclidean norm. So this RC conic quadratic program, even if it is large-scale with ellipsoidal uncertainty, can practically be solved, with the help of the techniques of interior point methods. For more detail see the research report of Ben-tal & Nemirovski (1996).

2-Ellipsoidal uncertainty sets form relatively wide family of polytopes (bounded sets given by finitely many linear inequalities), and can be used for cases of complicated convex sets.

3-An ellipsoid is given by a moderate data size; therefore it provides a convenient way to represent ellipsoidal uncertainty conveniently as input.

4-In some important cases, these are statistical reasons, which show the need of ellipsoidal uncertainty

One should know that, even ellipsoids are in different forms, and used for different cases. Consider for example the flat ellipsoids in the space $E = R^{m \times n}$ of data matrices A. Such an ellipsoid corresponds to the case where partial uncertainty exists. This means that, the entries in the data matrix satisfy a number of known linear equations for sure, and thus some of the entries in A are certain.

The other type of ellipsoids is the cylindrical ellipsoids, which are sets of the types "sum of a flat ellipsoid" and a linear subspace. Imposing several ellipsoidal restrictions on the matrix A will create these sets. Note that each of these restrictions deal with part of the entries. So the uncertain set U will have an upper and lower bounds on the entries of the matrix. This is prescribed as an intersection of m n ellipsoidal cylinders (Ben-Tal & Nemirovski, 1996).

So now let us see how we can cover all these cases from a mathematical point of view:

Let us define an ellipsoid in R^K as a set of the following form:

$U = \{\Pi(u): \| Qu \| \leq 1$, where Q is an M×L matrix, $u \rightarrow \Pi(u)$ is an affine embedding of certain R^L into R^K.

With this definition, all cases will be covered. The dimensions M,L ,and K and the singularity of the matrix Q, can help in determining the type of ellipsoid we have. We consider the following cases:

1-L=M=K and Q is nonsingular, and then we have the K- dimensional ellipsoids in R^K.

2-L=M<K and Q is nonsingular, then we have the flat ellipsoids.

3-If Q is singular, and then the ellipsoidal cylinders are in place.

We conclude this section by summarizing the following:

U is an ellipsoidal uncertainty $\in R^{m \times n}$, if:

U is given as an intersection of finitely many ellipsoids, and can mathematically be expressed as

1- $U = \prod_{l=0}^{K} Y(\Pi_l . Q_l)$, with data Q_l and Π_l.

U is bounded.

There is at least one matrix $A \in U$, which belongs to the relative interior of every ellipsoids U_i , i=1...K, $\forall\, l \leq K\ \exists\, u_l$ such that $A = (\Pi_l u^l)$ and $| Q_l u_l | < 1$.

3.10 Summary of Robust Optimization General Work

1-The (RC) of an uncertain "LP" problem with ellipsoidal or intersection of ellipsoids uncertainty is an explicitly conic quadratic program.

2-The (RC) of an uncertain convex quadratically constrained quadratic programming problem with ellipsoidal uncertainty is an explicit semi-definite program, while a general-type intersection-ellipsoidal uncertainty leads to an NP hard (RC).

3 - The conic quadratic problem will have the same result as in 2 under some minor restrictions.

4- In the case of uncertain semi-definite problems with a general –type ellipsoidal uncertainty, the (RC) is NP-hard.

5- Derivation of an explicit form of the (RC) of an affine parameterized uncertain problem, such as the geometric problem with uncertain coefficients of the monomials, and develops a specific saddle point form of the (RC) for these problems (Ben-Tal & Nemirovski, 1998).

The number of researchers who have studied the robust optimization approach and used mathematical programming are not that many, such as in the case of the study of convex programming with set-inclusive constraints and applications to inexact linear programming (A.L.Soyster, 1973). From his work, other studies were originated such as the work of (Singh, 1982), which is a continuation of Soyster's work. Another extension of Soyster's work was done by Falk (1976). Also some have developed the robust counterpart approach, mainly as applied to uncertain semi-definite programming (Oustry et al., 1996).

The robust discrete optimization with some applications was investigated theoretically (Kouvelis and Yu 1997).

Chapter 4

Robust Optimization in Support Vector Machine Learning

4.1 Uncertainty and Optimization

In this section, an explanation of the relationship between the idea of robust optimization and support vector machines will be given. Why robust optimization idea becomes interesting to too many researchers? The reason is that, there are some contradictions between the real-world data, and the realm of traditional mathematical programming; therefore reconciliation between them becomes necessary. When operation researchers try to construct a model of a real-world system, they always find incomplete, noisy or uncertain data. On the other hand, in the world of mathematical programming, it is assumed that the model is deterministic something that does not hold generally in the real world. It has been found that large error bounds arise when one solves mean value problems (Birge 1982). Other inadequate solutions for worst-case problems were also observed. An approach to incorporate uncertainty in traditional mathematical programming is to use stochastic programming (Zenios et al., 1995). Another way is to use the idea of sensitivity analysis, which was employed by some management scientists; the objective is to

uncover the facts about the influence of data uncertainty on the model's construction or recommendations (Zenios et al., 1995). Thus, and under these legitimate pressures, creating model formulations that, by design, will lead to solutions, which are less sensitive to the model data, than the usual existing methods, becomes a necessity. Another approach is to introduce the interval coefficients methodology in LP formulations. It is based on the fact, that some or all of the coefficients of the LP are described as intervals. Then, best and worst optimum for the model are found along with the point setting of the interval coefficients that yield these two extremes (Chinneck and Ramadan, 2000). Other approaches were also introduced, such as stochastic linear programming and sensitivity analysis (Mulvey et al., 1995) that describe a reactive approach to controlling uncertainty. The sensitivity of a solution to changes in the input data is measured, but it offers no tool by which this sensitivity can be controlled. Now in the robust optimization approach, the uncertainty is considered in problem design itself, through some mathematical formulations. It generates solutions that are less sensitive to realizations of the model data. Therefore the robust optimization approach has some advantages over the above-mentioned approaches (Mulvey et al., 1995).

4.2 Pattern Recognition

Let us consider the following SVM optimization problem in its primal form,

$$\text{Min} \parallel w \parallel^2$$

S.t

$$Aw + \tilde{b} \geq 0,$$

where $A = [(y_1 x_1)^T, \ldots (y_l x_l)^T]$, $\tilde{b}^T = [(y_1 b - 1, \ldots, y_l - 1]$.

The data of the above problem, A and $\tilde{b}$ sometimes are not completely known. In this case, we are generalizing the uncertainty to both x and y, ultimately to the vector $\tilde{b}$. What is known here is the domain U in the space of data an "uncertainty set", which contains the actual unknown data. Since we need to satisfy the separation constraints, knowing only that the data belongs to an uncertainty set U, the only good way to meet the requirements is to be restricted to with robust feasible candidate solutions w, which satisfy all possible realizations of the uncertain constraints, i.e, such that,

$$Aw + \tilde{b} \geq 0 \; \forall \; (A, \tilde{b}) \in U.$$

The robust counterpart is the optimization problem:

$$\text{Min} \; \| w \|^2$$

S.t

$$Aw + \tilde{b} \geq 0 \; \forall \; (A, \tilde{b}) \in U.$$

It should be noticed that, it is a usual certain optimization problem, but not with linear constraints (Trafalis, 1999). Of course its structure will depend on the geometry of the uncertainty set U and can be very difficult. We could for example specify the uncertainty set in some R^n as an ellipsoid; the image of the unit Euclidean ball under an affine mapping or, generally, as an intersection of finitely many ellipsoids. In cases where the ellipsoid uncertainty is a simple one, the data of the i-th inequality

constraint $a_i^T w + \tilde{b}_i \geq 0$, are allowed to run independently of each other through respective ellipsoids. So assume that the uncertainty set is as follows:

$U = \{ (a_1, \tilde{b}_1, \ldots\ldots, a_l, \tilde{b}_l) \mid \exists\, u_i \text{ such that } \{u_i^T u_i \leq 1\}_{i=1}^{l}$,where $\begin{pmatrix} a_i^* \\ \tilde{b}_i^* \end{pmatrix} + u_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$

$i=1,\ldots,l \}$ where, $\begin{pmatrix} a_i^* \\ \tilde{b}_i^* \end{pmatrix}$ are the nominal data and u_i $i = 1,\ldots,l$, represent the data perturbations. There are some restrictions so that these perturbations can be enforced to vary in spheres and these can be expressed as $u_i^T u_i \leq 1$.Now w is robust feasible if and only if for every i =1…l we have:

$$0 \leq \min_{u_i : u_i^T \leq 1} [a_i^T [u] w + \tilde{b}_i [u], \begin{pmatrix} a_i[u] \\ \tilde{b}_i[u] \end{pmatrix} = \begin{pmatrix} a_i^* \\ b_i^* \end{pmatrix} + u_i]$$

$$= (a_i^*)^T w + \tilde{b}_i^* + \min_{u_i : u_i^T u_i \leq 1} u_i^T \begin{pmatrix} w \\ 1 \end{pmatrix}$$

$$= (a_i^*)^T w + \tilde{b}_i^* - \left\| \begin{pmatrix} w \\ 1 \end{pmatrix} \right\|_2$$

Therefore the SVM optimization problem is

$$\text{Min} \| w \|^2$$

S.t

$$\left\| \begin{pmatrix} w \\ 1 \end{pmatrix} \right\|_2 \leq (a_i^*)^T w + \tilde{b}_i^*$$

The above optimization problem belongs to the class of second order cone programming problems (SOCP) (Boyd et al., 1998).

4.3 Pattern Classification Training Examples

4.3.1 Introduction

In this section, two examples will be given. The first deals with zero uncertainty, which means that the data are considered to be precise. The second example deals with uncertainty, that is the input data will be formulated with uncertainty; on the other hand, the uncertainty will be given different values to observe how the hyperplane will change accordingly. In constructing this example, L_1 norm will be used to compute the distance between the training points and the separating hyperplane. Equation (2.13) represents this distance which can be stated as $d_1(H^+, H^-) = \frac{|(b+1)-(b-1)|}{\|w\|_\infty} = \frac{2}{\max_j |w_j|}$, j = 1,...,d.

Therfore we want to maximize $d_1(H^+, H^-)$, which is equivalent to minimize $\max|w_j|$. This can be done through linear optimization if an auxiliary variable α, and two sets of constraints are added as $\alpha \geq |w_j|$. As explained in section 2.3.1, since the L_1 norm is used to find the distance, then the learning problem in the example can be formulated as a linear programming problem as it has been stated in section two. Remember that the two hyperplanes are H^+ and H^-, which are equal to: $H^+ = w.x_i + b = 1$, and $H^- = w.x_i + b = -1$ respectively. Hence, the example can be stated as follows:

Let the training data consist of four points, each of which is represented by a vector $x_i \in \Re^d$, where i = 1,2,3,4, and assign to each of those four points a label y_i with $y_i \in \{-1, +1\}$. This means that the separation will be between two classes only. Hence, constructing a learning machine which separate the data of two training sets in pattern space finds its geometrical

equivalent in identifying a hyperplane that will separate the data points labeled by $y_i = +1$, from the data points labeled by $y_i = -1$. Hence, two cases will be considered in formulating the problem: the first is without considering uncertainty (uncertainty is zero), and the second is considering the uncertainty.

4.3.2 Precise Data Case (No Uncertainty)

The formulation of the "LP" problem without having uncertainty is as follows,

$$\min \alpha \qquad (4.1)$$

S.t

$$w \cdot x^i + b \geq 1 \qquad i \in P$$

$$w \cdot x^i + b \leq -1 \qquad i \in N$$

$$-\alpha \leq w_j \leq \alpha \qquad j = 1,2,$$

where P is the set of positive examples and N is the set of negative examples. Remember, that w_j and x^i are vectors, and $w.x^i$ is the dot product of these two vectors. Now, let us use the AND function to illustrate the above formulation (table 1 describes the input-output relationship for the AND function).

x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Table 1- AND function

Using the data in table 1, (4.1) will have the following form:

$$\min \alpha \tag{4.2}$$

S.t

$$w_1 + w_2 + b \geq 1$$

$$w_1 - w_2 + b \leq -1$$

$$-w_1 + w_2 + b \leq -1$$

$$-w_1 - w_2 + b \leq -1$$

$$\alpha \geq w_1, \quad \alpha \geq w_2$$

$$\alpha \geq -w_1, \quad \alpha \geq -w_2$$

We bring problem (4.2), in a form recognizable by Matlab optimization toolbox

$$\min \alpha \tag{4.3}$$

S.t

$$-w_1 - w_2 - b + 1 \leq 0$$

$$w_1 - w_2 - b + 1 \leq 0$$

$$-w_1 + w_2 + b + 1 \leq 0$$

$$-w_1 - w_2 + b + 1 \leq 0$$

$$\alpha \geq w_1, \quad \alpha \geq w_2$$

$$\alpha \geq -w_1, \quad \alpha \geq -w_2$$

4.3.3 Sensitivity Analysis Case

In this model, we are considering a traditional sensitivity analysis formulation; where perturbations are added to each data point along axes x_1, x_2 respectively. The perturbations are of magnitude $\sqrt{\eta}$, where $\eta \geq 0$. Therefore we get the following formulation:

$$\min \alpha$$

$$\text{S.t} \quad (w_1, w_2).\begin{pmatrix} 1+\sqrt{\eta} \\ 1+\sqrt{\eta} \end{pmatrix} + b \geq 1$$

$$(w_1, w_2).\begin{pmatrix} 1+\sqrt{\eta} \\ -1+\sqrt{\eta} \end{pmatrix} + b \leq -1$$

$$(w_1, w_2).\begin{pmatrix} -1+\sqrt{\eta} \\ 1+\sqrt{\eta} \end{pmatrix} + b \leq -1$$

$$(w_1, w_2).\begin{pmatrix} -1+\sqrt{\eta} \\ -1+\sqrt{\eta} \end{pmatrix} + b \leq -1$$

$$\alpha \geq w_1, \quad \alpha \geq w_2$$

$$\alpha \geq -w_1, \quad \alpha \geq -w_2, \ \eta \geq 0.$$

The final construction of the above formulation will be as follows,

$$\min \alpha \tag{4.4}$$

S.t

$$(1+\sqrt{\eta})\, w_1 + (1+\sqrt{\eta})\, w_2 + b - 1 \geq 0$$

$$(1+\sqrt{\eta})\, w_1 + (-1+\sqrt{\eta})\, w_2 + b + 1 \le 0$$

$$(-1+\sqrt{\eta})\, w_1 + (1+\sqrt{\eta})\, w_2 + b + 1 \le 0$$

$$(-1+\sqrt{\eta})\, w_1 + (-1+\sqrt{\eta})\, w_2 + b + 1 \le 0$$

$$\alpha \ge w_1, \quad \alpha \ge w_2$$

$$\alpha \ge -w_1, \quad \alpha \ge -w_2$$

Of course, when α equals to zero, then we will have the precise data case. The resulting hyperplanes of this sensitivity case are shown in the experimental result (table 4 and figure2). Note that η describes the magitude of uncertainty.

4.3.4 Robust Data Case (with bounded uncertainty)

In this case, the formulation of the problem will lead to a special kind of mathematical programming problem (Boyd et al., 1997) known as a conic quadratic programming. So let us use the AND function as above. Then:

$$\tilde{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1+z_1 \\ 1+z_2 \end{pmatrix}, \tag{4.5}$$

$$\tilde{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1+z_1 \\ -1+z_2 \end{pmatrix},$$

$$\tilde{x}_3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} -1+z_1 \\ 1+z_2 \end{pmatrix},$$

$$\tilde{x}_4 = \begin{pmatrix} -1 \\ -1 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} -1+z_1 \\ -1+z_2 \end{pmatrix},$$

where $z = (z_1, z_2)$ refers to the uncertainty vector. Now, let us substitute these variables in (4.1), and so the problem will have the following form:

$$\min \alpha$$

S.t

$$(w_1 \quad w_2).\begin{pmatrix} 1+z_1 \\ 1+z_2 \end{pmatrix}+b \geq 1$$

$$(w_1 \quad w_2).\begin{pmatrix} 1+z_1 \\ -1+z_2 \end{pmatrix}+b \leq -1$$

$$(w_1 \quad w_2).\begin{pmatrix} -1+z_1 \\ 1+z_2 \end{pmatrix}+b \leq -1$$

$$(w_1 \quad w_2).\begin{pmatrix} -1+z_1 \\ -1+z_2 \end{pmatrix}+b \leq -1$$

$-\alpha \leq w_1 \leq \alpha$, $-\alpha \leq w_2 \leq \alpha$, $\forall$ z such that $\|z\| \leq \sqrt{\eta}$.

Let us manipulate to obtain a simpler mathematical form:

$$\min \alpha \qquad (4.6)$$

S.t

$$w_1.z_1 + w_2.z_2 + w_1 + w_2 + b - 1 \geq 0$$

$$-w_1.z_1 - w_2.z_2 - w_1 + w_2 - b - 1 \geq 0$$

$$-w_1.z_1 - w_2.z_2 + w_1 - w_2 - b - 1 \geq 0$$

$$-w_1.z_1 - w_2.z_2 + w_1 + w_2 - b - 1 \geq 0$$

$$\alpha \geq w_1, \quad \alpha \geq w_2$$

$$\alpha \geq -w_1, \quad \alpha \geq -w_2 \ . \qquad \forall \text{ z such that } \|z\| \leq \sqrt{\eta}$$

The above problem can be written equivalently as:

$$\min \alpha \tag{4.7}$$

S.t

$$\langle w,z\rangle + w_1 + w_2 + b - 1 \geq 0$$

$$-\langle w,z\rangle - w_1 + w_2 - b - 1 \geq 0$$

$$-\langle w,z\rangle + w_1 - w_2 - b - 1 \geq 0$$

$$-\langle w,z\rangle + w_1 + w_2 - b - 1 \geq 0$$

$$\alpha \geq w_1, \quad \alpha \geq w_2$$

$$\alpha \geq -w_1, \quad \alpha \geq -w_2. \qquad \forall z \text{ such that } \|z\| \leq \sqrt{\eta}\,.$$

Notice, that <w,z> is the dot product of the two vectors w, z that appears in the constraints of (4.7). Now we are going to consider the worst case of feasibility by minimizing the dot product in each of the above constraints in (4.7), subject to the norm of the assumed uncertainty z, which has the value of less or equal to η, which mathematically can be expressed as:$\| z \|^2 \leq \eta$. Hence, the problem becomes:

$$\min \langle w, z \rangle \tag{4.8}$$

S.t

$$\|z\| \leq \sqrt{\eta}$$

Where η is a positive number. Let us make some algebraic simplification to see if a more suitable formulation of the original problem can be obtained, so that it can be solved in a simpler manner: By using Cauchy's Schwarz inequality we have:

$$|< w, z >| \leq \|w\|.\|z\| \Rightarrow - \|w\|.\|z\| \leq < w, z > \leq \|w\|.\|z\| \qquad (4.9)$$

By the constraints of (4.8) and (4.9), we have:

$$- \eta^{1/2}\|w\| \leq < w, z > \leq \eta^{1/2}\|w\| \qquad (4.10)$$

Hence, the minimum of the dot product of these two vectors w and z will be:

$$<w, z> = -\sqrt{\eta}\,\|w\| \qquad (4.11)$$

Finally, substitute (4.11) in problem (4.7), and then the construction of the robust optimization problem considering uncertainty will be as follows:

$$\min \alpha \qquad (4.12)$$

S.t

$$-\sqrt{\eta}\,\|w\| + w_1 + w_2 + b - 1 \geq 0$$

$$\sqrt{\eta}\,\|w\| - w_1 + w_2 - b - 1 \geq 0$$

$$\sqrt{\eta}\,\|w\| + w_1 - w_2 - b - 1 \geq 0$$

$$\sqrt{\eta}\,\|w\| + w_1 + w_2 - b - 1 \geq 0$$

$$\alpha \geq w_1, \quad \alpha \geq w_2$$

$$\alpha \geq -w_1, \quad \alpha \geq -w_2.$$

The above problem is a conic quadratic programming problem (Boyd et al 1998), and can be solved in polynomial time using interior point methods.

4.4 Robust "LP" Approach to Classification

4.4.1 Introduction

The Support Vector Machines approach is very flexible. The concept of maximizing the margin, with the help of kernelization, and duality can be used in the inference problem. This flexibility will help to produce a linear program (LP) model for classification, simply by changing the norm used to measure the margin. Thus, rather than using quadratic programming, it is possible to derive a kernel classifier which will lead to a linear programming (LP) one. So in the following primal SVM formulation

$$\begin{aligned} &\min_{w,b,z} 1/2\|w\|^2 + C\sum_{i=1}^{l} z_i \\ &st \\ &\quad y_i(w.x_i + b) + z_i \geq 1 \\ &\quad z_i \geq 0 \quad ,i = 1,..m, \end{aligned} \tag{4.13}$$

we maximize the margin between the supporting planes for each class where the distance can be measured by using the 2-norm. This is equivalent to minimizing the 2-norm of w. Instead of using the above quadratic programming (QP), one can change the model to maximize the margin by using the infinity norm, and then

minimizing the 1-norm of w (the sum of the absolute values of the components of w) is equivalent to:

$$\begin{aligned} &\min_{w,b,z} \|w\|_1 + C\sum_{i=1}^{l} z_i \\ &st \\ &\quad y_i(w.x_i + b) + z_i \geq 1 \\ &\quad z_i \geq 0 \quad ,i = 1,...,m, \end{aligned} \tag{4.14}$$

where the parameter C is the trade off between minimizing the misclassifications and maximizing the margin.

4.4.2 Robust Counter part of (4.14)

Let $x_i = \tilde{x}_i + u_i$, such that $\|u_i\| \leq \sqrt{\eta}$, where u_i is the uncertainty and $\tilde{x}_i$ is the center of the uncertainty sphere. Now substitute the new value of x in the constraints of (4.14), and then we will have the following:

$$\begin{aligned} &y_i(\langle w, \tilde{x}_i + u_i \rangle + b) + z_i \geq 1 \qquad \Leftrightarrow \\ &y_i(\langle w, \tilde{x}_i \rangle + \langle w, u_i \rangle + b) + z_i \geq 1 \quad \Leftrightarrow \\ &y_i\langle w, \tilde{x}_i \rangle + y_i\langle w, u_i \rangle + y_i b + z_i \geq 1 \\ &z_i \geq 0 \ , i = 1,...,l \ , \ \forall \ u_i \text{ such that } \|u_i\| \leq \sqrt{\eta} \, . \end{aligned} \tag{4.15}$$

Note that w is robust feasible if and only if for every i=1,...,l:

$$\min_{\|u_i\| \leq \sqrt{\eta}} y_i\langle w, \tilde{x}_i \rangle + y_i\langle w, u_i \rangle + y_i b + z_i \geq 1.$$

Therefore we need to minimize the dot product of w and u_i subject to $\|u_i\| \le \sqrt{\eta}$. Thus, we need to solve the following problem:

$$\min \langle w, u_i \rangle$$
$$st$$
$$\|u_i\| \le \sqrt{\eta}.$$

Using Cauchy's Schwarz inequality as in eq. (4.10) the minimum of $\langle w, u_i \rangle$ is equal to $-\sqrt{\eta}\|w\|$. By substituting this minimum in (4.15). We have:

$$\begin{aligned} y_i \langle w, \tilde{x}_i \rangle - y_i \sqrt{\eta}\|w\| + y_i b + z_i \ge 1 \\ z_i \ge 0,\ i = 1,\ldots,l. \end{aligned} \tag{4.16}$$

Therefore we need to solve the following robust formulation of problem (4.14):

$$\min_{w,b,z} \|w\|_1 + C\sum_{i=1}^{l} z_i$$

s.t (RO)

$$y_i \langle w, \tilde{x}_i \rangle - y_i \sqrt{\eta}\|w\| + y_i b + z_i \ge 1$$
$$z_i \ge 0, i = 1,\ldots,l.$$

The solution of the above problem provides both sparsity and robustness. Next we discuss the case where the data are nonlinear separable in the input space.

4.5 Kernelization

It is known that $w = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i$, where $\alpha_i \ge 0$ and $y_i = \pm 1$ (R).

Then $\|w\|^2$ can be expressed in terms of the training data as follows:

$$\|w\|^2 = \langle w, w \rangle = \langle \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i,\ y_j \alpha_j \mathbf{x}_j \rangle = \langle\ y_1 \alpha_1 \mathbf{x}_1 + y_2 \alpha_2 \mathbf{x}_2 + \ldots + y_l \alpha_l \mathbf{x}_l,\ y_1 \alpha_1 \mathbf{x}_1 +$$

$$y_2 \alpha_2 \mathbf{x}_2 + \ldots + y_l \alpha_l \mathbf{x}_l \rangle = \langle \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i,\ y_1 \alpha_1 \mathbf{x}_1 \rangle + \langle \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i,\ y_2 \alpha_2 \mathbf{x}_2 \rangle + \langle \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i,$$

$$y_j \alpha_j \mathbf{x}_j \rangle + \ldots + \langle \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i,\ y_l \alpha_l \mathbf{x}_l \rangle =$$

$$\sum_{i=1}^{l} y_i y_1 \alpha_i \alpha_1 \langle \mathbf{x}_i, \mathbf{x}_1 \rangle + \sum_{i=1}^{l} y_i y_2 \alpha_i \alpha_2 \langle \mathbf{x}_i, \mathbf{x}_2 \rangle + \ldots + \sum_{i=1}^{l} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^{l} y_i y_l \alpha_i \alpha_l \langle \mathbf{x}_i, \mathbf{x}_l \rangle$$

$$= \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad (T).$$

By defining $y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \tilde{k}(\mathbf{x}_i, \mathbf{x}_j)$, and substituting it in (T) we have:

$$\|w\|^2 = \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j \tilde{k}(x_i, x_j) = \alpha^t \tilde{k} \alpha \qquad \Leftrightarrow \|w\| = \sqrt{\alpha^t \tilde{k} \alpha} \quad .(S)$$

It is well known (Cristianini and Taylor, 2000), that $\tilde{k}$ is a positive definite matrix. If we make the necessary substitutions by using (R), (T), (S), the constraints of (RO) becomes:

$$y_i \langle \sum_{j=1}^{l} y_j \alpha_j x_j, \tilde{x}_i \rangle - \sqrt{\eta} y_i \sqrt{\alpha^t \tilde{k} \alpha} + y_i b + z_i \geq 1 \Leftrightarrow$$

$$z_i \geq 0, i = 1, \ldots, l.$$

Since $\|w\|_1 = \left\|\sum_{i=1}^{l} \alpha_i y_i x_i\right\| \le \sum_{i=1}^{l} |y_i| \alpha_i \|x_i\|_1 = \sum_{i=1}^{l} \alpha_i \|x_i\|_1$ and assuming that $\|x_i\|_1$ is bounded, minimizing $\|w\|_1$ is equivalent to minimizing $\sum_{i=1}^{l} \alpha_i$. Thus (RO) becomes as follows:

$$\begin{aligned} &\min_{\alpha,b,z} \sum_{i=1}^{l} \alpha_i + C \sum_{i=1}^{l} z_i \\ &st \\ &\quad y_i \sum_{j=1}^{l} y_j \alpha_j \langle x_j, \tilde{x}_i \rangle - \sqrt{\eta} y_i \sqrt{\alpha^t \tilde{k} \alpha} + y_i b + z_i \ge 1 \\ &\quad \alpha_i \ge 0, \quad z_i \ge 0, i = 1,..l \end{aligned} \tag{4.17}$$

Now $x_j = \tilde{x}_j + u_j$ and $\langle \mathbf{x}_j, \tilde{\mathbf{x}}_i \rangle = \langle \tilde{\mathbf{x}}_j + \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle = \langle \tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_i \rangle + \langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle = k(\tilde{x}_j, \tilde{x}_i) + \langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle$, where $\langle \tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_i \rangle = k(\tilde{x}_j, \tilde{x}_i)$. Since we consider a robust counterpart of the above constraints, we replace $\langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle$ with its minimum value. Specifically,

$\min \langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle = \|\mathbf{u}_j\| \cdot \|\tilde{\mathbf{x}}_i\| = -\sqrt{\eta} \|\tilde{\mathbf{x}}_i\|$.

By considering the above substitutions, the robust formulation of (4.14) will be as follows:

$$\min \sum_{i=1}^{l} \alpha_i + C \sum_{i=1}^{l} z_i$$

St (4.18)

$$-\sqrt{\eta} y_i \sqrt{\alpha^t \tilde{k} \alpha} + y_i \sum_{j=1}^{l} y_j \alpha_j \left[k(\tilde{x}_j, \tilde{x}_i) - \sqrt{\eta} \|\tilde{\mathbf{x}}_i\|\right] + y_i b + z_i \ge 1$$

$z_i \geq 0,\ \alpha_i \geq 0 \quad i = 1,\ldots,l$, where $\|\tilde{\mathbf{x}}_j\| = \sqrt{k(x_j, x_j)}$.

Now we can find the discriminant function for the optimal hyperplane to be:

$$f(x,\alpha,b,u) = \sum_{i=1}^{l} y_i \alpha_i \langle \mathbf{x_i}, \mathbf{x} \rangle + b$$

$$= \sum_{i=1}^{l} y_i \alpha_i \langle \tilde{\mathbf{x}}_i, +\mathbf{u_i}, \mathbf{x} \rangle + b$$

$$= \sum_{i=1}^{l} (y_i \alpha_i \langle \tilde{\mathbf{x}}_i, \mathbf{x} \rangle + \sum y_i \alpha_i \langle \mathbf{u_i}, \mathbf{x} \rangle) + b = \sum_{i=1}^{l} y_i \alpha_i \langle \tilde{\mathbf{x}}_i, \mathbf{x} \rangle + \sum_{i=1}^{l} y_i \alpha_i \langle \mathbf{u_i}, \mathbf{x} \rangle + b$$

Since we consider the worst case, we replace $\langle \mathbf{u_i}, \mathbf{x} \rangle$ by its minimum value $-\sqrt{\eta}\,\|\mathbf{x}\|$. Therefore by denoting $\hat{f}(x,\alpha,b)$ (robust discriminant function), we have:

$\hat{f}(x,\alpha,b) = \sum_{i=1}^{l} (y_i \alpha_i k(\tilde{x}_i, x) - y_i \alpha_i \sqrt{\eta}\|\mathbf{x}\|_1) + b$ or equivalently

$$\hat{f}(x,\alpha,b) = \sum_{i=1}^{l} y_i \alpha_i k(\tilde{x}_i, x) - \sqrt{\eta}\|\mathbf{x}\|_1 \sum_{i=1}^{l} y_i \alpha_i + b\,. \qquad (4.19)$$

4.5.1 Non-Linear Case (XOR Problem) with Precise Data

x_1	x_2	Y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	1

XOR- Problem

In this problem, we need to find the optimal separating hyperplane that classifies the above data without error. It is not possible to solve this problem with a linear decision boundary. However, a polynomial decision boundary of order 2 can separate these data by using the general equation (4.18) in section 4.6. Notice that since we are using the precise data the uncertainty η equals to zero. Therfore (4.18) becomes as follows:

$$\min \sum_{i=1}^{l} \alpha_i + C \sum_{i=1}^{l} z_i$$

St

$$y_i \left(\sum_{j=1}^{l} y_j \alpha_j [k(x_j, x_i)] + b \right) + z_i \geq 1 \qquad (4.20)$$

$$z_i \geq 0, \alpha_i \geq 0 \quad i = 1,..., l \; ; \; j = 1,..., l$$

Since the XOR problem has four input data points then (4.20) can be formulated accordingly, and it will look like the following:

$$\min[\alpha_1+\alpha_2+\alpha_3+\alpha_4+c(z_1+z_2+z_3+z_4)] \tag{4.21}$$

s.t

$$y_1[y_1\alpha_1k(x_1,x_1)+y_2\alpha_2k(x_1,x_2)+y_3\alpha_3k(x_1,x_3)+y_4\alpha_4k(x_1,x_4)+b]+z_1\geq 1$$

$$y_2[y_1\alpha_1k(x_2,x_1)+y_2\alpha_2k(x_2,x_2)+y_3\alpha_3k(x_2,x_3)+y_4\alpha_4k(x_2,x_4)+b]+z_2\geq 1$$

$$y_3[y_1\alpha_1k(x_3,x_1)+y_2\alpha_2k(x_3,x_2)+y_3\alpha_3k(x_3,x_3)+y_4\alpha_4k(x_3,x_4)+b]+z_3\geq 1$$

$$y_4[y_1\alpha_1k(x_4,x_1)+y_2\alpha_2k(x_4,x_2)+y_3\alpha_3k(x_4,x_3)+y_4\alpha_4k(x_4,x_4)+b]+z_4\geq 1$$

$$z_i\geq 0,\ \alpha_i\geq 0 \quad i=1,\ldots,l\ ;\ j=1,\ldots,l$$

As for the kernel, we have used the polynomial one with degree of 2 ($k_{ij} = [(x_i .x_j) +1]^2$). This inner product kernel is represented as a 4 by 4 matrix since we are using 4 points only as data points. Therefore

$$k=\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix}.$$

Now let us show how to find the elements of row number one:

$$k_{11} = (x_1, x_1) = \left[(1 \quad 1) \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 1 \right]^2 = 9$$

$$k_{12} = (x_1, x_2) = \left[(1 \quad 1) \begin{pmatrix} 1 \\ -1 \end{pmatrix} + 1 \right]^2 = 1$$

$$k_{13} = (x_1, x_3) = \left[(1 \quad 1) \begin{pmatrix} -1 \\ 1 \end{pmatrix} + 1 \right]^2 = 1$$

$$k_{14} = (x_1, x_4) = \left[(1 \quad 1) \begin{pmatrix} -1 \\ -1 \end{pmatrix} + 1 \right]^2 = 1.$$

Now we do the same thing for every element in the above matrix and hence the kernel matrix k_{ij} will be as follows:

$$k = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}.$$

By substituting the values of k_{ij} and y_j in (4.21) we then get the following problem:

$$\min[\alpha_1+\alpha_2+\alpha_3+\alpha_4+c(z_1+z_2+z_3+z_4)] \tag{4.22}$$

s.t

$$1.[(1.\alpha_1.9)+(-1.\alpha_2.1)+(-1.\alpha_3.1)+(1.\alpha_4.1)+b]+z_1 \geq 1$$

$$-1.[(1.\alpha_1.1)+(-1.\alpha_2.9)+(-1.\alpha_3.1)+(1.\alpha_4.1)+b]+z_2 \geq 1$$

$$-1.[(1.\alpha_1.1)+(-1.\alpha_2.1)+(-1.\alpha_3.9)+(1.\alpha_4.1)+b]+z_3 \geq 1$$

$$1.[(1.\alpha_1.1)+(-1.\alpha_2.1)+(-1.\alpha_3.1)+(1.\alpha_4.9)+b]+z_4 \geq 1$$

$$z_i \geq 0,\ \alpha_i \geq 0 \quad i=1,...,l\ ;\ j=1,...,l$$

The final construction of (4.22) after some manipulation will be as follows:

$$\min[\alpha_1+\alpha_2+\alpha_3+\alpha_4+c(z_1+z_2+z_3+z_4)] \tag{4.23}$$

s.t

$$1.[9\alpha_1-\alpha_2-\alpha_3+\alpha_4+b]+z_1-1 \geq 0$$

$$-1.[-\alpha_1+9\alpha_2+\alpha_3-\alpha_4-b]+z_2-1 \geq 0$$

$$-1.[-\alpha_1+\alpha_2+9\alpha_3-\alpha_4-b]+z_3-1 \geq 0$$

$$1.[\alpha_1-\alpha_2-\alpha_3+9\alpha_4+b]+z_4-1 \geq 0$$

$$z_i \geq 0,\ \alpha_i \geq 0 \quad i=1,...,4\ ;\ j=1,...,4$$

Note that, to be able to solve this problem by Matlab software, we need to change the sign from ≥ 0 to ≤ 0, thus (4.23) becomes:

$$\min[\alpha_1+\alpha_2+\alpha_3+\alpha_4+c(z_1+z_2+z_3+z_4)] \qquad (4.24)$$

s.t

$$-1.[9\alpha_1-\alpha_2-\alpha_3+\alpha_4+b]-z_1+1\le 0$$

$$1.[-\alpha_1+9\alpha_2+\alpha_3-\alpha_4-b]-z_2+1\le 0$$

$$1.[-\alpha_1+\alpha_2+9\alpha_3-\alpha_4-b]-z_3+1\le 0$$

$$-1.[\alpha_1-\alpha_2-\alpha_3+9\alpha_4+b]-z_4+1\le 0$$

$$z_i\ge 0,\ \alpha_i\ge 0 \quad i=1,...,4\ ;\ j=1,...,4.$$

4.5.2 Non-Linear Case (XOR Problem) with Uncertain Data

In section 4.5.1, the precise case was implemented by using equation (4.20). In this section the robust case will be considered and so instead of using (4.20), we are replacing it with (4.18) which represents the robust optimization model. So (4.18) can be rewritten as follows:

$$\min \sum_{i=1}^{l}\alpha_i + C\sum_{i=1}^{l} z_i$$

St (4.25)

$$-\sqrt{\eta}\,y_i\sqrt{\alpha^t\tilde{k}\alpha} + y_i\sum_{j=1}^{l} y_j\alpha_j k(\tilde{x}_j,\tilde{x}_i) - y_i\sqrt{\eta}\sum_{j=1}^{l} y_j\alpha_j\sqrt{k(\tilde{x}_j,\tilde{x}_j)} + y_i b + z_i - 1\ge 0$$

$$z_i\ge 0,\ \alpha_i\ge 0 \quad i=1,...,l\ .$$

Remember that $\|\tilde{\mathbf{x}}_i\| = \sqrt{k(\tilde{x}_j, \tilde{x}_j)}$ which is square root of the diagonal element of the matrix k. Note that $\tilde{k}_{ij} = y_i y_j k$ or equivalently:

$$\tilde{k}_{ij} = \begin{bmatrix} 9 & -1 & -1 & 1 \\ -1 & 9 & 1 & -1 \\ -1 & 1 & 9 & -1 \\ 1 & -1 & -1 & 9 \end{bmatrix}.$$

When we use the Matlab software to solve this problem, the inequality in the constraints must be less than or equal to 0 or ≤ 0. So (4.25) becomes:

$$\min \sum_{i=1}^{l} \alpha_i + C \sum_{i=1}^{l} z_i$$

St (4.26)

$$\sqrt{\eta} y_i \sqrt{\alpha^t \tilde{k} \alpha} - y_i \sum_{j=1}^{l} y_j \alpha_j k(\tilde{x}_j, \tilde{x}_i) + y_i \sqrt{\eta} \sum_{j=1}^{l} y_j \alpha_j \sqrt{k(\tilde{x}_j, \tilde{x}_j)} - y_i b - z_i + 1 \leq 0$$

$$z_i \geq 0, \ \alpha_i \geq 0 \quad i = 1, \ldots, l.$$

So if we solve the XOR problem by using the input and output data in table 2, we will then have four constraints. Next we construct the first constraint g (1).

$$g(1)=\sqrt{\eta}.(1)\left[(\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4)\begin{pmatrix} 9 & -1 & -1 & 1 \\ -1 & 9 & 1 & -1 \\ -1 & 1 & 9 & -1 \\ 1 & -1 & -1 & 9 \end{pmatrix}\begin{pmatrix}\alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4\end{pmatrix}\right]^{1/2} -$$

$$(1)\left[(1.\alpha_1.9)+(-1.\alpha_2.1)+(-1.\alpha_3.1)+(1.\alpha_4.1)\right] +$$

$$\sqrt{\eta}.(1)\left[(1.\alpha_1.\sqrt{9})+(-1.\alpha_2.\sqrt{9})+(-1.\alpha_3.\sqrt{9})+(1.\alpha_4.\sqrt{9})\right] -(1).(b).(z_1)+1\le 0$$

So the constraint g (1) above is equivalent to the following:

$$g(1)=\sqrt{\eta}\left[\begin{matrix}\alpha_1(9\alpha_1-\alpha_2-\alpha_3+\alpha_4)+\alpha_2(-\alpha_1+9\alpha_2+\alpha_3-\alpha_4)+\alpha_3(-\alpha_1+\alpha_2+9\alpha_3-\alpha_4)+ \\ \alpha_4(\alpha_1-\alpha_2-\alpha_3+9\alpha_4)\end{matrix}\right]^{1/2} -$$

$$\left(9\alpha_1-\alpha_2-\alpha_3+\alpha_4\right)+\sqrt{\eta}\left(3(\alpha_1-\alpha_2-\alpha_3+\alpha_4)\right)-b-z_1+1\le 0.$$

The same procedure as above will be conducted to construct the rest of the constraints; there are 3 more since the total data points are 4.The XOR model becomes as follows:

$$\min \sum_{i=1}^{l}\alpha_i + C\sum_{i=1}^{l} z_i \qquad (4.27)$$

$$g(1)=\sqrt{\eta}\left[\begin{array}{l}\alpha_1(9\alpha_1-\alpha_2-\alpha_3+\alpha_4)+\alpha_2(-\alpha_1+9\alpha_2+\alpha_3-\alpha_4)+\alpha_3(-\alpha_1+\alpha_2+9\alpha_3-\alpha_4)+\\ \alpha_4(\alpha_1-\alpha_2-\alpha_3+9\alpha_4)\end{array}\right]^{1/2}-$$

$$[9\alpha_1-\alpha_2-\alpha_3+\alpha_4|+\sqrt{\eta}[3(\alpha_1-\alpha_2-\alpha_3+\alpha_4|)|-b-z_1+1\le 0.$$

$$g(2)=\sqrt{\eta}\left[\begin{array}{l}\alpha_1(9\alpha_1-\alpha_2-\alpha_3+\alpha_4)+\alpha_2(-\alpha_1+9\alpha_2+\alpha_3-\alpha_4)+\alpha_3(-\alpha_1+\alpha_2+9\alpha_3-\alpha_4)+\\ \alpha_4(\alpha_1-\alpha_2-\alpha_3+9\alpha_4)\end{array}\right]^{1/2}+$$

$$[\alpha_1-9\alpha_2-\alpha_3+\alpha_4|-\sqrt{\eta}[3(\alpha_1-\alpha_2-\alpha_3+\alpha_4)|+b-z_2+1\le 0.$$

$$g(3)=\sqrt{\eta}\left[\begin{array}{l}\alpha_1(9\alpha_1-\alpha_2-\alpha_3+\alpha_4)+\alpha_2(-\alpha_1+9\alpha_2+\alpha_3-\alpha_4)+\alpha_3(-\alpha_1+\alpha_2+9\alpha_3-\alpha_4)+\\ \alpha_4(\alpha_1-\alpha_2-\alpha_3+9\alpha_4)\end{array}\right]^{1/2}+$$

$$[\alpha_1-\alpha_2-9\alpha_3+\alpha_4|-\sqrt{\eta}[3(\alpha_1-\alpha_2-\alpha_3+\alpha_4)|+b-z_3+1\le 0.$$

$$g(4)=\sqrt{\eta}\left[\begin{array}{l}\alpha_1(9\alpha_1-\alpha_2-\alpha_3+\alpha_4)+\alpha_2(-\alpha_1+9\alpha_2+\alpha_3-\alpha_4)+\alpha_3(-\alpha_1+\alpha_2+9\alpha_3-\alpha_4)+\\ \alpha_4(\alpha_1-\alpha_2-\alpha_3+9\alpha_4)\end{array}\right]^{1/2}-$$

$$[\alpha_1-\alpha_2-\alpha_3+9\alpha_4|+\sqrt{\eta}[3(\alpha_1-\alpha_2-\alpha_3+\alpha_4)|-b-z_4+1\le 0.$$

$$z_i\ge 0,\ \alpha_i\ge 0\quad i=1,\ldots,l$$

Chapter 5

Robust Optimization in Support Vector Machine "Regression Analysis"

5.1 Introduction

We consider the regression problem in which a response y is modeled using functions of input vectors x in R^d in the form $f(x) = \sum_j b_j \phi_j(x) + b$, where $\phi_1(x)$, $\phi_2(x)$... is a collection of possible nonlinear features. The traditional SVM solution takes b_j of the form $b_j = \sum_{i=1}^{l} \alpha_i \phi_j(x_i)$, where $x_1,\ldots, x_l$ are the observed input vectors in the data set of size l. Then the regression estimator takes the form: $f(x) = \sum_{i=1}^{l} \alpha_i k(x, x_i) + b$, where the kernel $k(x, x_i) = \sum_j \phi_j(x)\phi_j(x_i)$, and the $\alpha_1,\ldots\alpha_l$ solve a quadratic optimization problem designed to produce weights of smallest l_2 norm subject to producing fits of error ε to the observed responses y_1, y_2, ... y_l in the data sets provided such a solution exists.

Seeking sparsity on l_1 norm may be used in place of the l_2 norm. Unfortunately, accurate fits to the data may require that weights end up with a large norm, leading

also to a large statistical uncertainty concerning the values of α, b and f in the presence of noise.

Assuming the "noise" to be constrained to a given sphere S or more generally to an ellipsoid E, a robust optimization approach is proposed with the objective of developing a robust regressor with good generalization properties.

This chapter is organized as follows: In section 5.1, an introduction for the support vector regression is given. In section 5.2, the basic idea of support vector regression (SVR) is developed. In section 5.3, the linear epsilon- insensitive loss function is focused on. In section 5.4, the generalization of SVR is being developed. In section 5.5, the kernelization is used for the non-linear case.

5.2 Basic Idea

Let us consider a set of given training data $\{(x_1,y_1)...(x_l,y_l)\}$, where l represents the number of samples or the training set size, where $x_i \in \Re^d$ and $y_i \in \Re$.

SVM regression uses the ε-insensitive loss function, (Vapnik, 1995), as a tool to find a function f (x) that has at most ε deviation from the original or actual obtained targets y_i for all the training data, and at the same time, is as flat as possible. It is assumed that the function f is linear in the ϕ space. Specifically $f(x) = < w, x > +b$. Therefore our objective is to find a small w in terms of length.This can be accomplished by minimizing the norm of w, i.e. $||w||^2$,(Smola and Schhölkopf, 1998). Thus, if the deviation between the actual and predicted value is

smaller than ε, then the regression function is acceptable. Therefore errors less than ε will be neglected, but at the same time, no deviation larger than ε will be accepted either. The above idea can be expressed mathematically as follows. Find a w in $\mathbb{R}^d$ such that

$-\varepsilon \leq < w, x_i > -b-y_i \leq \varepsilon, \forall i = 1,\ldots, l$ or $|y_i - \langle w, x_i \rangle - b| \leq \varepsilon, \forall i = 1,\ldots,l.$

From a geometric point of view those inequalities can be visualized as a band or a tube of 2ε around the regression function and any point inside the tube is considered to be without error. On the other hand points outside the band are infeasible.

5.3 Linear Epsilon- Insensitive Loss Function

In Vapnik (1995), it can be seen that the support vector machine regression formulation, is described by the following primal optimization problem:

$$\begin{aligned} &\text{Min } 1/2 \; ||w||^2 \\ &\text{St} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (5.1) \\ &-< w,x_i > -b + y_i \leq \varepsilon \\ &< w,x_i > + b - y_i \leq \varepsilon \qquad \forall i = 1,\ldots,l, \end{aligned}$$

It should be pointed out, that (5.1) has been assumed to be feasible, but this is may not be always the case; sometimes it is wise to allow for some errors. Thus slack variables z and $\hat{z}$ are introduced for the two kinds of training errors. The first

calculates the error for underestimating the function, and the second calculates the error for overestimating the function. It should be noticed that these slack variables are different from zero for points outside the tube, and progressively decrease, until they reach the zero value for points inside the tube according to the loss function used. The above description is known as the ε-SV regression. (5.1) can be modified to cope with the infeasible constraints. So we need to minimize the norm of w subject to the above constraints with possible infeasibilities slack variables z *and* $\hat{z}$ respectively. Specifically the task is to solve the following optimization problem:

$$\text{Min } 1/2 \ ||w||^2 + C\sum_{i=1}^{l}(z_i + \hat{z}_i)$$

$$\text{St} \qquad (5.2)$$

$$y_i - \langle w, x_i \rangle - b - z_i \leq \varepsilon \qquad \forall i = 1,\ldots,l,$$

$$-y_i + \langle w, x_i \rangle + b - \hat{z}_i \leq \varepsilon$$

$$z_i, \hat{z}_i \geq 0 .$$

The positive constant C is the trade off between the flatness of the regression function f (x) corresponding to the above optimization problem (flatness means how small w should be), and the amount of deviation larger than ε that can be tolerated or allowed. The above formulation can be used for the linear ε-insensitive loss function. The corresponding regression function f (x) to problem (5.2) is as follows:

$$f(x) = \langle w, x \rangle + b, \qquad (5.3)$$

where x is in the φ space. As in the classification case, one can compute the Lagrangian dual and by using kernels functions, a nonlinear regressor case can then be constructed. The corresponding regression function to the nonlinear case is as follows:

$$f(x) = \sum \alpha_i k(x_i, x) + b. \tag{5.4}$$

We will see how these functions were derived, when we construct the general case for regression.

5.4 Generalization

Using the above problem (5.2), we will follow the same logic as in the case of classification, where the data points $\{x_i\}_{i=1,\ldots,l}$ had some noise (uncertainty). Specifically, let $x_i = \tilde{x}_i + u_i$, *such that* $\|u_i\| \le \sqrt{\eta}$, where u_i represents a bounded uncertainty and $\eta > 0$. By substituting this equality in the constraints of (5.2), we will get:

$$\begin{aligned} & y_i - \langle w, \tilde{x}_i + u_i \rangle - b - z_i \le \varepsilon \\ & -y_i + \langle w, \tilde{x}_i + u_i \rangle + b - \hat{z}_i \le \varepsilon \\ & z_i, \hat{z}_i \ge 0, \quad i = 1,\ldots,l \end{aligned} \tag{5.5}$$

$$\forall\, u_i \in \Re^d, \text{ such that } \|u_i\| \le \sqrt{\eta}.$$

With some simple mathematical manipulation, (5.5) can be transformed into:

$$\begin{aligned} & y_i - \langle w, \tilde{x}_i \rangle - \langle w, u_i \rangle - b - z_i \le \varepsilon \\ & \langle w, \tilde{x}_i \rangle + \langle w, u_i \rangle + b - y_i - \hat{z}_i \le \varepsilon \\ & z_i, \hat{z}_i \ge 0, \quad i = 1,\ldots,l, \end{aligned} \tag{5.6}$$

$\forall$ $u_i \in \Re^d$, such that $\|u_i\| \leq \sqrt{\eta}$.

In order to satisfy inequalities (5.6) in a "robust way", we need to minimize the dot product of w and u_i subject to $\|u_i\| \leq \sqrt{\eta}$. Thus, we need to solve the following problem:

$$\begin{aligned} &\min \langle w, u_i \rangle \\ &st \\ &\|u_i\| \leq \sqrt{\eta}. \end{aligned}$$

The minimum value of the above problem provides the worst possible scenario for satisfaction of the inequality in (5.6). Note that the minimum value of $\langle w, u_i \rangle$ is equal to $-\sqrt{\eta}\|w\|$. Again substituting this minimum in (5.6):

$$\begin{aligned} & y_i - \langle w, \tilde{x}_i \rangle + \sqrt{\eta}\|w\| - b - z_i \leq \varepsilon \\ & -y_i + \langle w, \tilde{x}_i \rangle - \sqrt{\eta}\|w\| + b - \hat{z}_i \leq \varepsilon \\ & z_i, \hat{z}_i \geq 0, \quad i = 1,...,l, \ \forall u_i \in \Re^d, \text{with} \|u_i\| \leq \sqrt{\eta}. \end{aligned} \tag{5.7}$$

So the new problem becomes:

$$\text{Min } 1/2 \ \|w\|^2 + C\sum_{i=1}^{l}(z_i + \hat{z}_i)$$

St (5.8)

$$\begin{aligned} & \sqrt{\eta}\|w\| - \langle w, \tilde{x}_i \rangle - b + y_i - z_i \leq \varepsilon \\ & -\sqrt{\eta}\|w\| + \langle w, \tilde{x}_i \rangle + b - y_i - \hat{z}_i \leq \varepsilon \\ & z_i, \hat{z}_i \geq 0, \quad i = 1,...,l, \ \forall u_i \in \Re^d, \text{with} \|u_i\| \leq \sqrt{\eta}. \end{aligned}$$

5.5 Kernelization

In the previous section, the linear regression problem was provided. In this section, the above problem can be reformulated to deal with non-linear regression problems. As in the classification case, we can express the weight vector as a linear combination of the data vectors; specifically $\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i$. Using a similar analysis as in the classification case in chapter 4, the length of the weight vector can also be expressed as: $\|w\| = \sqrt{\alpha' k \alpha}$. Now since $\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i$, $\langle \mathbf{w}, \tilde{\mathbf{x}}_i \rangle$ can also be expressed as:

$\langle \sum_{j=1}^{l} \alpha_j \mathbf{x}_j, \tilde{\mathbf{x}}_i \rangle = \sum_{j=1}^{l} \alpha_j \langle \mathbf{x}_j, \tilde{\mathbf{x}}_i \rangle = \sum_{j=1}^{l} \alpha_j k\langle \mathbf{x}_j, \tilde{\mathbf{x}}_i \rangle$, where α_j is a real number unrestricted in sign. Now by setting $\mathbf{x}_j = \tilde{x}_j + u_j$, the dot product of $\mathbf{x}_j$ and $\tilde{x}_i$ can then be expressed as:

$$\langle \mathbf{x}_j, \tilde{\mathbf{x}}_i \rangle = \langle \tilde{\mathbf{x}}_j + \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle = \langle \tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_i \rangle + \langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle .$$

Again from 2.5.1 (Definition 2), $\langle \tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_i \rangle$ can be replaced by its kernelization $k(\tilde{x}_j, \tilde{x}_i)$, and $\langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle$ by its minimum value $-\sqrt{\eta}\|\tilde{\mathbf{x}}_i\|$. Remember that $\|\mathbf{u}_j\| \le \sqrt{\eta}$. Thus: $\langle \mathbf{w}, \tilde{\mathbf{x}}_i \rangle = \sum_{j=1}^{l} \alpha_j k\langle x_j, \tilde{x}_i \rangle$ and $\langle \mathbf{x}_j, \tilde{\mathbf{x}}_i \rangle = \langle \tilde{\mathbf{x}}_j + \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle = \langle \tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_i \rangle + \langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle$ .By setting $\langle \mathbf{u}_j, \tilde{\mathbf{x}}_i \rangle = -\sqrt{\eta}\|\tilde{\mathbf{x}}_i\|$ we have:

$\langle \mathbf{w}, \tilde{\mathbf{x}}_i \rangle = \sum_{j=1}^{l} \alpha_j [k(\tilde{x}_j, \tilde{x}_i) - \sqrt{\eta}\,\|\tilde{\mathbf{x}}_i\|] = \sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) - (\sum_{j=1}^{l} \alpha_j \sqrt{\eta}\,\|\tilde{\mathbf{x}}_i\|)$. Thus, problem (5.8) can be stated as:

$$\frac{1}{2}\alpha' k\alpha + C\sum_{i=1}^{l}(z_i + \hat{z}_i)$$

s.t (5.9)

$$\sqrt{\eta}\sqrt{\alpha' k\alpha} - \sum_{j=1}^{l}\alpha_j k(\tilde{x}_j, \tilde{x}_i) + (\sum_{j=1}^{l}\alpha_j\sqrt{\eta})\|\tilde{\mathbf{x}}_i\| - b + y_i - z_i \le \varepsilon$$

$$-\sqrt{\eta}\sqrt{\alpha' k\alpha} + \sum_{j=1}^{l}\alpha_j k(\tilde{x}_j, \tilde{x}_i) - (\sum_{j=1}^{l}\alpha_j\sqrt{\eta})\|\tilde{\mathbf{x}}_i\| + b - y_i - \hat{z}_i \le \varepsilon$$

$z_i, \hat{z}_i \ge 0,\ i = 1,...,l\ ;\ j = 1,...,l;\ \alpha$ *is unrestricted.*

We will call this model the robust Support Vector Machine with non-linear kernel. Again making some necessary substitutions in the equation of the regression, we can derive the final regression function and expressed as:

$$f(x, \alpha, b) = \sum_{i=1}^{l}\alpha_i\langle\tilde{\mathbf{x}}_i, \mathbf{x}\rangle + b$$

$$f(x, \alpha, b, u) = \sum_{i=1}^{l}\alpha_i\langle\tilde{\mathbf{x}}_i + \mathbf{u}_i, \mathbf{x}\rangle + b$$

$$f(x, \alpha, b, u) = \sum_{i=1}^{l}\alpha_i\langle\tilde{\mathbf{x}}_i, \mathbf{x}\rangle + \sum_{i=1}^{l}\alpha_i\langle\mathbf{u}_i, \mathbf{x}\rangle + b$$

Notice that $\langle\tilde{\mathbf{x}}_i, \mathbf{x}\rangle = k(\tilde{x}_i, x)$, and by Cauchy-Scharwz inequality we have $|\langle\mathbf{u}_i, \mathbf{x}\rangle| \le \|\mathbf{u}_i\|\|\mathbf{x}\|$. Considering the worst case, we replace $\langle\mathbf{u}_i, \mathbf{x}\rangle$ by its minimum value $-\sqrt{\eta}\|\mathbf{x}\|$ and denoting by $\hat{f}(x, \alpha, b)$ (robust estimator) we have:

$$\hat{f}(x,\alpha,b) = \sum_{i=1}^{l} \alpha_i k(\tilde{x}_i, x) - \sqrt{\eta}\,\|\mathbf{x}\|(\sum_{i=1}^{l} \alpha_i) + b \text{, where } \|\mathbf{x}\| = \sqrt{k(x,x)}\,. \qquad (5.10)$$

In the case where we do not have any uncertainties (η equals to zero), using non-linear kernels such as the polynomial kernel, (5.9) will be equivalent to the following:

$$\frac{1}{2}\alpha' k\alpha + C\sum_{i=1}^{l}(z_i + \hat{z}_i)$$

s.t (5.11)

$$-\sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) - b + y_i - z_i \le \varepsilon$$

$$\sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) + b - y_i - \hat{z}_i \le \varepsilon$$

$z_i, \hat{z}_i \ge 0,\ i = 1,...,l\ ;\ j = 1,...,l;\ \alpha$ *is unrestricted.*

We shall call this case the precise non-linear SVM. Note that (5.10) becomes:

$$f(x,\alpha,b) = \sum_{i=1}^{l} \alpha_i k(\tilde{x}_i, x) + b\,. \qquad (5.12)$$

In the case where we use the L_1 norm, (5.11) would lead to a linear programming problem such as:

$$\min \sum_{i=1}^{l} \alpha_i + C\sum_{i=1}^{l}(z_i + \hat{z}_i)$$

s.t (5.13)

$$-\sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) - b + y_i - z_i \le \varepsilon$$

$$\sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) + b - y_i - \hat{z}_i \le \varepsilon$$

$z_i, \hat{z}_i \ge 0,\ i = 1,...,l\ ;\ j = 1,...,l;\ \alpha$ *is unrestricted.*

We call this case precise linear SVM optimization problem, and the regression function for this case stays the same as in (5.12). The robust linear SVM, will be as follows:

$$\min \sum_{i=1}^{l} \alpha_i + C \sum_{i=1}^{l} (z_i + \hat{z}_i)$$

s.t (5.14)

$$\sqrt{\eta}\sqrt{\alpha' k \alpha} - \sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) + (\sum_{j=1}^{l} \alpha_j \sqrt{\eta}) \|\tilde{\mathbf{x}}_i\| - b + y_i - z_i \le \varepsilon$$

$$-\sqrt{\eta}\sqrt{\alpha' k \alpha} + \sum_{j=1}^{l} \alpha_j k(\tilde{x}_j, \tilde{x}_i) - (\sum_{j=1}^{l} \alpha_j \sqrt{\eta}) \|\tilde{\mathbf{x}}_i\| + b - y_i - \hat{z}_i \le \varepsilon$$

$z_i, \hat{z}_i \ge 0,\ i = 1,...,l\ ;\ j = 1,...,l;\ \alpha$ *is unrestricted*,

where $\|\tilde{\mathbf{x}}_i\|$ equals to $\sqrt{k(x_i, x_i)}$. In each of the above models, we can determine the mean square error, as a tool to the prediction behavior of our models, and graphs can be used to visualize this behavior, that is how the solution of our model changes with the changes of the uncertainty eita (η).

Chapter 6

Computational Results

6.1 Results for Pattern Recognition or Classification

6.1.1 Tests on Synthetic Problems

The AND function

Description

All experiments for both classification and regression have been conducted in MATLAB. We used a 2.4 GHZ Intel processor, Dell workstation-computer with 2 GB RAM and running Microsoft windows 2000 professional version with service pack 2 installed.

Preliminary experiments were conducted on the AND function (linear case), which contains 4 data points with label assignments a label of +1 and –1 respectively as can be seen in table 1, Appendix C. Two experiments were performed; one for the sensitivity analysis case and the other one for the robust case. A comparison and illustration of the results are shown in Tables 3 and 4, Appendix C and figures1 and 2, Appendix E. In the sensitivity case, uncertainty is considered only in two directions along the axes of x_1 and x_2 respectively where in the robust case, the uncertainty is considered in all directions since the idea of robustness is to consider a sphere in which the uncertainty is bounded and represented by all possible rays of that sphere.

The linear kernel was used. We found that when we set the value of the uncertainty (eita) equal to 0.5, no separation of the two groups of data points could be performed in the sensitivity analysis case, while in the robust case, we found two SV points rather than having no separation at all. The value of the uncertainty at which no separation could be done for the robust case is 0.7. Thus, we can see that the robust case is a better performer than the sensitivity analysis one. Also, we noticed that in the case of sensitivity analysis, the separating hyperplanes had different slopes; while in the robust case they had the same slope (parallel translation) with respect to the optimal plane of the precise case. We believe that this phenomenon is due to the fact that we are considering a spherical uncertainty in the robust case with the same bound or radius for all data points.

The Exclusive-OR Problem (XOR) - Non-linear case

Description

In the AND function case, we have defined hyperplanes as linear functions in the input space. In this section, the XOR example demonstrates the mechanics of the support vector machine calculations for the nonlinear separable case. This problem is not linearly separable in the input space. We need to find an optimal separating hyperplane that classifies the data in table 2 without error. As we have mentioned above it is not possible to solve this problem with a linear decision boundary. However, a polynomial decision boundary of order 2 can separate these data. The inner product kernel for polynomials of order 2 is defined as $k(x,y) = (1+ \langle x, y \rangle)^2$.

The solution to the resulting SVM optimization problem (4.23) is $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$ indicating that all four data points are support vectors. The decision function using the above polynomial kernel will be:

$$D(x) = \sum_{i=1}^{l} \alpha_i y_i k(x_i, x) = (0.125) \sum_{i=1}^{4} y_i [(x_i, x) + 1]^2.$$

This decision function separates the data with a maximum margin. We have solved the XOR problem with different values for the uncertainty, using the L_1 norm. Of course when uncertainty equals to zero that is, no uncertainty is involved, the solution of the XOR problem is the same as in the case of the precise SVM formulation.

According to our robust constructions, some specific real world problems can be solved, taking into account the factor of uncertainty. Thus, if one assumes that the uncertainty values will vary in a specific range, then it is easy to find the solution for that specific problem, just by plugging the value of the uncertainty in our robust formulation. We have observed that as the value of the uncertainty decreases, the solution will get closer to the one with precise data.

6.1.2 Tests on Real World Problems

Echocardiogram Classification Application

Description and Analysis

Each exemplar contains information collected from patients who have had a recent heart attack. The problem is to determine if the patient will survive for 1 year following their heart attack. The most difficult part is predicting that the patient will NOT survive. There are relatively few exemplars.

Number of Exemplars: 60

Attribute Elements per Exemplar: 7 continuous, 1 symbolic (discrete). Specifically the attributes (components) of input vectors are as follows:

1. Age-at-heart-attack: age in years when heart attack occurred.

2. Pericardial-effusion (binary): Pericardial effusion is the fluid around the heart. (0=no fluid, 1= fluid).

3. Fractional-shortening: a measure of contractility around the heart (lower numbers are increasingly abnormal).

4. Epss: E-point septal separation is another measure of contractility. Larger numbers are increasingly abnormal.

5. Lvdd: left ventricular end-diastolic dimension. This is a measure of the size of the heart at end-diastole. Large hearts tend to be sick hearts.

6. Wall-motion-score: a measure of how the segments of the left ventricle are moving.

7. Wall-motion-index: it equals wall-motion-score divided by number of segments seen. Usually 12-13 segments are seen in an echocardiogram. Use this variable instead of the wall motion score.

8. mult : a derivate variable which can be ignored

Number of Outputs: 1

Output Encoding: 1 for survived

In the experimental setting of this problem (echocardiogram), the total data points were 60. We used 40 points as training points and 20 as testing. We observed that there were no big changes in the values of the mean square testing error (mse), which occurred with the changes of the uncertainties values. This advocates for the robustness (stability) of the proposed model. We were satisfied with the results we obtained, but from the statistics point of view, we thought that it would be better if we apply the same experiment on different data points. We then generated two other data points from the original one randomly and named them echocardio1 and echocardio2 respectively, and repeated the same experiment on these two new data. The results were similar to the original one. Tables 6,7 and 8, Appendix F show the results of these three data sets. I should put a note that during these experiments, sometimes I had to change the value of the trade off C to get better classification by getting better values of the mse.

Breast cancer Classification Application

Description and Analysis

In this problem we try to classify breast cancer data using 9 attributes. The 2 classes are benign and malignant repectively:

Number of Exemplars: 100

Data attributes per Exemplar: 9 continuous

1. Clump Thickness 1 - 10
2. Uniformity of Cell Size 1 - 10
3. Uniformity of Cell Shape 1 - 10
4. Marginal Adhesion 1 - 10
5. Single Epithelial Cell Size 1 - 10
6. Bare Nuclei 1 - 10
7. Bland Chromatin 1 - 10
8. Normal Nucleoli 1 - 10
9. Mitoses 1 – 10

Number of Outputs: 1

Output Encoding:

Output1 = benign?

In the experiment of this problem (breast cancer), the total data points were 100. We used 60 points for training and 40 for testing. We observe stability of the method, that is, no big changes in the values of the mean square testing error (mse)

occurred with the changes of the uncertainties values. The results were similar to the original one. I should put a note here, that during these experiments sometimes i had to change the value of the trade off C to get better classification by getting better values of the mse. Results can be seen in table 9, Appendix F.

6. 2 Results for Regression Analysis

6.2.1 Tests on Synthetic Problem

Description and Analysis

The data of this problem was generated according to the following equation: f(x) = $x^2_1 + x^2_2$. This data consists of 100 points. In our experimental setting, we randomly select 11 data sets where 30 points used for training and 70 for testing. We then call our C++ code, into the Matlab and run those eleven data sets with different values of uncertainties (η). The purpose of our experiments was to find the mean square error for the training and testing points for each data set. Then we calculated the average mean square error for the eleven groups of data points with the same value of the uncertainty (η). Then we start making the changes for the values of the uncertainties. The result was very encouraging. We run 77 experiments for those eleven data (7 experiments for each data) since 7 values of (η) were used in these experiments. I should point out that, there has been a consistency with the changes of the values of the uncertainty (η) and the values of the mean square error for both testing and training points, that is when the uncertainty value increases, the value of the mse also

increases but with the same proportion. This shows that the result obtained by this model is stable as we expected. Also, we can visualize these analytical results by the graphs we obtained for the different values of (η). The result can be seen in tables 10 and 11, Appendix D and the figures can also be seen in the list of regressions' graphs, figures 7 and 8, Appendix F.

6.2.2 Tests on Real World Problems

Lynx Application Problem

Description and Analysis

LYNX TRAPPING PREDICTION:

This data set contains the number of lynx trapped in the MacKenzie River District of Northwest Canada between 1821 and 1934. It has been used to illustrate a number of methods associated with time series analysis.

Number of Exemplars: 114

Data Elements per Exemplar: 1

Number of Outputs: prediction

Output Encoding: prediction

Problem Type: Time Series Prediction

In this application, the total points used were 40 points; 30 points for training and 10 points for testing. We tried many experiments but only two of them gave reasonable results. The first is when the uncertainty (η) equals to 0 and the other one is when eita equals to 0.0001. When (η) equals to 0, the testing mse were 0.3726 and when

(η) increases, the value of the testing mse were also increased. But these increases are proportional to the increases of the uncertainty values, which means that, the stability of the model's solution exists.

Chapter 7

Summary, Recommendations for Future Works

7.1 Summary

In this work, we have developed new methods for solving problems in pattern classification and regression analysis with uncertainty in the data in feature space. Specifically we consider spherical uncertainties (data are restricted in spherical regions of radius $\sqrt{\eta}$ in feature space). The changes in the behavior of the solutions of these two types of problems were examined under different values of the uncertainties described through a parameter called η. The resulting robust discriminant functions that showed stability in several data sets was very encouraging. These results were consistent with the goal of this dissertation. Problems from real world such as echocardiogram, breast cancer and the lynx, were examined for illustrations. The contribution of this dissertation was to investigate the robustness and stability of the SVMs model under bounded perturbations of the input data in the feature space.

7.2 Recommendations for Future Work

Preliminary experimentations has shown that robust techniques possess a good stability behaviour in terms of generalization and a promising potential for predicting solutions under unknown perturbations of the data in feature space. Future research might focus on studies of the statistical properties of the robust SVM solution and performance in stochastic settings. Analyzing the numerical properties of the proposed models in large-scale settings is a topic for further research. Extensions to this work to deal with models, with perturbations in a subset of the data points and with different ellipsoidal uncertainties could be investigated. Also we recommend extending this research to models where the uncertainties can be expressed as convex sets and investigate connections with positive semidefinite programming. Extensions to other methods such as the least square SVM can be examined and developed. It is also a good idea to work on identifying the uncertainties of the data using statistical methods. Since we only consider the perturbation in the input data, considering perturbations both in input and output data is a good topic for further research. Finally, one can investigate the problem of the selection of the value of the trade off C in an automated procedure based on a given set of training data.

Appendix A

C++ and MATLAB Code for Pattern Recognition / Classification

```
-- C++ code
#include <fstream.h>
#include <string.h>
/*****************************************************************************
        This code generates constrain equations for the classification with dimension
        n.
        w(1) ... w(n) : represents ai
        w(n+1)...w(2n) : represents zi
        w(2n+1) : represents b
*****************************************************************************/

int main()
{
        // Declaration of some variables used in the program, such as the file name
        // The input and output streams
        ifstream fin;
        ofstream fout;
        char iFileName[50], temp[53];

        // Reading the data file name in order to read the data
        cout << "Enter the data file name: ";
        cin >> iFileName;
        fin.open(iFileName);

        // We use the file name 'class1.m' to store the constrains
        strcpy(temp,"class1");
        strcat(temp, ".m");
        fout.open(temp);

        // Read how many points we want to use for training
        cout << "Enter the number of points: ";
        int NumOfPoints;
        cin >> NumOfPoints;

        // Define the array dimensions that are going to hold the input data
        float *y;
        int Dimension = 2;
        float x[31][1001];      // 30 dimensions, 1000 data points
        y = new float [NumOfPoints + 1];
        cout << "Enter the dimension: ";
        cin >> Dimension;

        /*************************************************************
                Reading Stage: Reading the data from the file
```

```
**********************************************************************/
for (int i = 1; i <= NumOfPoints; i++)
{
        for (int j=1; j <= Dimension; j++)
                fin >> x[j][i];
        fin >> y[i];
}
fin.close();

// Continue reading the parameters, C and Eita.
cout << "Enter the value of C: ";
float C;
cin >> C;

cout << "Enter eta: ";
float eta;
cin >> eta;

/**********************************************************************
 *      Start writing the output file that contains the constrains 'class1.m'    *
**********************************************************************/
fout << "%l = " << NumOfPoints << endl
        << "%dimension = " << Dimension << endl
        << "%C = " << C << endl
        << "%eita = " << eta << endl;

// Start writing the declaration of the function
fout << "function [f,g]=class1(w)" << endl
        << "k=[";

// Finding polynomial kernel.
// In this program, the polynomial version of the kernel has been used.
ktemp = 0;
for (i = 1; i <= NumOfPoints; i++)
{

        for (int j = 1; j <= NumOfPoints; j++)
        {
                // find the k(i,j) for the dimensional data
                ktemp = 0;
                for (int k=1; k <= Dimension; k++)
                        ktemp = ktemp + x[k][i]*x[k][j];
                ktemp ++;
                ktemp *= ktemp;
```

```
                fout << ktemp << " ";

            }
            fout << ";" << endl;
        }
        fout << "];" << endl;

//find polynomial ktildit
fout << "ktilda=[";

        ktemp = 0;
        for (i = 1; i <= NumOfPoints; i++)
        {
                for (int j = 1; j <= NumOfPoints; j++)
                {
                        // find the k(i,j) for the dimensional data
                        ktemp = 0;
                        for (int k=1; k <= Dimension; k++)
                                ktemp = ktemp + x[k][i]*x[k][j];
                        ktemp ++;
                        ktemp *= ktemp;
                        fout << y[i]*y[j]*ktemp << " ";
                }
                fout << ";" << endl;
        }

        fout << "];" << endl;

        /*************************************************************
         *                  Start writing the objective function                    *
         *************************************************************/
        fout << "f= (";
        for (j = 1; j <= NumOfPoints; j++)
        {
                fout <<"w(" << j << ")";
                if (j < NumOfPoints)
                        fout << "+";
        }

        fout << ")+" << C << "*(";

        for (i=1; i <= NumOfPoints; i++)
        {
                fout << "w(" << NumOfPoints+i << ")";
```

```
        if (i < NumOfPoints)
                fout << "+";
}
fout << ");" << endl;

float ytemp;

/*****************************************************************
 *                        Start writing the constraints
 *
 *****************************************************************/
ytemp = 1;
for (i=1; i <= NumOfPoints; i++)
{
        fout << "g(" << i << ")= sqrt(" << eta << ")*" << y[i] <<"*sqrt(";
        for (int k = 1; k <= NumOfPoints; k++)
        {
                fout << "w(" << k << ")*(";
                for (int j = 1; j <= NumOfPoints; j++)
                {
                        fout <<"w(" << j << ")*ktilda(" << j << "," << k << ")";
                        if (j < NumOfPoints)
                                fout << "+";
                }
                fout << ")";
                if (k < NumOfPoints)
                        fout << "+";
        }
        fout << ")";
        if (y[i] > 0)
                fout << "-" << y[i];
        else
                fout << "+" << -y[i];

        fout << "*(0";

        for (int j = 1; j<= NumOfPoints; j++)
        {
                if (y[j] > 0)
                        fout << "+";
                fout << y[j] << "*w(" <<j<<")*k(" << j << "," << i << ")";
        }
        fout << ")";
        if (y[i] > 0)
```

```
                fout << "+" << y[i];
            else
                fout << y[i];
            fout << "*sqrt(" << eta << ")*(0";
            for (j = 1; j<= NumOfPoints; j++)
            {
                if (y[j] > 0)
                    fout << "+";
                fout << y[j] << "*w(" <<j<<")*sqrt(k(" << j << "," << j <<
                    "))";
            }
            fout << ")";

            if (y[i] > 0)
                fout << "-" << y[i];
            else
                fout << "+" << -y[i];

            fout << "*w(" << 2*NumOfPoints+1 << ")-w(" << NumOfPoints + i
                << ") + 1;" << endl;
        }
        for (i = NumOfPoints+1; i <= 3*NumOfPoints; i++)
        {
            fout << "g(" << i << ")= -w(" << i-NumOfPoints << ");" << endl;
        }
        /*****************************************************************
         *                          End of File                          *
         *****************************************************************/
        fout.close();
        return 0;
}
```

Matlab Code

This code solves the AND function in the robust case.

```
%This code solves the classification problem for the AND function

%Preparing the intial values for the analysis
eita = 0;                % uncertainity
w0 = zeros(1,4);         % intial values
vlb = -inf;              % no lower bound
vub = inf;               % no upper bound
options = 1;
w = constr('andclass',w0,options,vlb,vub);    %start the analysis
w'

%The code below finds the yhat (predicted function) resulted from the analysis and
plots the classification graph

load anddata.txt
x1 = anddata(:,1);
x2 = anddata(:,2);
y = anddata(:,3);
pos = 1;          %number of +ve yhat
neg = 1;          %number of -ve yhat
% Here we try to separate the +ve and the -ve yhat
for i=1:4
   if (y(i) > 0)
      px1(pos) = x1(i);
      px2(pos) = x2(i);
      pos = pos + 1;
   else
      nx1(neg) = x1(i);
      nx2(neg) = x2(i);
      neg = neg + 1;
   end
end
%here we draw the figure
figure(1);
hold on
%We use the '+' to represent positive yhat, and the 'o' for negative yhat.
plot (px1,px2,'+')
plot (nx1,nx2,'o');
```

This file graphs the AND function in the robust case

```
% This script file draws the classification graph for the AND function
[xcont ycont] = meshgrid(-5:1:5, -5:1:5);
zcont = w(1)*xcont+w(2)*ycont+w(4);
contour(xcont,ycont,zcont,[0 0],'b');

hold on
```

```
%This code solves the classification problem for the AND function using the

%Sensitivity analysis case


%Preparing the intial values for the analysis
eita = 0;                % uncertainity
w0 = zeros(1,4);         % intial values
vlb = -inf;              % no lower bound
vub = inf;               % no upper bound
options = 1;
w = constr('andclass',w0,options,vlb,vub);    %start the analysis
w'

%The code below finds the yhat (predicted function) resulted from the analysis and
%plots the classification graph

load anddata.txt
x1 = anddata(:,1);
x2 = anddata(:,2);
y = anddata(:,3);
pos = 1;          %number of +ve yhat
neg = 1;          %number of -ve yhat
% Here we try to separate the +ve and the -ve yhat
for i=1:4
   if (y(i) > 0)
      px1(pos) = x1(i);
      px2(pos) = x2(i);
      pos = pos + 1;
   else
      nx1(neg) = x1(i);
      nx2(neg) = x2(i);
      neg = neg + 1;
   end
end
%here we draw the figure
figure(2);
hold on
%We use the '+' to represent positive yhat, and the 'o' for negative yhat.
plot (px1,px2,'+')
plot (nx1,nx2,'o');
```

```
%This code graphs the AND function for the sensitivity analysis case

% This script file draws the classification graph for the AND operation using the
Sensitivity analysis

[xcont ycont] = meshgrid(-5:1:5, -5:1:5);
zcont = w(1)*xcont+w(2)*ycont+w(4);
contour(xcont,ycont,zcont,[0 0],'b');
```

```
%This code solves XOR classification problem and the rest of the classification

%This is a matlab script file that runs the constr command and finds the yhat for
%classification.

n = 10;    %Number of training Points in the data
dimension = 2;  %Number of dimentions
eita =0.0;
datapoints =92;   %Number of total data points in a specific data file
gridoption = 1;    % 1: turn it on, 0: turn it off

% Here is the code for the classification training
w0=zeros(1,2*n+1);
vlb=-inf;
vub=inf;
options(14)=100000;
% the file name 'class1' contains the function to be optimized
w=constr('class1',w0,options,vlb,vub);
w'                % To show the result of w

%extract alpha from w (the order of w is: alpha(n points), z(n points) and b)
alpha = w(1:n);
%load the data file for the breastcancer (or any data file in general)
load breastcancer.txt
x = breastcancer (1:n,1:dimension);  %extract training x
y = breastcancer (1:n,dimension+1);  %extract training y
xdata = breastcancer (1:datapoints,1:dimension);    %extract testing x
yy = breastcancer (1:datapoints, dimension+1);   %extract testing y
b = w(2*n+1);

[q,r] = size(xdata);
xx = x;

yhat = ftest(n,dimension,x,xx,y,alpha,eita,b);    %yhat function
yhat=sign(yhat)    %convert yhat to +1/-1


%here we calculate the Mean Square Error
trainingmse = y-yhat;
trainingmse = trainingmse.^2;
trainingmse = sum(trainingmse);
trainingmse = trainingmse/n;
trainingmse
```

```
%This percentage for the wrong yhat (misclassified)
ErrorTrainingPercentage = 0;
for i=1:n
  if (y(i) ~= yhat(i))    % if y and yhat are different, increase the misclassified points
by 1
     ErrorTrainingPercentage = ErrorTrainingPercentage + 1;
  end
end

ErrorTrainingPercentage = ErrorTrainingPercentage / n *100;
ErrorTrainingPercentage
```

```
%This code graphs the XOR problem only

% This script file draws the classification graph for the XOR operation

% Create meshgrid data matrx for the contour plot.
[xcont ycont] = meshgrid(-5:1:5, -5:1:5);
zcont = y(1)*w(1)*(1+xcont + ycont).^2 + y(2)*w(2)*(1+xcont-ycont).^2 +
y(3)*w(3)*(1-xcont+ycont).^2 + y(4)*w(4)*(1-xcont-ycont).^2 -
sqrt(eita)*3*(y(1)*w(1) +y(2)*w(2) + y(3)*w(3) + y(4)*w(4));
contour(xcont,ycont,zcont,[-.5,.5]);
% This code draws the input point.
% Points with y positive is drawn in +
% Points with y negative is drawn in o
load data.txt
x1 = data(:,1);
x2 = data(:,2);
y = data(:,3);
pos = 1;
neg = 1;
for i=1:n
   if (y(i) > 0)
      px1(pos) = x1(i);
      px2(pos) = x2(i);
      pos = pos + 1;
   else
      nx1(neg) = x1(i);
      nx2(neg) = x2(i);
      neg = neg + 1;
   end
end

hold on
% plot x1, x2 with +ve y
plot (px1,px2,'+');
% plot x1, x2 with -ve y on the same previous graph
plot (nx1,nx2,'o');
```

```
%This code graphs general classification problems

% This is a matlab script file that is used to plot the trainig and the testing results for
the classification
% Two graphs are used:
% 1) Original y vs yhat, where y values represented by '+' and yhat values represented
by 'o'
% 1) Original y vs training yhat
% 2) Original y vs testing yhat

training = n;   % number of training points
testing = datapoints - n;        %number of testing points
z = zeros(a1,a2);
temp = [0:datapoints-1];
yhatdata = ftest(n,dimension,x,xdata,yy,alpha,eita,b);          %find yhat
yhatdata=sign(yhatdata)                  % is used to detect the sign of yhat
%This percentage for the wrong yhat (misclassified)
testingpercentage = 0;
for i=n+1:datapoints
   if (yy(i) ~= yhatdata(i))    % if y and yhat are different, increase the misclassified
points by 1
      testingpercentage = testingpercentage + 1;
   end
end

% Calculate the testing MSE
testingpercentage = testingpercentage / (datapoints-n) * 100;
testingpercentage

% first graph, plot original y, where y values represented by '+' and yhat values
represented by 'o'
plot(temp(1:training),yy(1:training),'o');
hold on;
plot(temp(1:training),yhat(1:training),'+');
if (gridoption == 1)
   grid;
end
title('comparison between y (o) & yhat (+)');
xlabel('index');
ylabel('y, yhat');
hold off;

% Second graph, plot original y (o) vs training yhat (+)
figure(2);
```

```
clf
plot(temp(training+1:datapoints),yy(training+1:datapoints),'o');
hold on;
plot(temp(training+1:datapoints),yhatdata(training+1:datapoints),'+');
if (gridoption == 1)
   grid;
end
title('comparison between y (o) & yhat (+)');
xlabel('index');
ylabel('y, yhat');
hold off;


% Third graph, plot original y (o) vs testing yhat (+)
figure(3);
clf
plot(x(:,1),x(:,2),'o');
if (gridoption == 1)
   grid;
end
title('Data plot');
xlabel('x1');
ylabel('x2');
hold off;


% Calculate the MSE for the testing sample
testingmse = yy(training+1:datapoints)- yhatdata(training+1:datapoints);
testingmse = testingmse.^2;
testingmse = sum(testingmse);
testingmse = testingmse/testing;
testingmse

%This percentage for the wrong yhat (misclassified)
testingpercentage = 0;
for i=n+1:datapoints
   if (yy(i) ~= yhatdata(i))    % if y and yhat are different, increase the misclassified
points by 1
      testingpercentage = testingpercentage + 1;
   end
end

testingpercentage = testingpercentage / (datapoints-n);
testingpercentage
```

```
%This code solves for the decision function

% This function represents the decision function for classification.
% Call this function in order to find the yhat after running the constr command
% This file is called by the fcode.m to find the training yhat, and by the
% fgraph.m in order to find the testing yhat

function y=ftest(n,d,x,xx,y,alpha,eita,b)
temp = 0;
temp2 = 0;
%x is the training data
%xx is the testing data
%temp is used to find the term yi*alpha*k(x,xi)
for i=1:n
   temp = temp +  y(i)*alpha(i)*((xx*x(i,1:d)'+1).^2);
end
[q,r] = size(xx);

%k is used to find ||x||
for i=1:q
   k(q) = xx(i,1:d)*xx(i,1:d)' + 1;
end

%temp is used to find the term yi*sqrt(eita)*sum(K(x,x)*alpha)
for i=1:n
   temp2 = temp2 - k'.*y(i)*sqrt(eita)*alpha(i);
end
temp = temp + temp2 + b;

y = temp;
```

Appendix B

C++ and MATLAB Code for Regression Analysis

This code solves the alpha_K_alpha in the objective function.

```
#include <fstream.h>
#include <string.h>
/*************************************************************************
This code generates constrains equations for the Robust SVM
Regression with n dimensional input.
The maximum number of input dimension is 30, with maximum of
1000 data points.

        w(1) ... w(n) : represents ai
        w(n+1)...w(2n) : represents zi
        w(2n+1)...w(3n) : represents z^i
        w(3n+1) : represents b
*************************************************************************/

int main()
{
        // Define some variables to be used in the program, such as the file name.
        ifstream fin;
        ofstream fout;
        char iFileName[50], temp[53];

        // Reading the data file name in order to read the data
        cout << "Enter the data file name: ";
        cin >> iFileName;
        fin.open(iFileName);

        // We use the file name 'rnc.m' to store the constraints in.
        strcpy(temp, "rnc");
        strcat(temp, ".m");
        fout.open(temp);

        // Read how many points we want to use for training the SVM
        cout << "Enter the number of points: ";
        int NumOfPoints;
        cin >> NumOfPoints;

        /*************************************************************
                        Start reading the data from the data file
        *************************************************************/
        float *y;
        int Dimension = 2;
        float x[31][1001];      // 30 dimensions, 1000 data points
```

```
        y = new float [NumOfPoints + 1];
        cout << "Enter the dimension: ";
        cin >> Dimension;

        for (int i = 1; i <= NumOfPoints; i++)
        {
                for (int j=1; j <= Dimension; j++)
                        fin >> x[j][i];
                fin >> y[i];
        }
        fin.close();

        // Read the C, epsilon and eita.
        cout << "Enter the value of C: ";
        int C;
        cin >> C;

        cout << "Enter epsilon: ";
        float epsilon;
        cin >> epsilon;

        cout << "Enter eta: ";
        float eta;
        cin >> eta;

/*************************************************************************
*               Start writing the output file that contains the constrains      *
*                                       rnc.m                                   *
*************************************************************************/
        fout   << "%l = " << NumOfPoints << endl
                  << "%dimension = " << Dimension << endl
                  << "%C = " << C << endl
                  << "%eita = " << eta << endl
                  << "%epsilon = " << epsilon << endl;

        // Writing the minimization function
        // Here we start writing the declaration of the function
        fout << "function [f,g]= rnc(w)" << endl
                  << "k=[";

        // Finding the polynomial kernel
        ktemp = 0;
        for (i = 1; i <= NumOfPoints; i++)
        {
```

```
            for (int j = 1; j <= NumOfPoints; j++)
            {
                    // find the k(i,j) for the dimensional data
                    ktemp = 0;
                    for (int k=1; k <= Dimension; k++)
                            ktemp = ktemp + x[k][i]*x[k][j];
                    ktemp ++;
                    ktemp *= ktemp;
                    fout << ktemp << " ";
            }
            fout << ";" << endl;
    }
    fout << "];" << endl;

    // Finding alpha * k * alpha term
    fout << "f= 0.5*(";
    for (k = 1; k <= NumOfPoints; k++)
    {
            fout << "w(" << k << ")*(";
            for (int j = 1; j <= NumOfPoints; j++)
            {
                    fout <<"w(" << j << ")*k(" << j << "," << k << ")";
                    if (j < NumOfPoints)
                            fout << "+";
            }
            fout << ")";
            if (k < NumOfPoints)
                    fout << "+";
    }
    fout << ")+" << C << "*(";

    for (i=1; i <= NumOfPoints; i++)
    {
            fout << "(w(" << NumOfPoints+i << ")+w(" << 2*NumOfPoints+i <<
"))+";
    }
    fout << "0);" << endl;

    /************************************************************
                    Start writing the constrains
    ************************************************************/
    // First set of constrains
    float ytemp;
    for (i=1; i <= NumOfPoints; i++)
```

```
{
        ytemp = y[i] - epsilon;
        fout << "g(" << i << ")= sqrt(" << eta << ")* (";
        for (int k = 1; k <= NumOfPoints; k++)
        {
                fout << "w(" << k << ")*(";
                for (int j = 1; j <= NumOfPoints; j++)
                {
                        fout <<"w(" << j << ")*k(" << j << "," << k << ")";
                        if (j < NumOfPoints)
                                fout << "+";
                }
                fout << ")";
                if (k < NumOfPoints)
                        fout << "+";
        }

        fout << ")";
        for (int j = 1; j<= NumOfPoints; j++)
        {
                fout << "-w(" <<j<<")*k(" << j << "," << i << ")";
        }

        fout << "+sqrt(" << eta << ")*(";

        for (j = 1; j<= NumOfPoints; j++)
        {
                fout << "w(" <<j<<")+";
        }

        fout << "0)*sqrt(k(" << i << "," << i << "))-w(" << 3*NumOfPoints+1
<< ")-w(" << NumOfPoints + i << ")";

        if (ytemp > 0)
                fout << "+" << ytemp;
        else
                fout << "-" << -ytemp;

        fout << ";" << endl;
}

// Second set of Constrains
for (i=1; i <= NumOfPoints; i++)
{
```

```
                ytemp = -y[i] - epsilon;
                fout << "g(" << NumOfPoints + i << ")= -1*sqrt(" << eta << ")*(";
                for (int k = 1; k <= NumOfPoints; k++)
                {
                        fout << "w(" << k << ")*(";
                        for (int j = 1; j <= NumOfPoints; j++)
                        {
                                fout <<"w(" << j << ")*k(" << j << "," << k << ")";
                                if (j < NumOfPoints)
                                        fout << "+";
                        }
                        fout << ")";
                        if (k < NumOfPoints)
                                fout << "+";
                }

                fout << ")";

                for (int j = 1; j<= NumOfPoints; j++)
                {
                        fout << "+w(" <<j<<")*k(" << j << "," << i << ")";
                }

                fout << "-sqrt(" << eta << ")*(";

                        for (j = 1; j<= NumOfPoints; j++)
                {
                        fout << "w(" <<j<<")+";
                }
                fout << "0)*sqrt(k(" << i << "," << i << "))+w(" <<
3*NumOfPoints+1 << ")-w(" << 2*NumOfPoints + i << ")";

                if (ytemp > 0)
                        fout << "+" << ytemp;
                else
                        fout << "-" << -ytemp;

                fout << ";" << endl;
        }

        // Writing the constrains related to ai,zi,z^i > 0
        for (i = 2*NumOfPoints+1; i <= 4*NumOfPoints; i++)
        {
                fout << "g(" << i << ")= -w(" << i-NumOfPoints << ");" << endl;
```

```
        }

        /*************************************************************
                                    End of File
        *************************************************************/
        fout.close();
        return 0;
}
```

```
%This code solves for the alpha_alphahat in the objective function.

#include <fstream.h>
#include <string.h>
/*************************************************************************
This code generates constraints equations for the Robust LP SVM Rgression (alpha-
alpha hat) with n dimensional input.
The maximum number of input dimension is 30, with maximum of 1000 data points.
        w(1) ... w(n) : represents ai
        w(n+1)...w(2n) : represents a^i
        w(2n+1)...w(3n) : represents zi
        w(3n+1)...w(4n) : represents z^i
        w(4n+1) : represents b
*************************************************************************/

int main()
{
        ifstream fin;
        ofstream fout;
        char iFileName[50], temp[53];

        // Reading the data file name in order to read the data
        cout << "Enter the data file name: ";
        cin >> iFileName;
        fin.open(iFileName);

        // The file name used to store the constrains is called 'rnc.m'
        strcpy(temp, "rnc");
        strcat(temp, ".m");
        fout.open(temp);


        // Read how many points we want to use for training the SVM
        cout << "Enter the number of points: ";
        int NumOfPoints;
        cin >> NumOfPoints;

        /************************************************************
                        Start reading the data from the data file
        ************************************************************/
        float *y;
        int Dimension = 2;
        float x[31][1001];      // 30 dimensions, 1000 data points
        y = new float [NumOfPoints + 1];
```

```
cout << "Enter the dimension: ";
cin >> Dimension;

for (int i = 1; i <= NumOfPoints; i++)
{
        for (int j=1; j <= Dimension; j++)
                fin >> x[j][i];
        fin >> y[i];
}

fin.close();

// Read the C, epsilon and eita.
cout << "Enter the value of C: ";
int C;
cin >> C;

cout << "Enter epsilon: ";
float epsilon;
cin >> epsilon;

cout << "Enter eta: ";
float eta;
cin >> eta;

/*****************************************************************
 *       Start writing the output file that contains the constrains      *
 *                                  rnc.m                                *
*****************************************************************/
fout << "%l = " << NumOfPoints << endl
        << "%dimension = " << Dimension << endl
        << "%C = " << C << endl
        << "%eita = " << eta << endl
        << "%epsilon = " << epsilon << endl;

// Start writing the declaration of the function
fout << "function [f,g]= rnc(w)" << endl
        << "k=[";

// Finding the polynomial kernel
ktemp = 0;
for (i = 1; i <= NumOfPoints; i++)
{
```

```
            for (int j = 1; j <= NumOfPoints; j++)
            {
                    // find the k(i,j) for the dimensional data
                    ktemp = 0;
                    for (int k=1; k <= Dimension; k++)
                            ktemp = ktemp + x[k][i]*x[k][j];
                    ktemp ++;
                    ktemp *= ktemp;
                    fout << ktemp << " ";

            }
            fout << ";" << endl;
    }

    fout << "];" << endl;

    // finding the alpha + alphahat term

    fout << "f= (";
    for (k = 1; k <= NumOfPoints; k++)
    {
            fout << "(w(" << k << ")+w(" << NumOfPoints+k << "))";
            if (k < NumOfPoints)
                    fout << "+";

    }
    fout << ")+" << C << "*(";

    for (i=1; i <= NumOfPoints; i++)
    {
            fout << "(w(" << 2*NumOfPoints+i << ")+w(" << 3*NumOfPoints+i
<< "))";
            if (i < NumOfPoints)
                    fout << "+";
    }
    fout << ");" << endl;

    /*************************************************************
                            Start writing the constrains
    *************************************************************/
    // First set of constrains
    float ytemp;
    for (i=1; i <= NumOfPoints; i++)
    {
```

```
                ytemp = y[i] - epsilon;
                fout << "g(" << i << ")= sqrt(" << eta << ")* (";
                for (int k = 1; k <= NumOfPoints; k++)
                {
                        fout << "w(" << k << ")*(";
                        for (int j = 1; j <= NumOfPoints; j++)
                        {
                                fout <<"w(" << j << ")*k(" << j << "," << k << ")";
                                if (j < NumOfPoints)
                                        fout << "+";
                        }
                        fout << ")";
                        if (k < NumOfPoints)
                                fout << "+";
                }

                fout << ")";
                for (int j = 1; j<= NumOfPoints; j++)
                {
                        fout << "-(w(" <<j<<")-w(" << NumOfPoints+j << "))*k(" << j
<< "," << i << ")";
                }

                fout << "+sqrt(" << eta << ")*(";

                for (j = 1; j<= NumOfPoints; j++)
                {
                        fout << "(w(" <<j<<")-w(" << NumOfPoints + j << "))";
                        if (j < NumOfPoints)
                                fout << "+";
                }

                fout << ")*sqrt(k(" << i << "," << i << "))-w(" << 4*NumOfPoints+1
<< ")-w(" << 2*NumOfPoints + i << ")";

                if (ytemp > 0)
                        fout << "+" << ytemp;
                else
                        fout << "-" << -ytemp;

                fout << ";" << endl;
        }

        // Second set of Constrains
```

```
        for (i=1; i <= NumOfPoints; i++)
        {
                ytemp = y[i] - epsilon;
                fout << "g(" << NumOfPoints+i << ")= -sqrt(" << eta << ")* (";
                for (int k = 1; k <= NumOfPoints; k++)
                {
                        fout << "w(" << k << ")*(";
                        for (int j = 1; j <= NumOfPoints; j++)
                        {
                                fout <<"w(" << j << ")*k(" << j << "," << k << ")";
                                if (j < NumOfPoints)
                                        fout << "+";
                        }
                        fout << ")";
                        if (k < NumOfPoints)
                                fout << "+";
                }

                fout << ")";
                for (int j = 1; j<= NumOfPoints; j++)
                {
                        fout << "+(w(" <<j<<")-w(" << NumOfPoints+j << "))*k(" <<
j << "," << i << ")";
                }

                fout << "-sqrt(" <<eta << ")*(";

                for (j = 1; j<= NumOfPoints; j++)
                {
                        fout << "(w(" <<j<<")-w(" << NumOfPoints + j << "))";
                        if (j < NumOfPoints)
                                fout << "+";
                }

                fout << ")*sqrt(k(" << i << "," << i << "))+w(" << 4*NumOfPoints+1
<< ")-w(" << 2*NumOfPoints + i << ")";

                if (ytemp > 0)
                        fout << "-" << ytemp;
                else
                        fout << "+" << -ytemp;

                fout << ";" << endl;
        }
```

```
        // Writing the constrains related to ai, a^i,zi,z^i > 0
        for (i = 2*NumOfPoints+1; i <= 6*NumOfPoints; i++)
        {
                fout << "g(" << i << ")= -w(" << i-2*NumOfPoints << ");" << endl;
        }

        /*************************************************************
                                        End of File
        *************************************************************/
        fout.close();
        return 0;
}
```

MATLAB Code for Regression Analysis (alpha_k_alpha)

```
%This code solves the regression RSV

%This is a matlab script file that runs the constr command and finds the yhat
%for the regression

n = 30;          %Number of training Points in the data
dimension =1;    %Number of Dimension for X
eita = 0;                %eita
alldata = 100; %The total number of the data sample (training + testing)
gridoption = 1;    % 1: turn grids on, 0: turn grids off

%Prepare the parameters for the constr command
w0=zeros(1,3*n+1);   %the initial values
vlb=-inf;                        %lower bound, unrestricted
vub=inf;                         %upper bound, unrestricted
options(14)=100000;  %Maximum number of itirations
w=constr('rnc',w0,options,vlb,vub);   %Start the analysis
w'               %Show the results

% Start calculating yhat
% first, load the data from the data file.
alpha = w(1:n);
load lynx.txt
x = lynx(1:n,1:dimension);
y = lynx(1:n,dimension+1);
xdata =lynx(1:alldata,1:dimension);
yy = lynx(1:alldata,dimension+1);
[q,r] = size(xdata);
b = w(3*n+1);
xx = x;

% second, call the objective function
yhat = ftest(n,dimension,x,xx,alpha,eita,b);      %yhat function. x is the training data,
xx is the input data
yhat             % Show the result

% here we calculate the Mean Square Error
trainingmse = y-yhat;
trainingmse = trainingmse.^2;
trainingmse = sum(trainingmse);
trainingmse = trainingmse/n;
trainingmse              %Show the result
```

```
%This code graphs the regression for SVR

% This is a matlab script file that is used to plot
% 1) Original y vs training yhat
% 2) Original y vs testing yhat
% in order to compare the results of the SVM analysis with the original data

training = n;
testing = alldata - n;
yhatdata = ftest(n,dimension,x,xdata,alpha,eita,b); %find yhat for all the data samples

% Figure 1 is having two sub plots:
% 1) Original y vs index
% 2) training yhat vs index
figure(1);
clf;
temp = [0:q-1];

% First subplot, original y
subplot(2,1,1);
plot(temp,yy,'b');
if (gridoption == 1)
   grid;
end

title('original data');
xlabel('index');
ylabel('y');

%second subplot: training yhat compared to original y
subplot(2,1,2);
plot(temp,yy,'b');
hold on;
plot(temp(1:training),yhat(1:training),'r');
if (gridoption == 1)
   grid;
end
title('comparison between the original data (blue) & the training points (red)');
xlabel('index');
ylabel('y, yhat');
hold off;
% Figure 2 is having two sub plots:
% 1) Original y vs index
% 2) testing yhat vs index
```

```
figure(2);
clf
% First subplot, original y
subplot(2,1,1);
plot(temp,yy,'b');
if (gridoption == 1)
   grid;
end
title('original data');
xlabel('index');
ylabel('y');

%second subplot: training yhat compared to original y
subplot(2,1,2);
plot(temp,yy,'b');
hold on;
plot(temp(training+1:q),yhatdata(training+1:q),'r');
if (gridoption == 1)
   grid;
end
title('comparison between the original data (blue) & the testing points (red)');
xlabel('index');
ylabel('y, yhat');
hold off;

% Calculate the MSE for the testing sample
testingmse = yy(training+1:q)- yhatdata(training+1:q);
testingmse = testingmse.^2;
testingmse = sum(testingmse);
testingmse = testingmse/testing;
testingmse
```

```
%This code finds the yhat for the regression RSV

% This function represents the objective function for regression.
% Call this function in order to find the yhat after running the constr command
% This file is called by the fcode.m to find the training yhat, and by the
% fgraph.m in order to find the testing yhat

function y=ftest(n,d,x,xx,alpha,eita,b)
%x is the data matrix (size of X is n)
%xx is the input matrix (size of xx is undefined)
%n number of points
%d dimension
temp = 0;
% Here we find the term Sum(alpha * |x|)
for i=1:n
   temp = temp + alpha(i)*((xx*x(i,1:d)' + 1).^2);
end

[q,r] = size(xx);
for i=1:q
   k(q) = xx(i,1:d)*xx(i,1:d)' + 1;
end

% Here we find the term sqrt(eita)*sqrt(k(x,x))*alpha
for i=1:n
   temp = temp - k'.*sqrt(eita)*alpha(i);
end

% Add b
temp = temp + b;

% The final result
y = temp;
```

MATLAB Code for Regression Analysis (alpha_ alphahat)

```
%This code solves the regression for RSV with alpha_alphahat

%This is a matlab script file that runs the constr command and finds the yhat
%for the regression, using the alpha*k*alpha term

n = 10;    %Number of training Points in the data
dimension =2;    %Number of Dimension for X
eita = 0;                %eita
alldata = 100;  %The total number of the data sample (training + testing)
gridoption = 1;     % 1: turn grids on, 0: turn grids off

%Prepare the parameters for the constr command
w0=zeros(1,4*n+1);   %the initial values
vlb=-inf;                        %lower bound, unrestricted
vub=inf;                         %upper bound, unrestricted
% you can add the option 14 code line in place of the line below
%options=optimset('MaxFunEvals',600);      % remove this line and put the option 14
in its place
options(14)=100000;              %Maximum number of itirations
w=constr('rnc',w0,options,vlb,vub);              %Start the analysis
w'               %Show the results

alpha = w(1:n);
alphahat = w(n+1:2*n);
b = w(4*n+1);

% Start calculating yhat
% first, load the data from the data file.

load data.txt
x = data(1:n,1:dimension);
y = data(1:n,dimension+1);
xdata =data(1:alldata,1:dimension);
yy = data(1:alldata,dimension+1);
[q,r] = size(xdata);
xx = x;

% second, call the objective function
yhat = ftest(n,dimension,x,xx,alpha,alphahat,eita,b);      %yhat function, x is the base
data, xx is the input data
yhat
```

```
%here we calculate the Mean Square Error
trainingmse = y-yhat;
trainingmse = trainingmse.^2;
trainingmse = sum(trainingmse);
trainingmse = trainingmse/n;
trainingmse
```

```
%This code graphs the regression for SVR with alpha_alphahat

% This is a matlab script file that is used to plot
% 1) Original y vs training yhat
% 2) Original y vs testing yhat
% in order to compare the results of the SVM analysis with the original data
training = n;
testing = alldata - n;
yhatdata = ftest(n,dimension,x,xdata,alpha,alphahat,eita,b);

% Figure 1 is having two sub plots:
% 1) Original y vs index
% 2) training yhat vs index
figure(1);
clf;
temp = [0:q-1];


% First subplot, original y
subplot(2,1,1);
plot(temp,yy,'b');
if (gridoption == 1)
   grid;
end

title('original data');
xlabel('index');
ylabel('y');

%second subplot: training yhat compared to original y
subplot(2,1,2);
plot(temp,yy,'b');
hold on;
plot(temp(1:training),yhat(1:training),'r');
if (gridoption == 1)
   grid;
end
title('comparison between the original data (blue) & the training points (red)');
xlabel('index');
ylabel('y, yhat');
hold off;
% Figure 2 is having two sub plots:
% 1) Original y vs index
% 2) testing yhat vs index
```

```
figure(2);
clf
% First subplot, original y
subplot(2,1,1);
plot(temp,yy,'b');
if (gridoption == 1)
   grid;
end
title('original data');
xlabel('index');
ylabel('y');

%second subplot: training yhat compared to original y
subplot(2,1,2);
plot(temp,yy,'b');
hold on;
plot(temp(training+1:q),yhatdata(training+1:q),'r');
if (gridoption == 1)
   grid;
end
title('comparison between the original data (blue) & the testing points (red)');
xlabel('index');
ylabel('y, yhat');
hold off;


% Calculate the MSE for the testing sample
testingmse = yy(training+1:q)- yhatdata(training+1:q);
testingmse = testingmse.^2;
testingmse = sum(testingmse);
testingmse = testingmse/testing;
testingmse
```

```
%This code finds yhat for regression RSV with alpha_alphahat

% This function represents the objective function for regression.
% Call this function in order to find the yhat after running the constr command
% This file is called by the fcode.m to find the training yhat, and by the
% fgraph.m in order to find the testing yhat

function y=ftest(n,d,x,xx,alpha,alphahat,eita,b)
%x is the data matrix (size of X is n)
%xx is the input matrix (size of xx is undefined)
%n number of points
%d dimension
temp = 0;
% Here we find the term Sum((ai - a^i) * |x|)
for i=1:n
   temp = temp + (alpha(i)-alphahat(i))*((xx*x(i,1:d)' + 1).^2);
end

[q,r] = size(xx);
for i=1:q
   k(q) = xx(i,1:d)*xx(i,1:d)' + 1;
end

% Here we find the term sqrt(eita)*sqrt(k(x,x))*(ai-ai)
for i=1:n
   temp = temp - k'.*sqrt(eita)*(alpha(i)-alphahat(i));
end

temp = temp + b;
% The final result
y = temp;
```

Appendix C

Tables for Classification

x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Table 1- The AND Function

x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	1

Table 2- The XOR problem

Trade Off C	Uncertainty Eita η	W_1	W_2	Alpha	Offset b	Comment
10,000	0	1	1	1	-1	Optimal Separation
10,000	0.1	1	1	1	-0.5528	Separation
10,000	0.2	1	1	1	-0.3675	Separation
10,000	0.3	1	1	1	-0.2254	Separation
10,000	0.4	1	1	1	-0.1056	Separation
10,000	0.5	1	1	1	0	2 SV at the point (o)
10,000	0.6	1	1	1	0.0954	2 SV at the point (o)
10,000	0.7	1	1	1	0.1832	No separation
10,000	1.0	1	1	1	0.4142	No separation

Table 3- The Robust Case for the AND function

Trade Off C	Uncertainty Eita η	W_1	W_2	Alpha	Offset b	Comments
10,000	0	1	1	1	-1	Optimal Separation
10,000	0.1	1	0.7597	1	-1.3162	Separation
10,000	0.2	1	0.6910	1	-1.4472	Separation
10,000	0.3	1	0.6461	1	-1.5477	Separation
10,000	0.35	1	0.6283	1	-1.5916	1 SV at the point (+).
10,000	0.4	1	0.6126	1	-1.6325	1 SV at the point (+).
10,000	0.5	1	0.5858	1	-1.7071	No Separation.
10,000	0.6	1	0.5635	1	-1.7746	No Separation
10,000	0.7	1	0.5445	1	-1.8367	No Separation
10,000	1.0	1	0.5000	1	-2.000	No Separation

Table 4- The Sensitivity Analysis Case for the AND function.

Trade off C	Uncertainy Eita η	Alpha α_i i=1,..,4	z_i Slack variable	b The offset	Error% Misclass	Separation
10,000	0	$\alpha_1=\alpha_2=.125$ $\alpha_3=\alpha_4=.125$	0	0	0	Yes
10,000	0.1	$\alpha_1=\alpha_4=.1507$ $\alpha_2=\alpha_3=.0993$	0	0.0199	0	Yes
10,000	0.2	$\alpha_1=\alpha_4=.1701$ $\alpha_2=\alpha_3=.0799$	0	0.0468	0	Yes
10,000	0.3	$\alpha_1=\alpha_4=.1874$ $\alpha_2=\alpha_3=.0626$	0	0.1151	0	Yes
10,000	0.4	$\alpha_1=\alpha_4=.1093$ $\alpha_2=\alpha_3=.1407$	0	0.5218	0	Yes
10,000	0.5	$\alpha_1=\alpha_4=.0537$ $\alpha_2=\alpha_3=.1963$	0	0.8605	0	Yes
10,000	0.6	$\alpha_1=\alpha_4=.0262$ $\alpha_2=\alpha_3=.2238$	0	1.0298	0	Yes
10,000	0.7	$\alpha_1=\alpha_4=.0140$ $\alpha_2=\alpha_3=.2360$	0	1.0915	0	Yes
10,000	0.8	$\alpha_1=\alpha_4=.0082$ $\alpha_2=\alpha_3=.2418$	0	1.1090	0	Yes
10,000	0.9	$\alpha_1=\alpha_4=.0053$ $\alpha_2=\alpha_3=.2447$	0	1.1077	0	Yes
10,000	1.0	$\alpha_1=\alpha_2=.125$ $\alpha_3=\alpha_4=.125$	0	0.7071	0	Yes
10,000	1.05	$\alpha_1=\alpha_4=.0845$ $\alpha_2=\alpha_3=.1655$	0	0.7676	0	Yes
10,000	1.06	$\alpha_1=\alpha_4=.2462$ $\alpha_2=\alpha_3=.0038$	0	1.1730	50	No Separation

Table 5- Results for the XOR problem (Non-Linear Case) using the uncertainty in the feature space.

Trade off C	Uncertainty Eita η	Training Error %	#of Training points misclassified	Testing Error %	# of Testing points misclassified
1000	0	10	4	20	4
1000	0.03	10	4	25	5
8500	0.05	12.5	5	20	4
8500	0.07	12.5	5	25	5
8500	0.09	7.5	3	25	5

Table 6- Results for the Echocardiogram Classification Application (Original Data).

Trade off C	Uncertainty Eita η	Training Error %	#of Training points misclassified	Testing Error %	# of Testing points misclassified
1000	0	17.5	7	30	5
1000	0.03	27	11	35	7
3000	0.05	20	8	30	6
5000	0.07	17.5	7	35	7
8500	0.09	15	6	25	5

Table 7- Results for the Echocardiogram1 Classification Application (Generated Randomly).

Trade off C	Uncertainty Eita η	Training Error %	#of Training points misclassified	Testing Error %	# of Testing points misclassified
1000	0	0	0	40	8
10	0.03	0	0	40	8
1000 or 2000	0.05	0	0	40	8
1000	0.07	0	0	40	8
500	0.09	0	0	40	8

Table 8- Results for the Echocardiogram2 Classification Application (Generated Randomly).

Trade off C	Uncertainty Eita η	Training Error %	#of Training points misclassified	Testing Error %	# of Testing points misclassified
1000	0	0	0	5	2
1000	0.01	0	0	5	2
1000	0.02	0	0	5	2
1000	0.05	0	0	5	2

Table 9- Results for the Breast Cancer Classification Application

Appendix D

Tables for Regression

C	Epsilon	Eita	mse(training)	mse(testing)	Data#
20000	0.0001	0	6.98E-20	8.33E-20	Original
20000	0.0001	0	2.65E-18	1.76E-18	1
20000	0.0001	0	6.28E-20	6.91E-20	2
20000	0.0001	0	1.50E-19	1.82E-19	3
20000	0.0001	0	4.41E-19	4.59E-19	4
20000	0.0001	0	8.14E-18	9.51E-18	5
20000	0.0001	0	3.30E-18	6.08E-18	6
20000	0.0001	0	1.92E-19	1.63E-19	7
20000	0.0001	0	2.23E-20	2.79E-20	8
20000	0.0001	0	1.92E-19	2.25E-19	9
20000	0.0001	0	6.89E-21	7.63E-21	10
20000	0.0001	0.000001	2.00E-06	2.00E-06	Original
20000	0.0001	0.000001	2.00E-06	2.00E-06	1
20000	0.0001	0.000001	2.00E-06	2.00E-06	2
20000	0.0001	0.000001	2.00E-06	2.00E-06	3
20000	0.0001	0.000001	2.00E-06	2.00E-06	4
20000	0.0001	0.000001	2.00E-06	2.00E-06	5
20001	0.0001	0.000001	2.00E-06	2.00E-06	6
20000	0.0001	0.000001	2.00E-06	2.00E-06	7
20000	0.0001	0.000001	2.00E-06	2.00E-06	8
20000	0.0001	0.000001	2.00E-06	2.00E-06	9
20000	0.0001	0.000001	2.00E-06	2.00E-06	10

Table 10- Experiments for the synthetic regression problem.

20000	0.0001	0.00001	2.00E-05	2.00E-05	Original
20000	0.0001	0.00001	2.00E-05	2.00E-05	1
20001	0.0001	0.00001	2.00E-05	2.00E-05	2
20001	0.0001	0.00001	2.00E-05	2.00E-05	3
20000	0.0001	0.00001	2.00E-05	2.00E-05	4
20000	0.0001	0.00001	2.00E-05	2.00E-05	5
20000	0.0001	0.00001	2.00E-05	2.00E-05	6
20000	0.0001	0.00001	2.00E-05	2.00E-05	7
20000	0.0001	0.00001	2.00E-05	2.00E-05	8
20000	0.0001	0.00001	2.00E-05	2.00E-05	9
20000	0.0001	0.00001	2.00E-05	2.00E-05	10
20000	0.0001	0.0001	2.00E-04	2.00E-04	Original
20000	0.0001	0.0001	2.00E-04	2.00E-04	1
20000	0.0001	0.0001	2.00E-04	2.00E-04	2
20001	0.0001	0.0001	2.00E-04	2.00E-04	3
20000	0.0001	0.0001	2.00E-04	2.00E-04	4
20000	0.0001	0.0001	2.00E-04	2.00E-04	5
20000	0.0001	0.0001	2.00E-04	2.00E-04	6
20001	0.0001	0.0001	2.00E-04	2.00E-04	7
20000	0.0001	0.0001	2.00E-04	2.00E-04	8
20000	0.0001	0.0001	2.00E-04	2.00E-04	9
20000	0.0001	0.0001	2.00E-04	2.00E-04	10

Continuation of Table 10

1000	0.0001	0.1	1.67E-01	1.67E-01	Original
1001	0.0001	0.1	1.67E-01	1.67E-01	1
1000	0.0001	0.1	1.67E-01	1.67E-01	2
1000	0.0001	0.1	1.67E-01	1.67E-01	3
1000	0.0001	0.1	1.67E-01	1.67E-01	4
1000	0.0001	0.1	1.67E-01	1.67E-01	5
1000	0.0001	0.1	1.67E-01	1.67E-01	6
1000	0.0001	0.1	1.67E-01	1.67E-01	7
1000	0.0001	0.1	1.67E-01	1.67E-01	8
1000	0.0001	0.1	1.67E-01	1.67E-01	9
1000	0.0001	0.1	1.67E-01	1.67E-01	10
1000	0.0001	0.5	5.00E-01	5.00E-01	Original
1000	0.0001	0.5	5.00E-01	5.00E-01	1
1000	0.0001	0.5	5.00E-01	5.00E-01	2
1000	0.0001	0.5	5.00E-01	5.00E-01	3
1000	0.0001	0.5	5.00E-01	5.00E-01	4
1000	0.0001	0.5	5.00E-01	5.00E-01	5
1000	0.0001	0.5	5.00E-01	5.00E-01	6
1000	0.0001	0.5	5.00E-01	5.00E-01	7
1000	0.0001	0.5	5.00E-01	5.00E-01	8
1000	0.0001	0.5	5.00E-01	5.00E-01	9
1000	0.0001	0.5	5.00E-01	5.00E-01	10

Continuation of Table 10

1000	0.0001	1	5.00E-01	6.67E-01	Original
1000	0.0001	1	5.00E-01	6.67E-01	1
1000	0.0001	1	5.00E-01	6.67E-01	2
1000	0.0001	1	5.00E-01	6.67E-01	3
1000	0.0001	1	5.00E-01	6.67E-01	4
1000	0.0001	1	5.00E-01	6.67E-01	5
1000	0.0001	1	6.67E-01	6.67E-01	6
1000	0.0001	1	5.00E-01	6.67E-01	7
1000	0.0001	1	6.70E-01	6.70E-01	8
1000	0.0001	1	5.00E-01	6.67E-01	9
1000	0.0001	1	5.00E-01	6.67E-01	10

Continuation of Table 10

The Uncertainty Eita (η)	Average of Testing Mean square Error (mse)
0	1.69E-18
0.000001	2.00E-6
0.00001	2.00E-5
0.0001	2.00E-4
0.1	1.67E-01
0.5	5.00E-01
1	6.667E-01

Table 11- The Average Mean Square Error (mse) with different values of Eita (η) for the Synthetic Regression Problem

C	Epsilon	Eita	Training Error (mse)	Testing Error (mse)
100	0.0001	0	0.6358	0.3726
10,000	0.0001	0.0001	1.1261	0.7384

Table 12- Tests on the Lynx Regression Problem

Appendix E

Figures for Classification

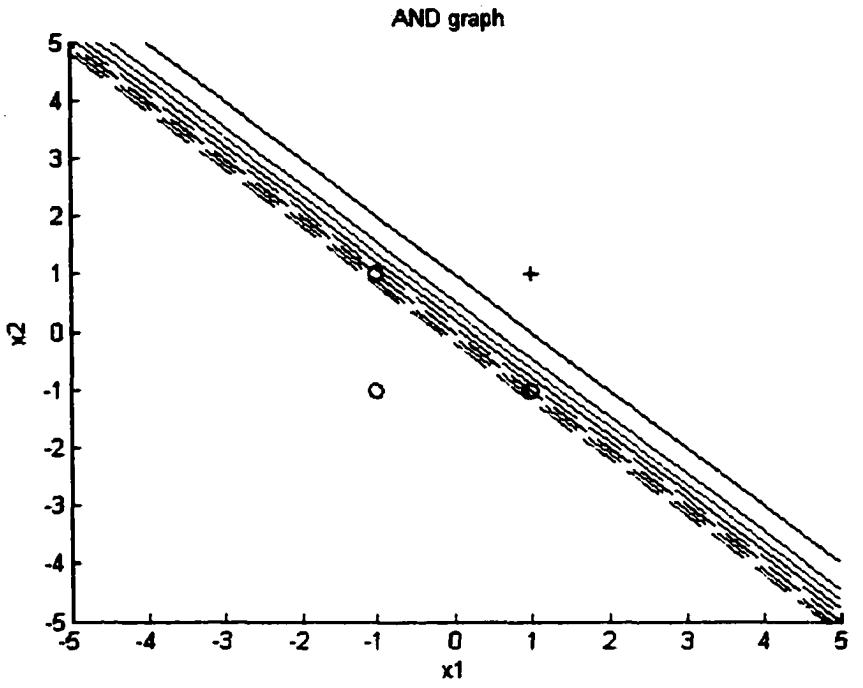


Fig1- The Robust Case for the AND Function with different values of the uncertainty (eita). Solid blue for eita = 0 (precise data). Solid red for eita = 0.1.Solid cyan for eita = 0.2. Solid green for eita = 0.3.Dash-red for eita = 0.4. Dash-blue for eita = 0.5.Dash-green for eita = 0.6.Dash-dot-blue for eita = 0.7. Solid Yellow for eita = 1.

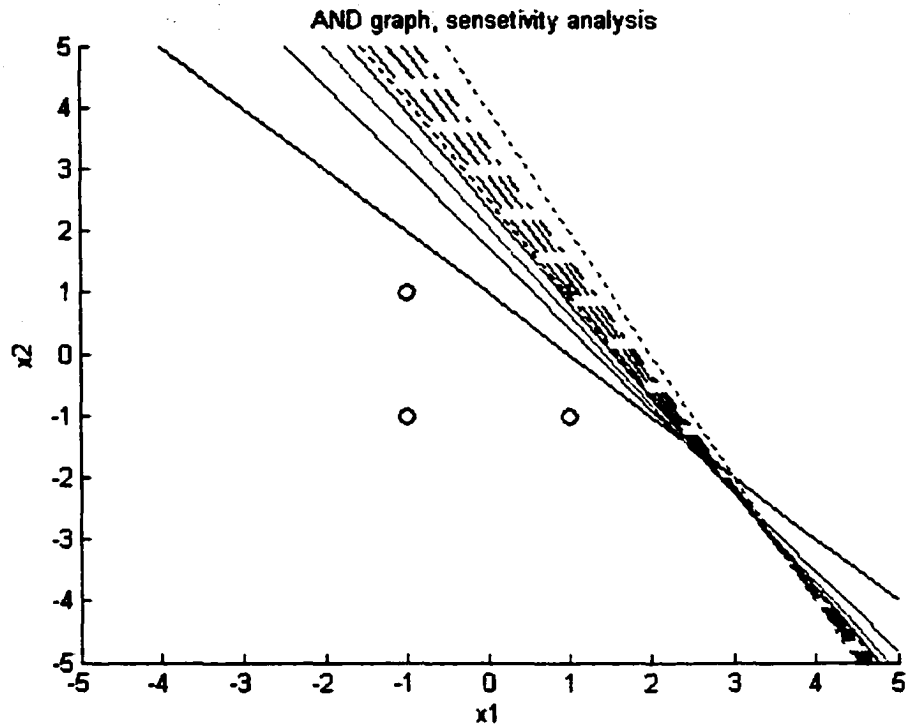


Fig-2 The And function (Sensitivity Analysis Case) with different values of the uncertainty (eita). Solid blue for eita = 0 (precise data). Solid red for eita = 0.1.Solid cyan for eita = 0.2. Solid green for eita = 0.3. Blue-dotted for eita = 0.35.Dash-red for eita = 0.4. Dash-blue for eita = 0.5. Dash-green for eita = 0.6.Dash-dot-blue for eita = 0.7. Dotted red &yellow for eita = 1.

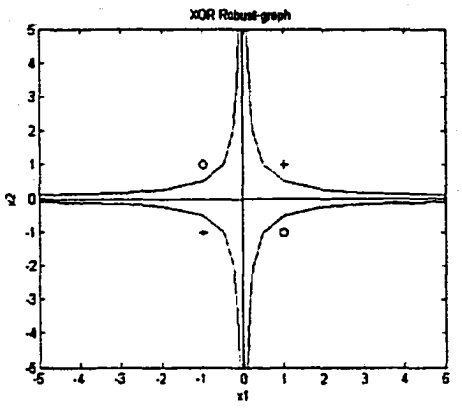


Fig3-XOR with eita = 0 (precise case)

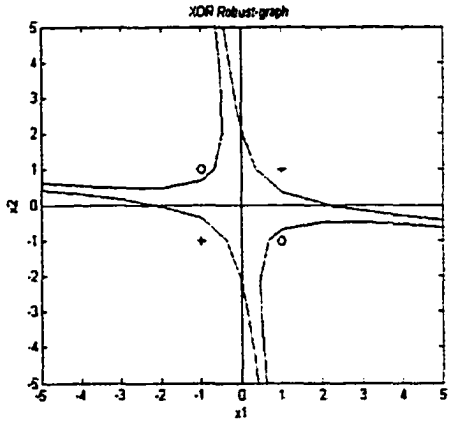


Fig4- XOR with eita = 0.1

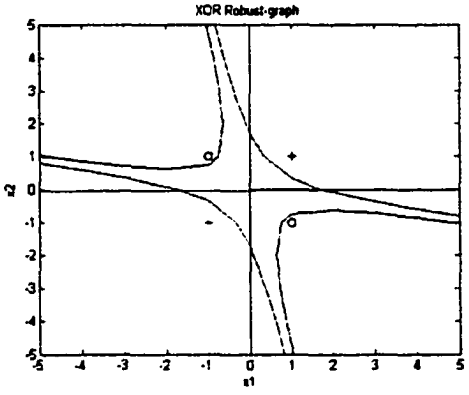


Fig5- XOR with eita = 0.2

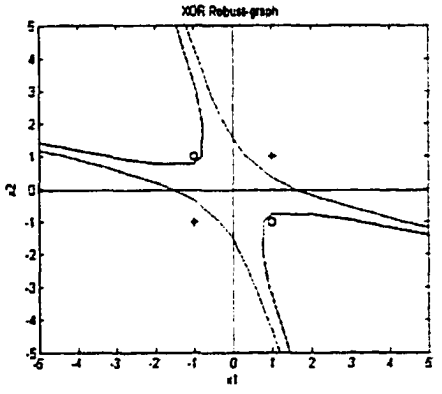


Fig6- XOR with eita = 0.3

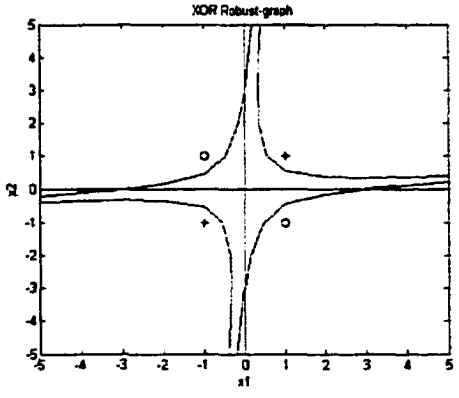


Fig7- XOR with eita = 0.4

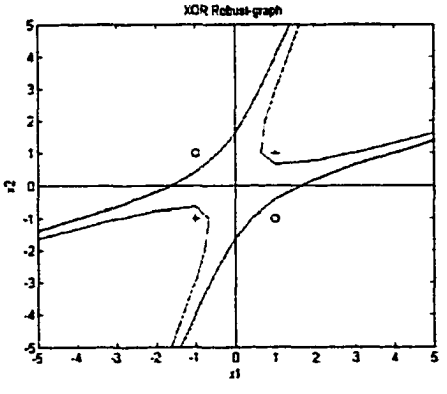


Fig8- XOR with eita = 0.5

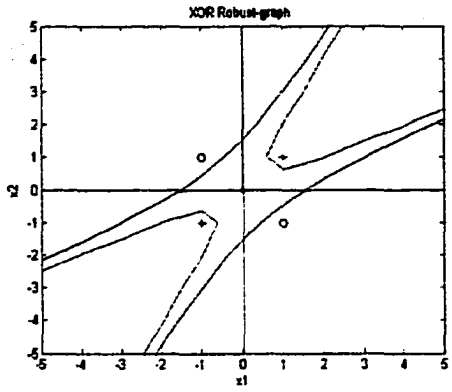


Fig9- XOR with eita = 0.6

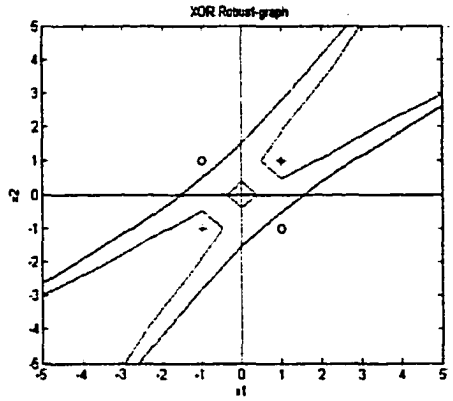


Fig10- XOR with eita = 0.7

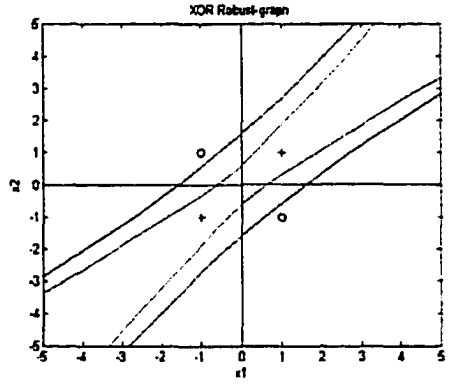


Fig11- XOR with eita = 0.8

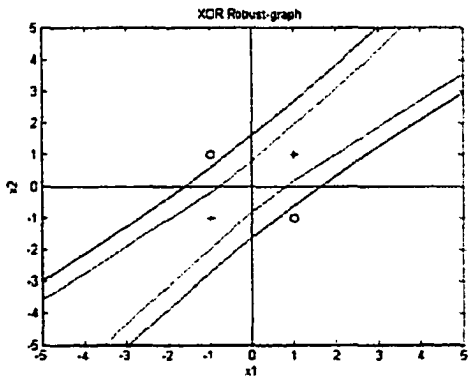


Fig12- XOR with eita = 0.9

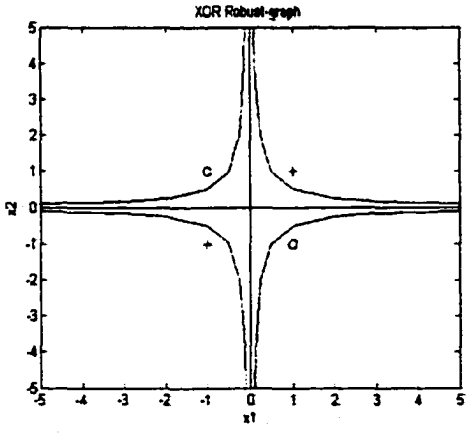


Fig13- XOR with eita = 1.0

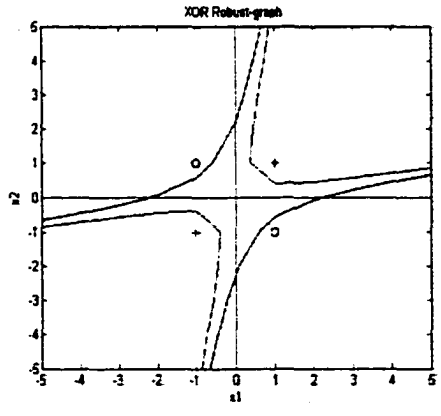


Fig14- XOR with eita = 1.05

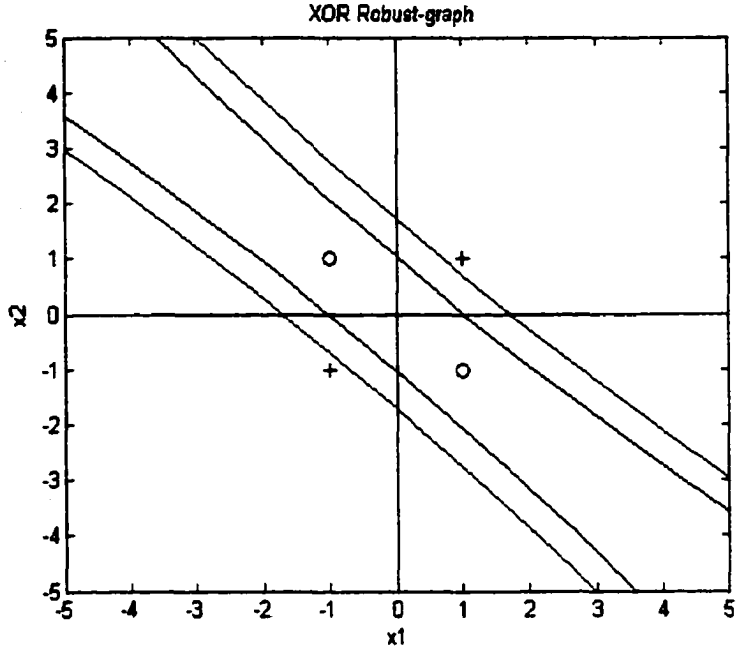


Fig15- XOR with eita = 1.06 (No separation)

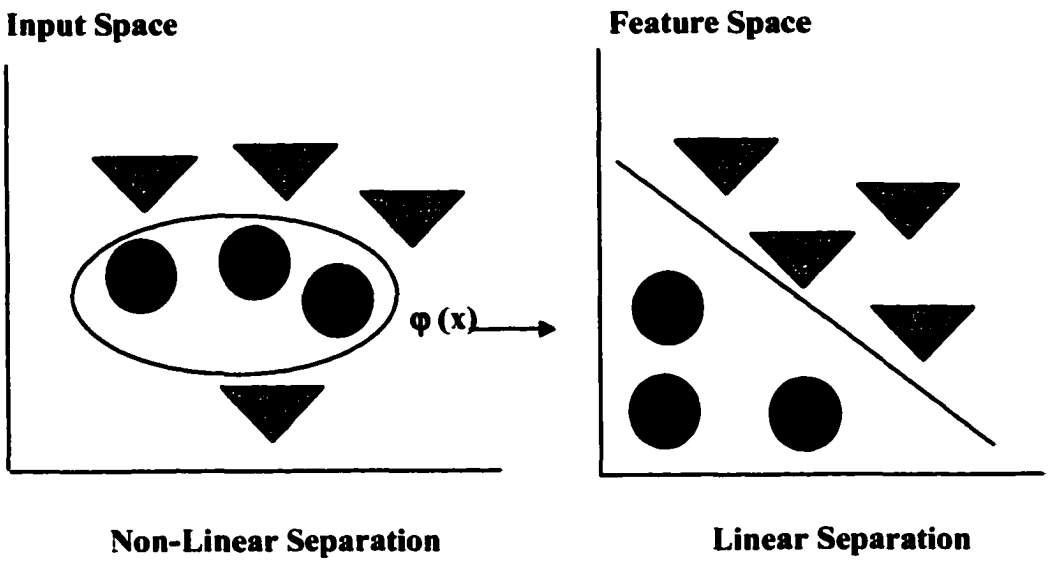


Figure 16 – The Kernel Method for Classification

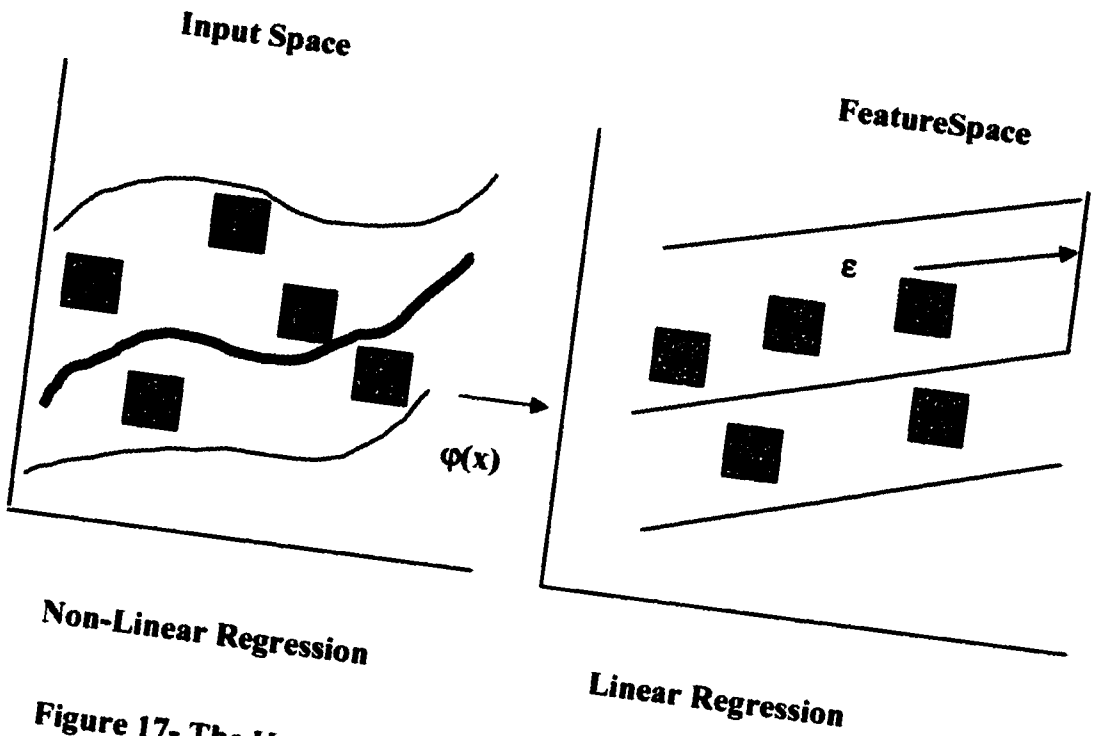


Figure 17- The Kernel Method for Regression

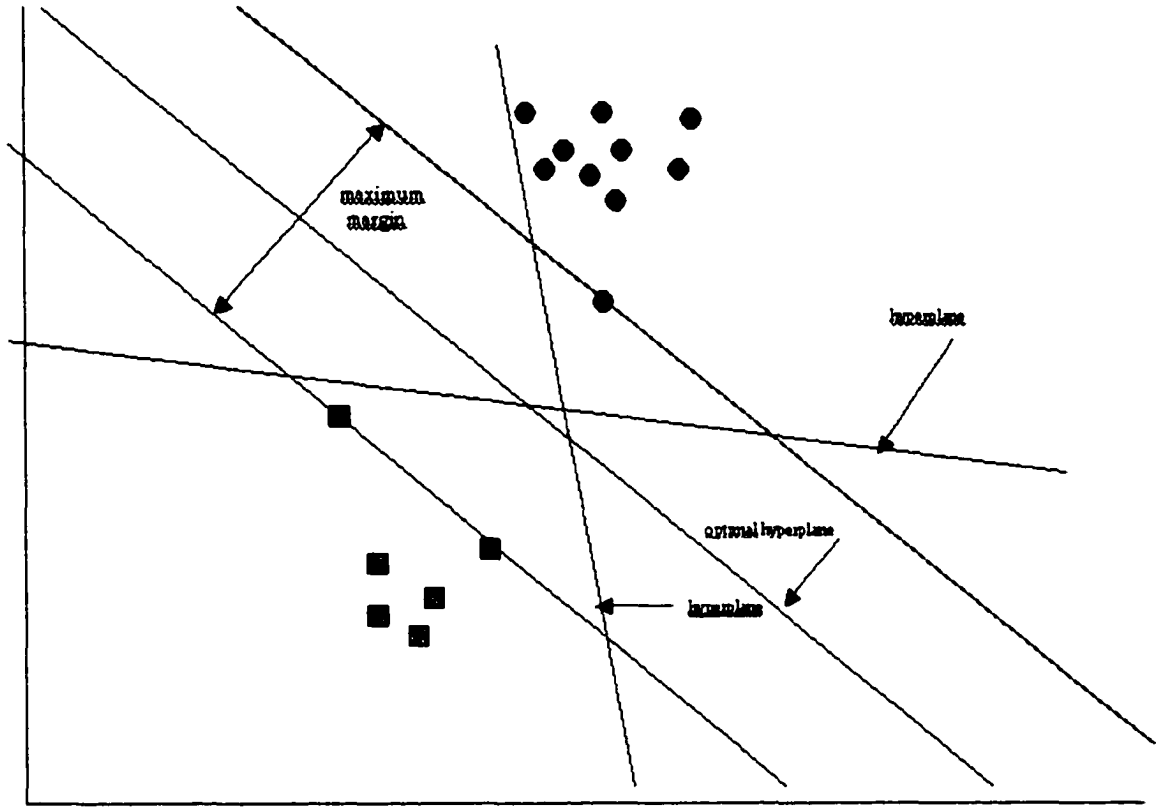


Figure 18- Separating hyperplane in two dimensional space. An optimal hyperplane in one with a maximum margin. The data points at the margin (indicated in black) are called the support vectors because they define the optimal hyperplane.

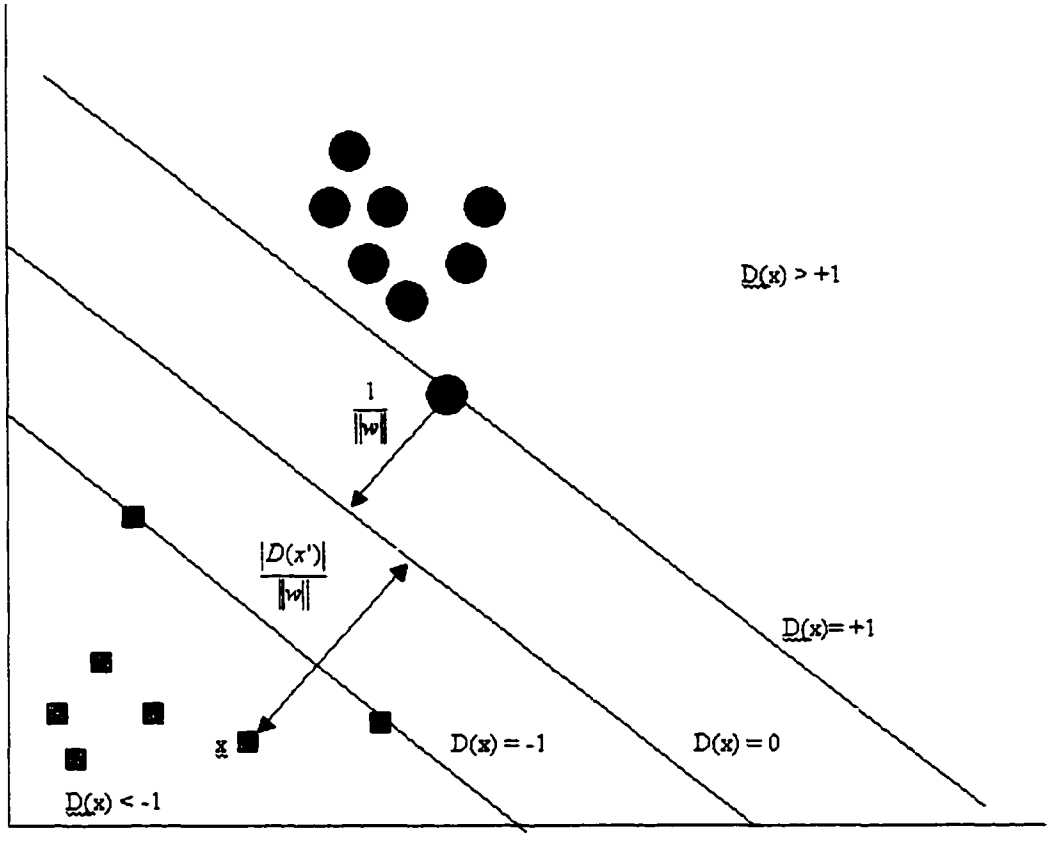


Figure19- The decision boundary of the optimal hyperplane is defined by Points x for which D (x) = 0. The distance between a hyperplane and any sample x' is $\frac{|D(x')|}{\|w\|}$. The distance between a support vector (which define the margin) and the optimal hyperplane is $\frac{1}{\|w\|}$.

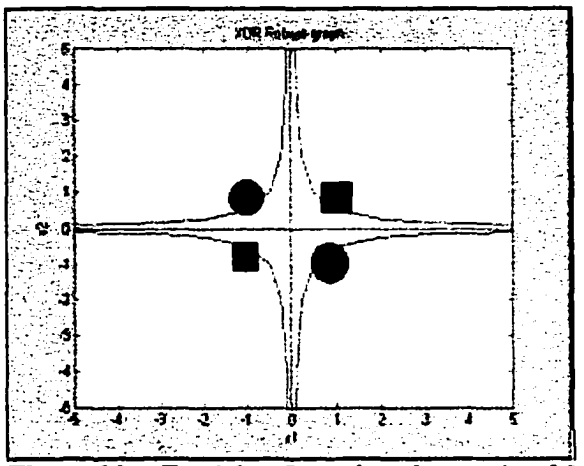


Figure20a- Decision function determined by the support vector machine with a feature space of order two polynomials. In the two-dimensional input space, the decision function is nonlinear.

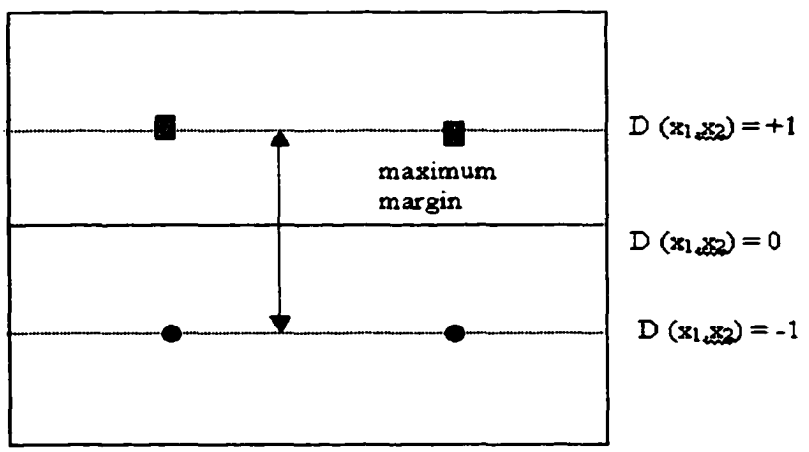


Figure20b- Decision function determined by the support vector machine with a feature space of order two polynomials. In the six-dimensional feature space, the decision function is linear with maximum margin.

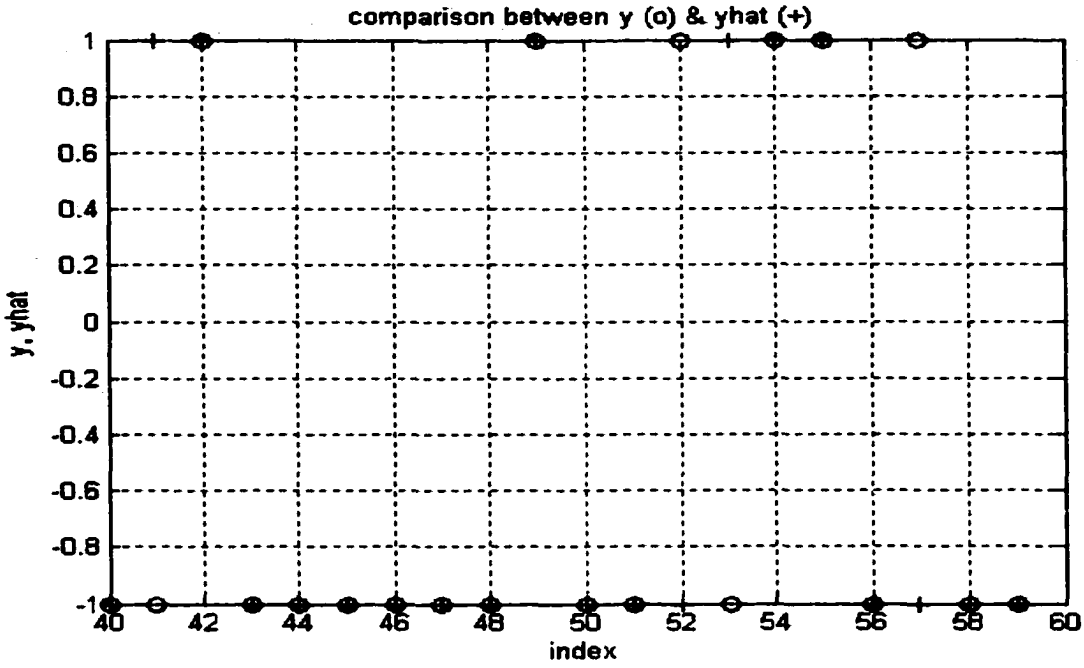


Figure21- Misclassified <u>testing</u> points with uncertainty eita = 0 for echo cardio data. There are 4 points misclassified. They are the empty circles.

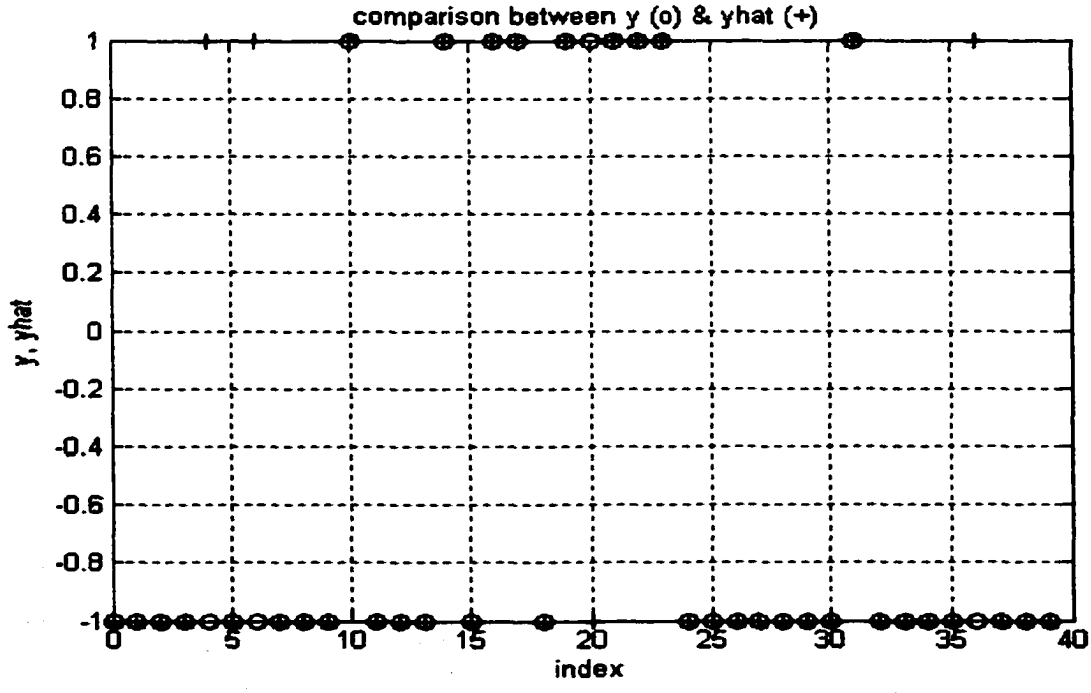


Figure22- Misclassified <u>training</u> points with uncertainty eita = 0 for echo cardio data. There are 4 points misclassified. They are the empty circles.

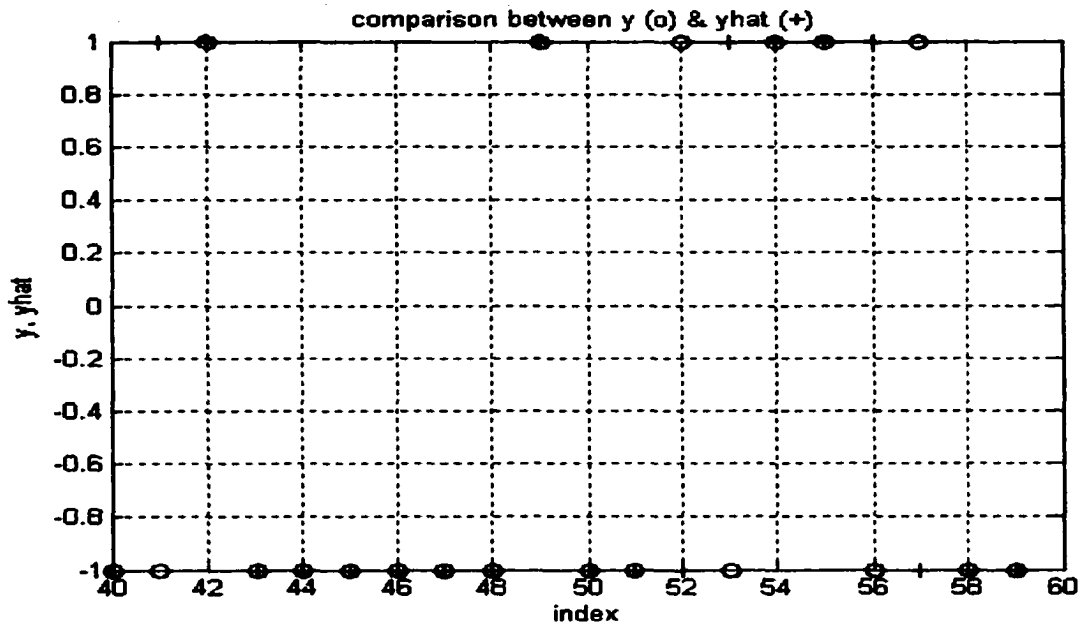


Figure23- Misclassified <u>testing</u> points with uncertainty eita = 0.03 for echo cardio data. There are 5 points misclassified. They are the empty circles.

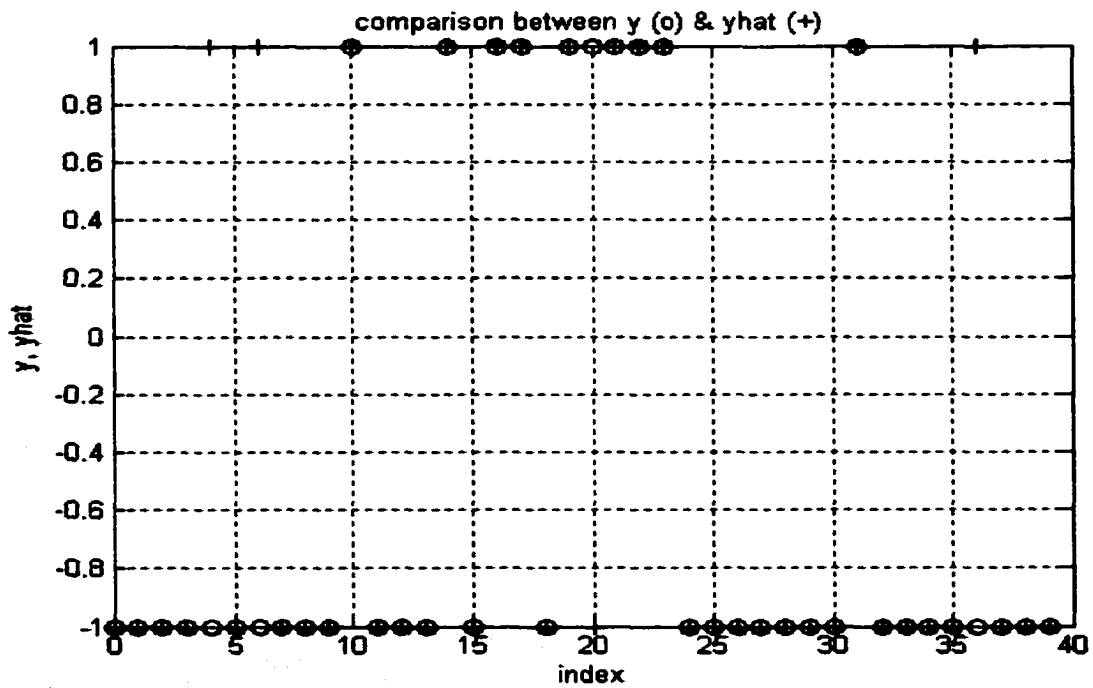


Figure24- Misclassified <u>training</u> points with uncertainty eita = 0.03 for echo cardio data. There are 4 points misclassified. They are the empty circles.

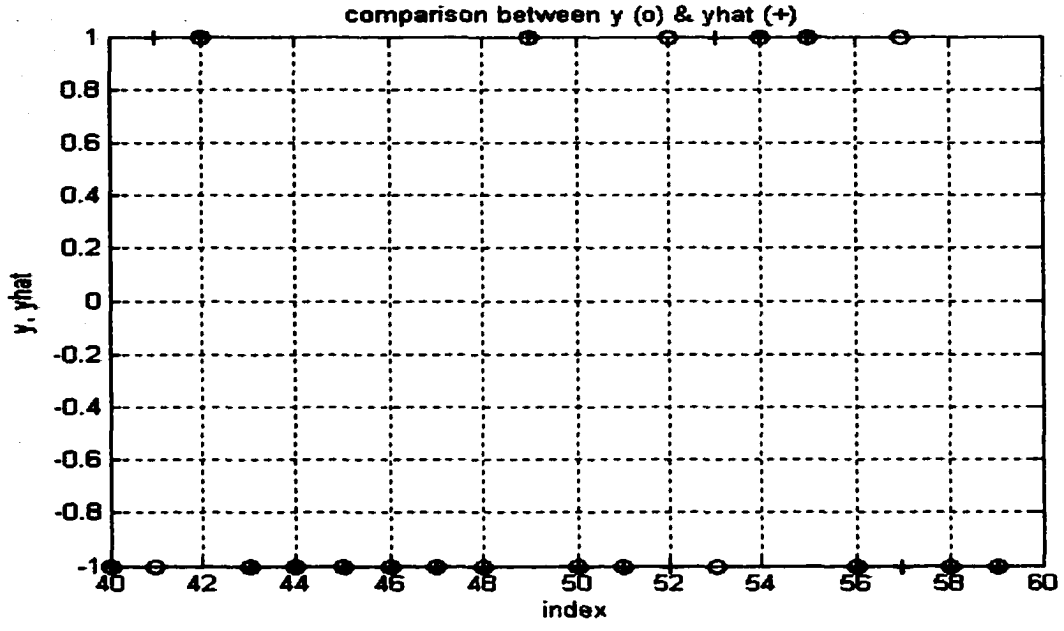


Figure25- Misclassified testing points with uncertainty eita = 0.05 for echo cardio data. There are 4 points misclassified. They are the empty circles.

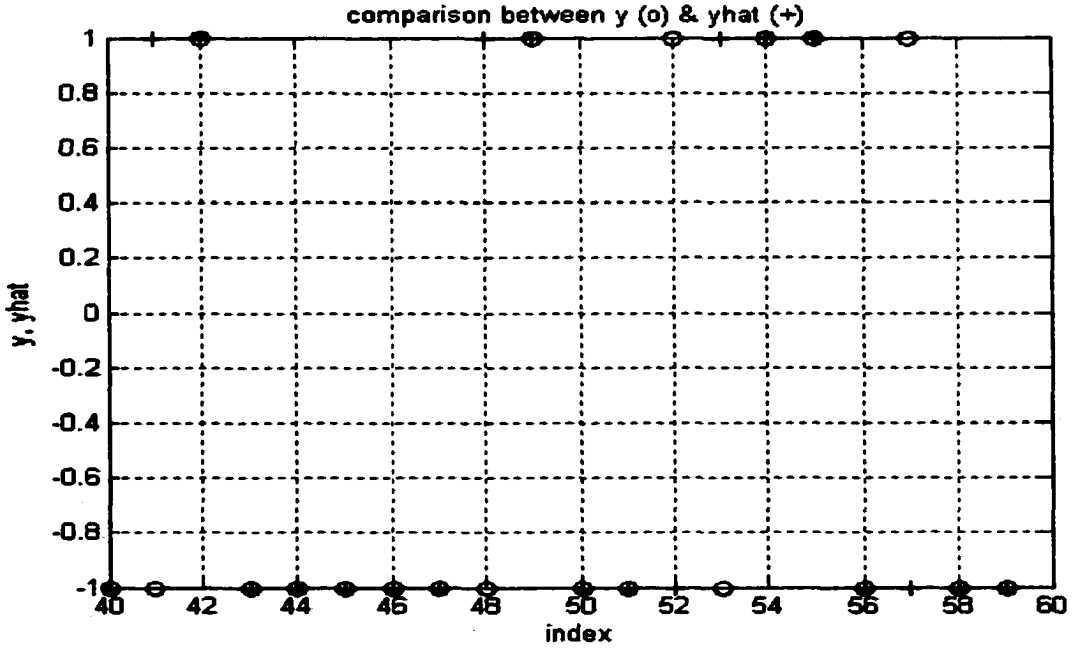


Figure26- Misclassified testing points with uncertainty eita = 0.07 for echo cardio data. There are 5 points misclassified. They are the empty circles.

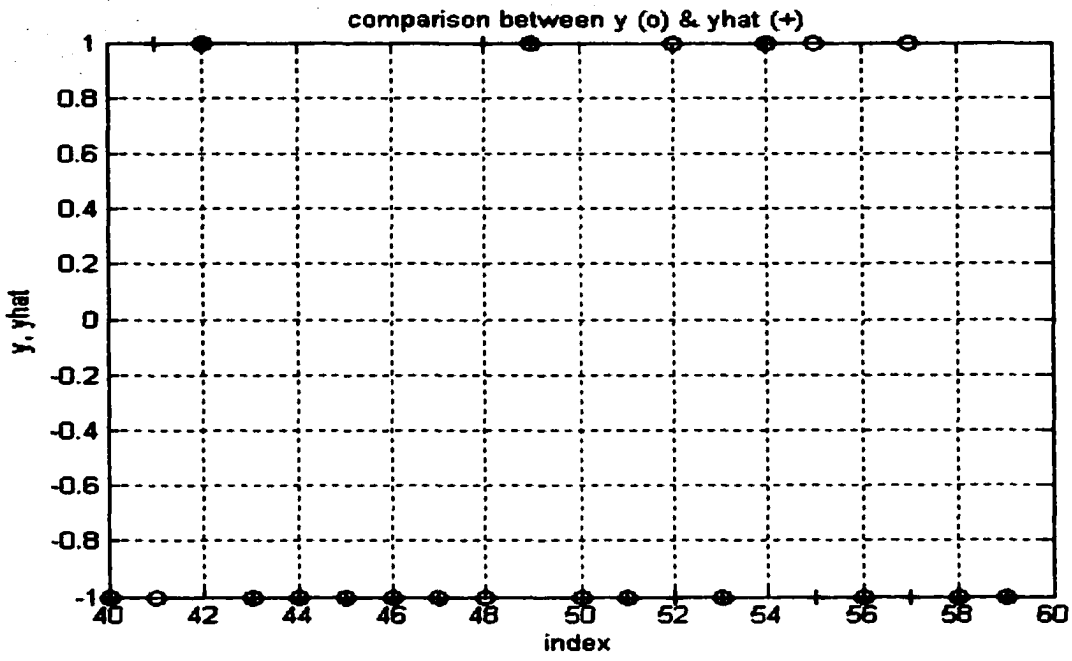


Figure27- Misclassified <u>testing</u> points with uncertainty eita = 0.09 for echo cardio data. There are 5 points misclassified. They are the empty circles.

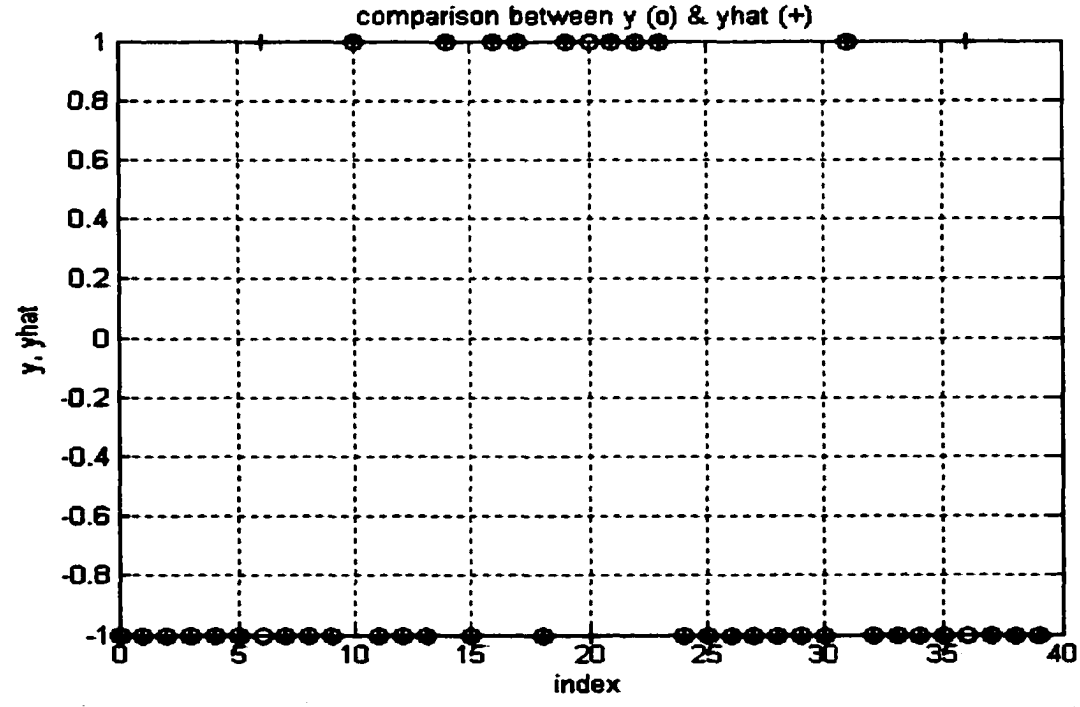


Figure28- Misclassified <u>training</u> points with uncertainty eita = 0.09 for echo cardio data. There are 3 points misclassified. They are the empty circles.

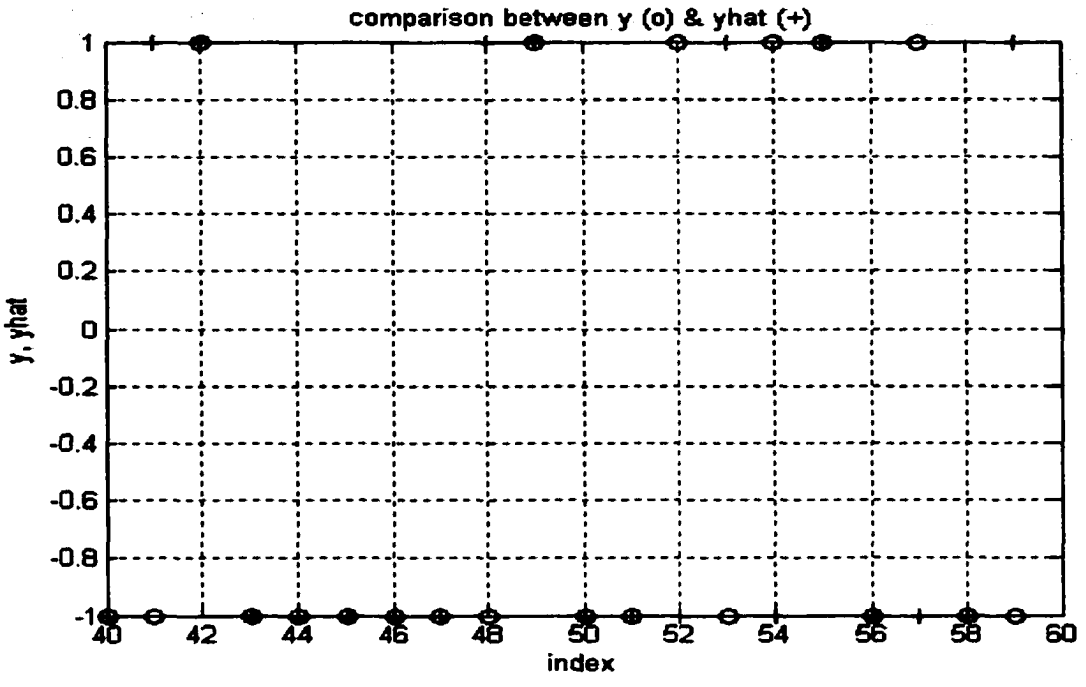


Figure29- Misclassified testing points with uncertainty eita = 0.5 for echo cardio data. There are 7 points misclassified. They are the empty circles.

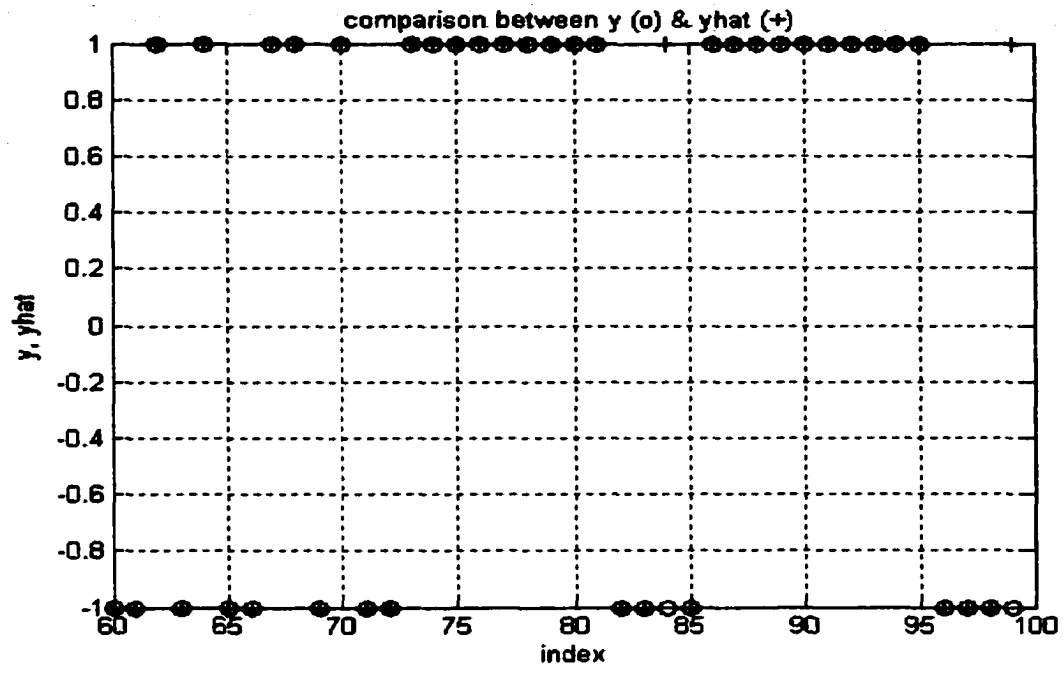


Figure30-Misclassified <u>testing</u> points with uncertainty eita = 0 for the breast cancer data. There are 2 misclassified points. They are the empty circles.

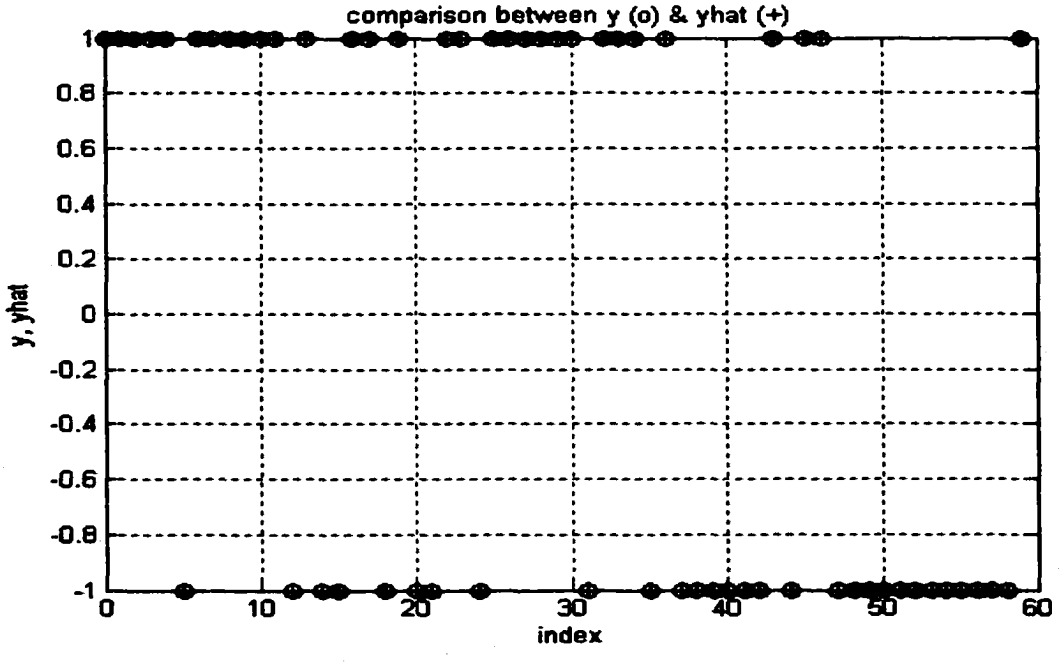


Figure31-Misclassified <u>training</u> points with uncertainty eita = 0 for the breast cancer data. There are zero misclassified points.

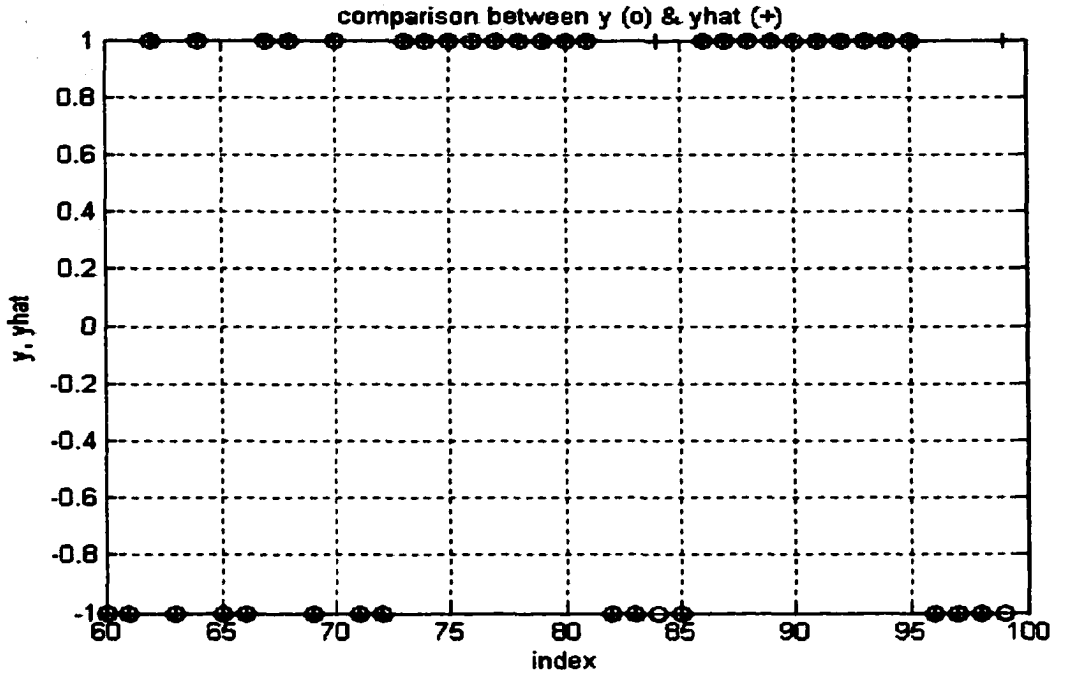


Figure32-Misclassified <u>testing</u> points with uncertainty eita = 0.01 for the breast cancer data. There are 2 misclassified points. They are the empty circles.

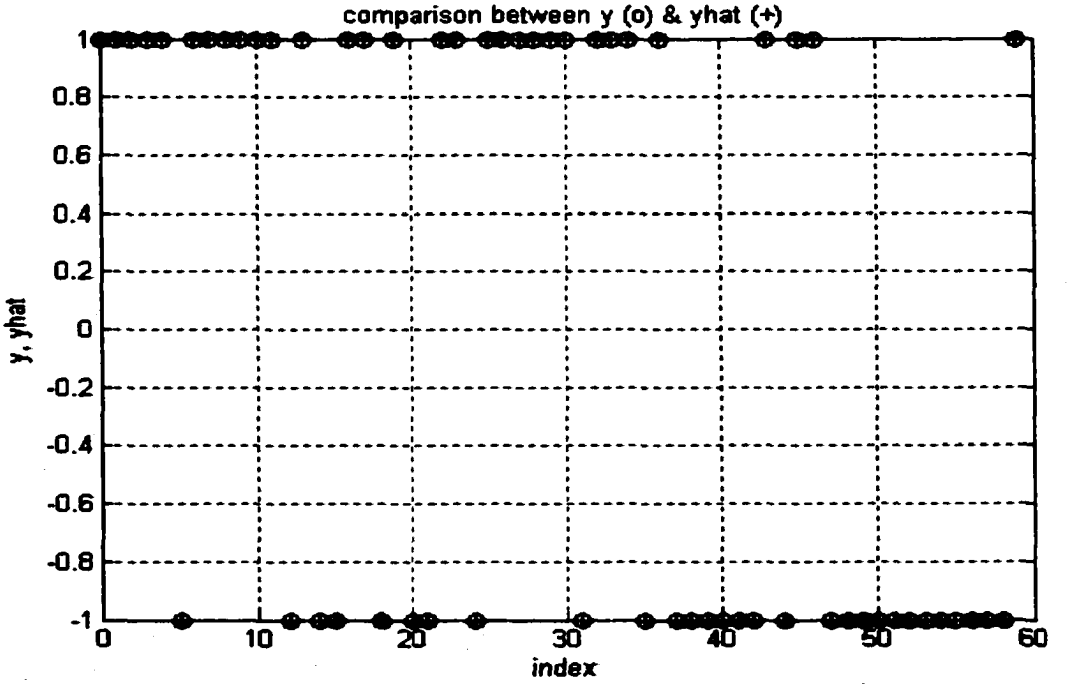


Figure33-Misclassified <u>training</u> points with uncertainty eita = 0.01 for the breast cancer data. There are zero misclassified points.

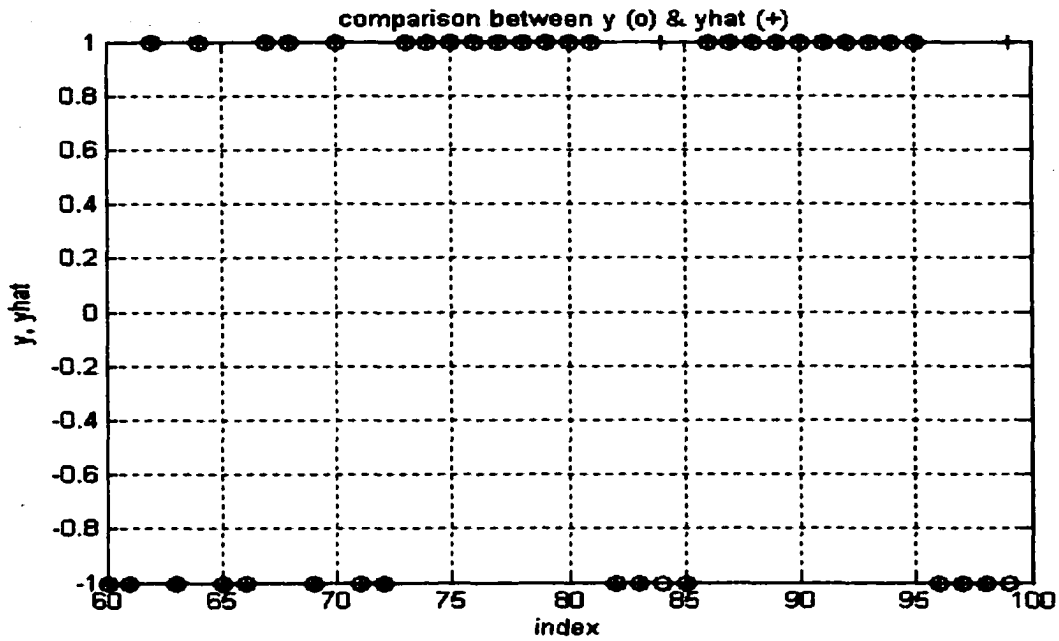


Figure34-Misclassified <u>testing</u> points with uncertainty eita = 0.05 for the breast cancer data. There are 2 misclassified points. They are the empty circles.

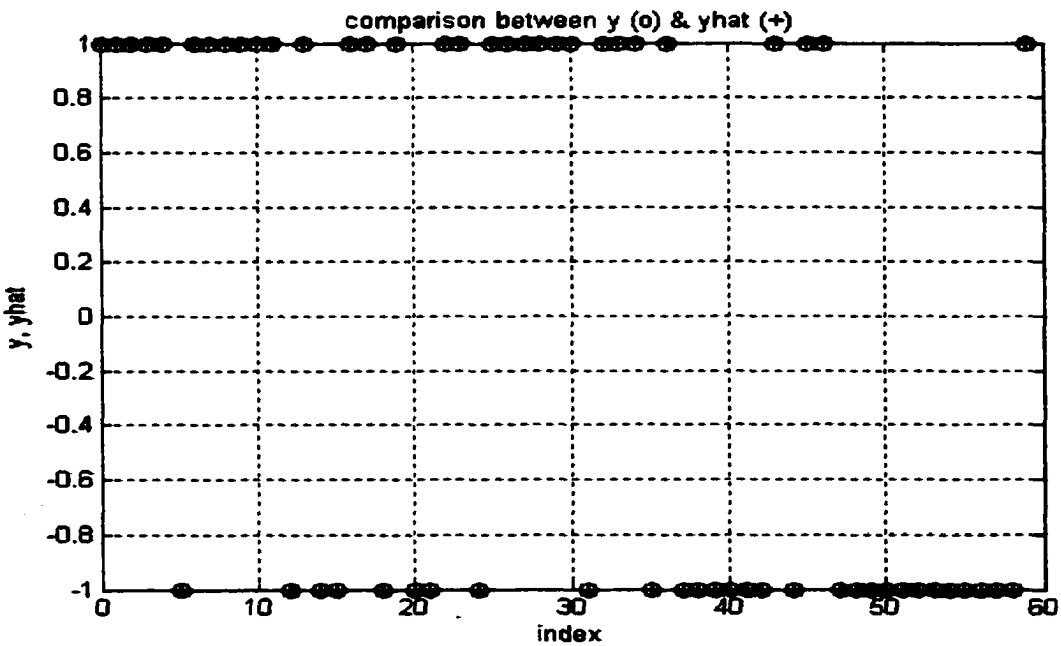


Figure35-Misclassified <u>training</u> points with uncertainty eita = 0.05 for the breast cancer data. There are zero misclassified points.

Appendix F

Figures for Regression

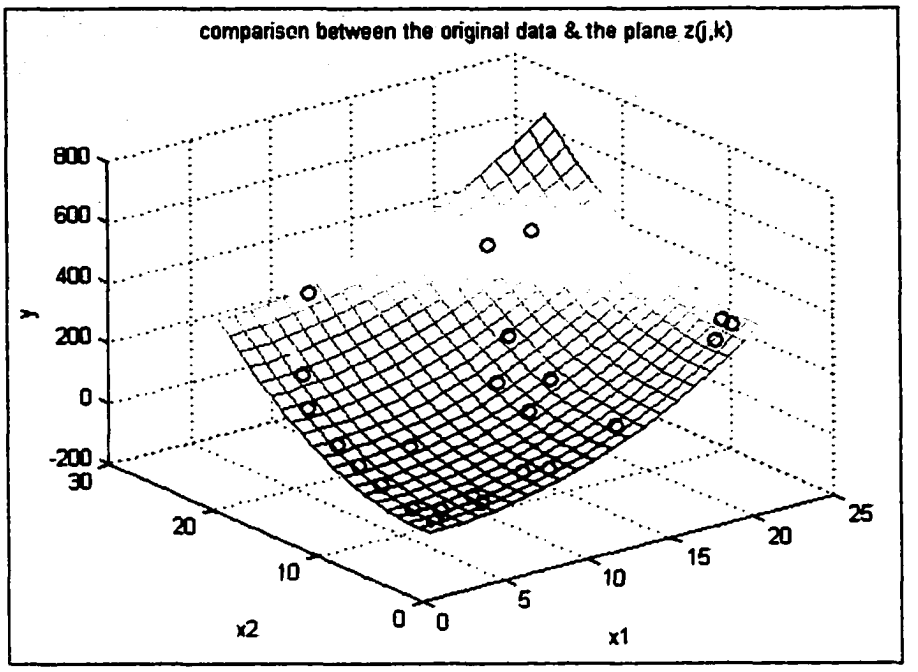


Figure1a- Synthetic Problem for Regression with Uncertainty (eita) = 0. Original Data

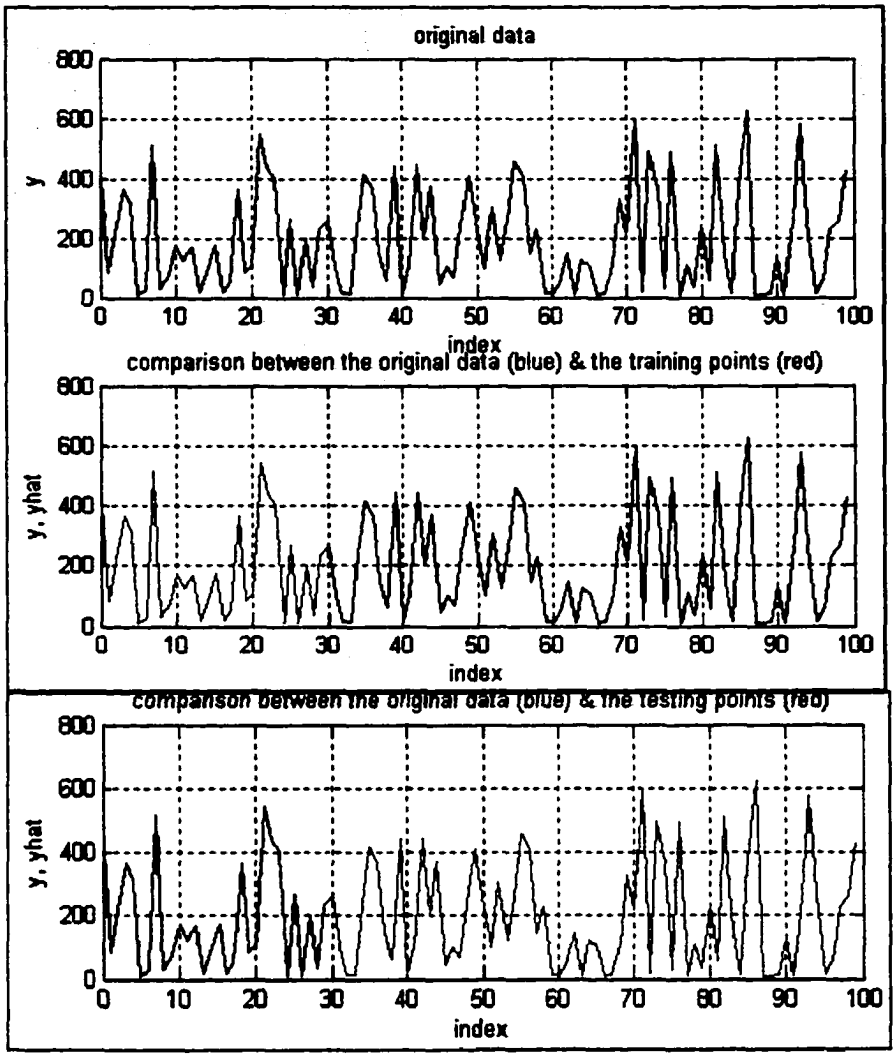


Figure1b- Synthetic problem for Regression with Uncertainty eita) = 0. Original Data.

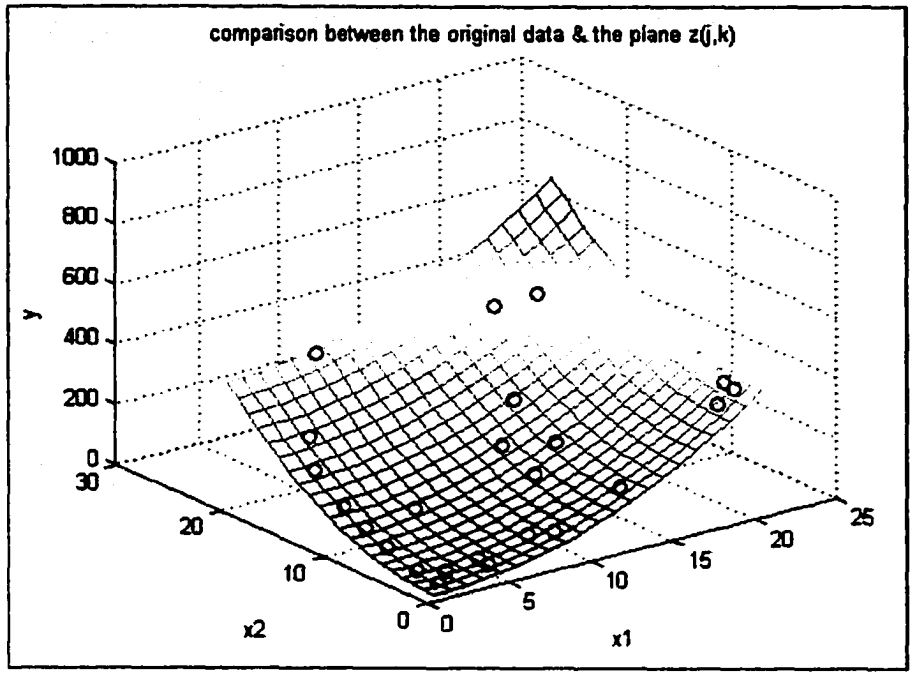


Figure2a- Synthetic Problem for Regression with Uncertainty (eita) = 0.000001. Original Data.

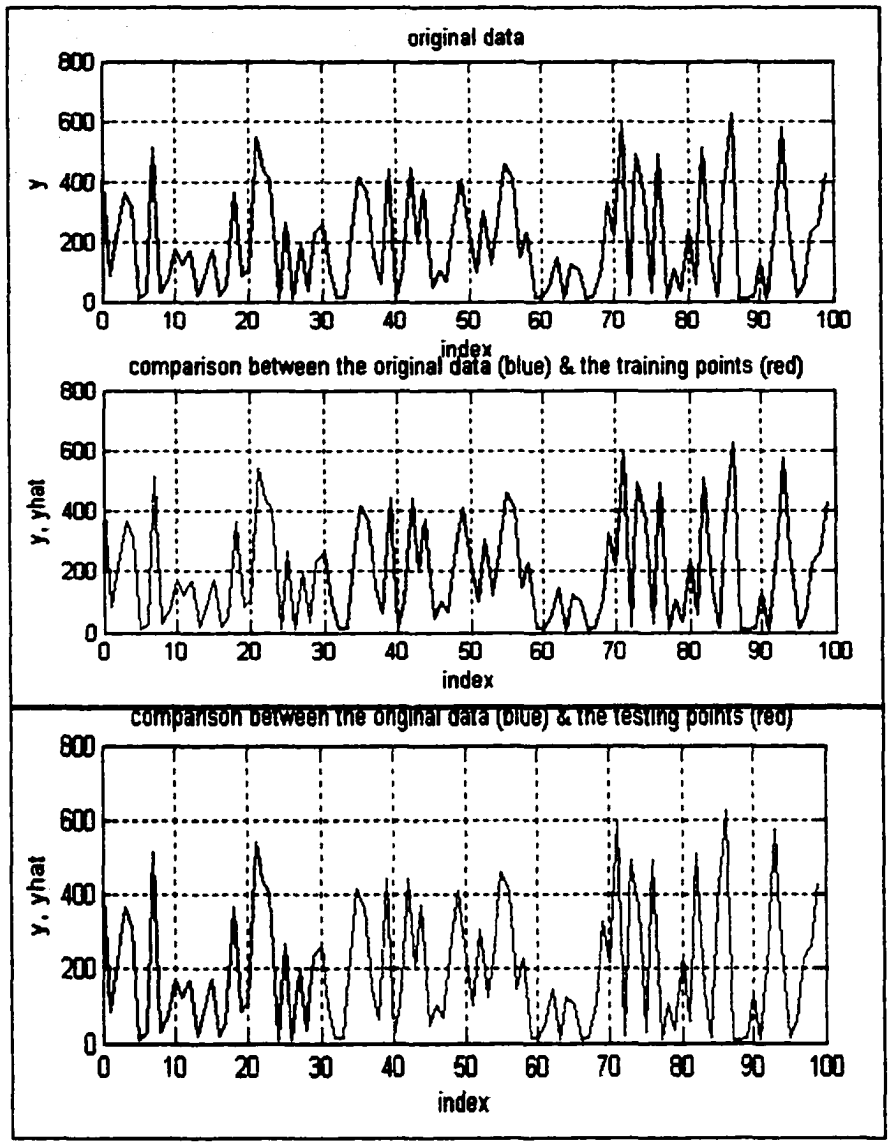


Figure2b- Synthetic problem for Regression

with Uncertainty eita) = 0.000001.Original Data.

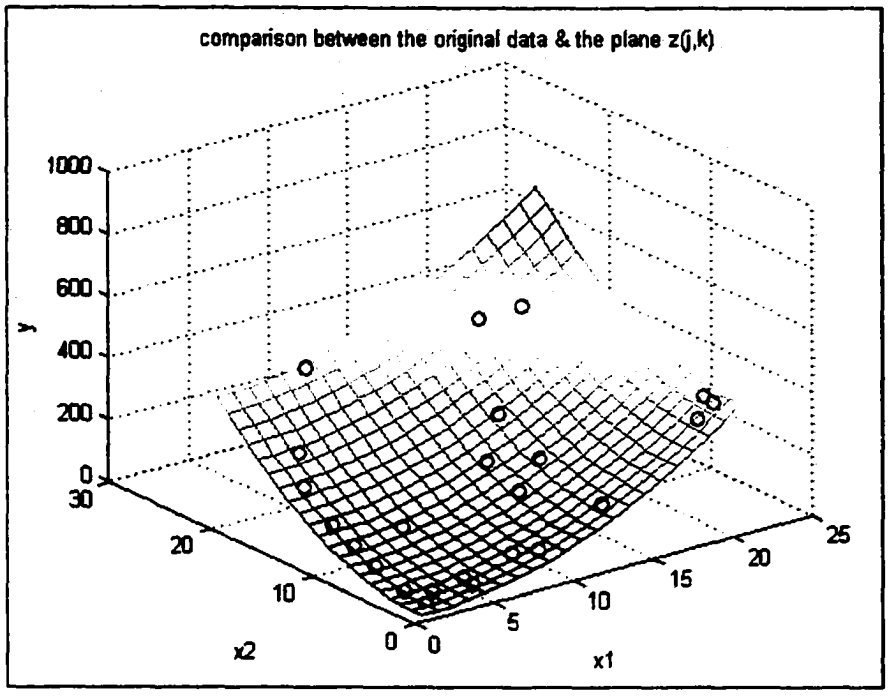


Figure3a- Synthetic Problem for Regression with Uncertainty (eita) = 0.00001. Original Data.

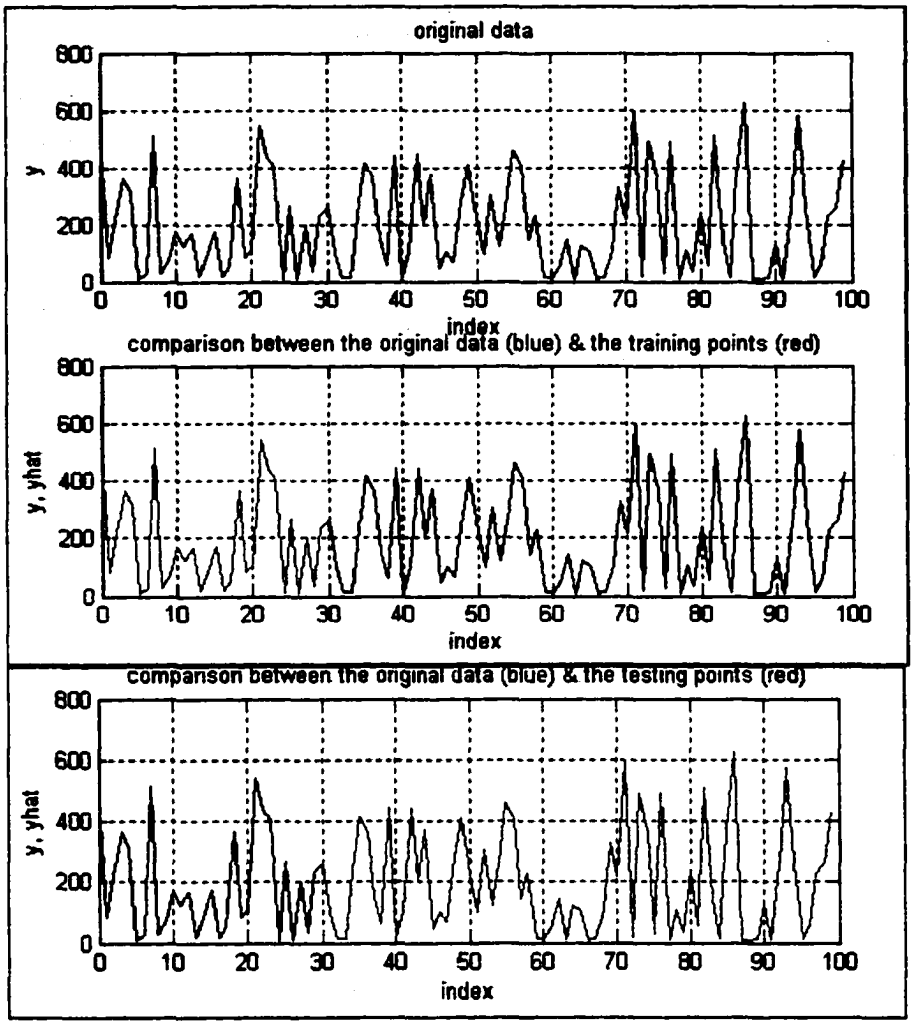


Figure3b- Synthetic problem for Regression

with Uncertainty eita) = 0.00001.Original Data.

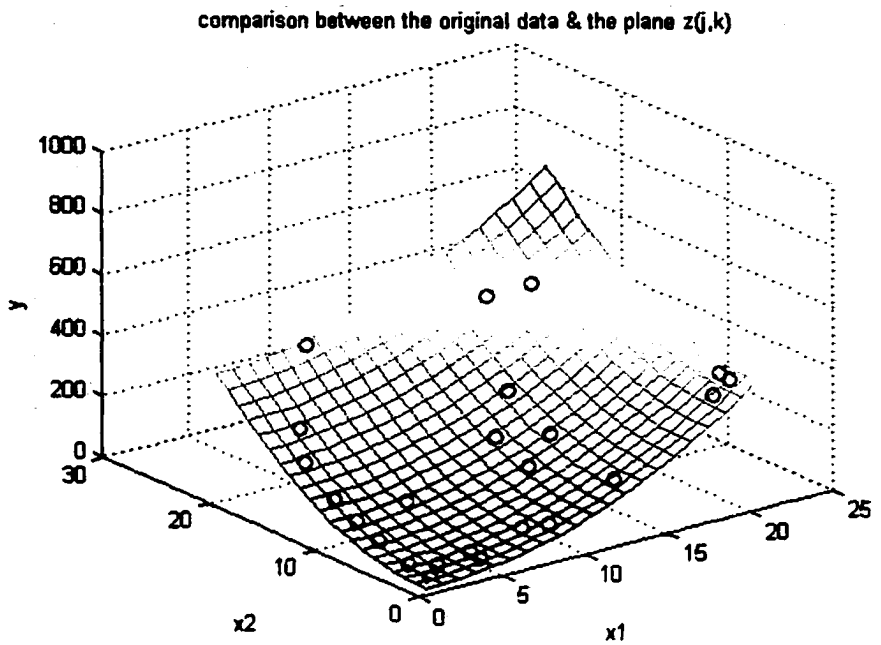


Figure4a- Synthetic Problem for Regression with Uncertainty (eita) = 0.0001. Original Data.

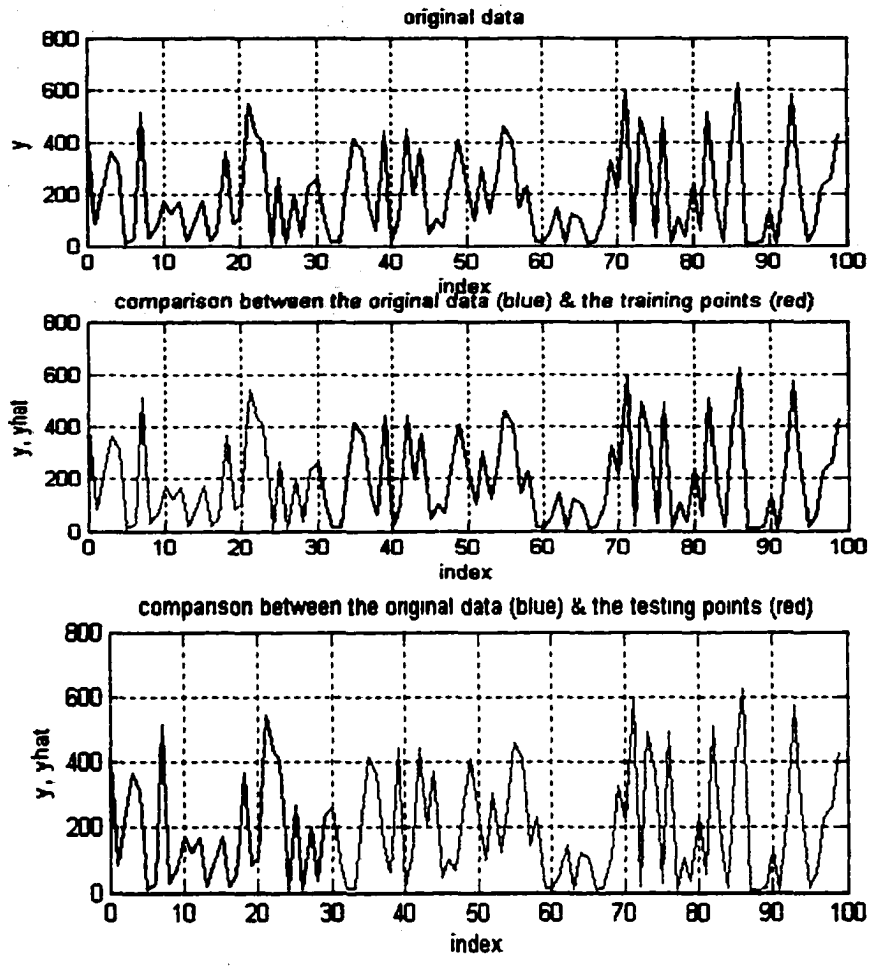


Figure4b- Synthetic problem for Regression

with Uncertainty eita) = 0.0001.Original Data.

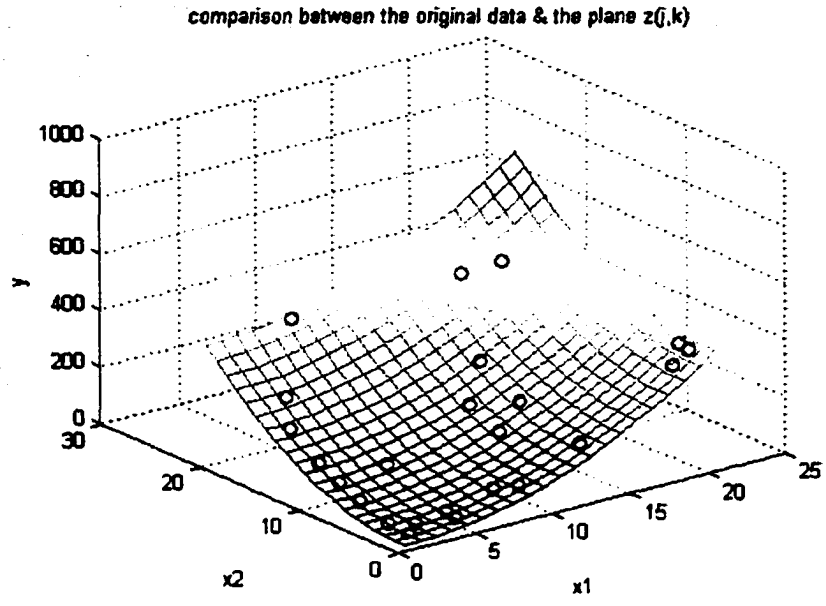


Figure5a- Synthetic Problem for Regression with Uncertainty (eita) = 0.1. Original Data.

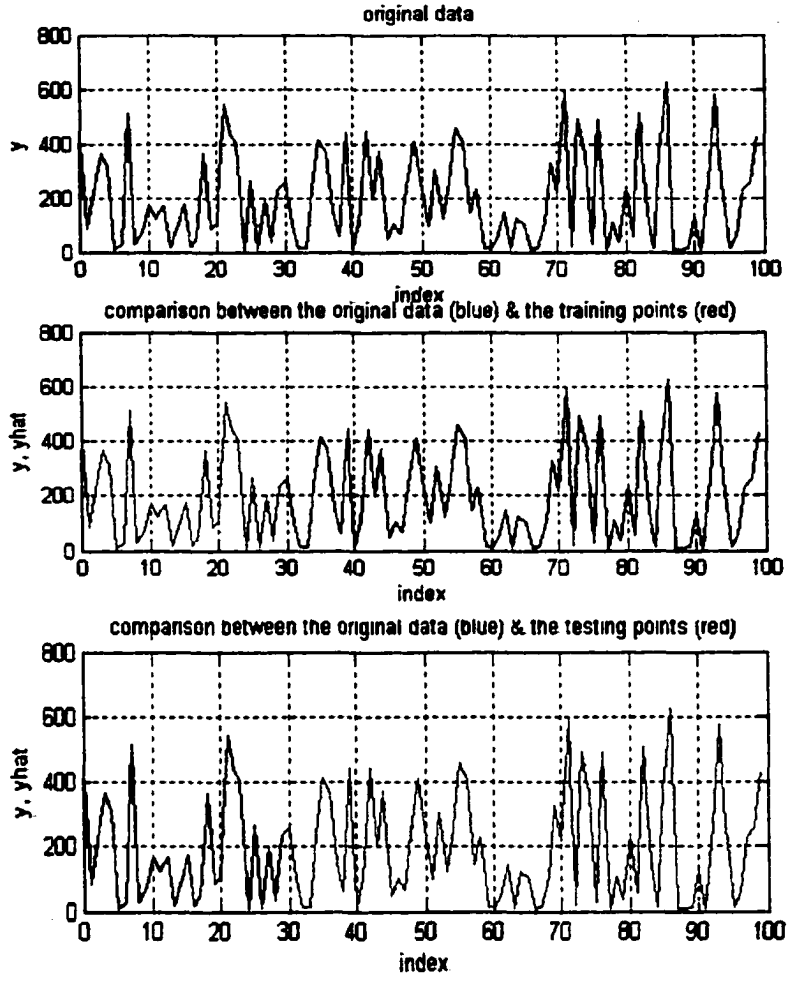


Figure5b- Synthetic problem for Regression

with Uncertainty eita) = 0.1.Original Data.

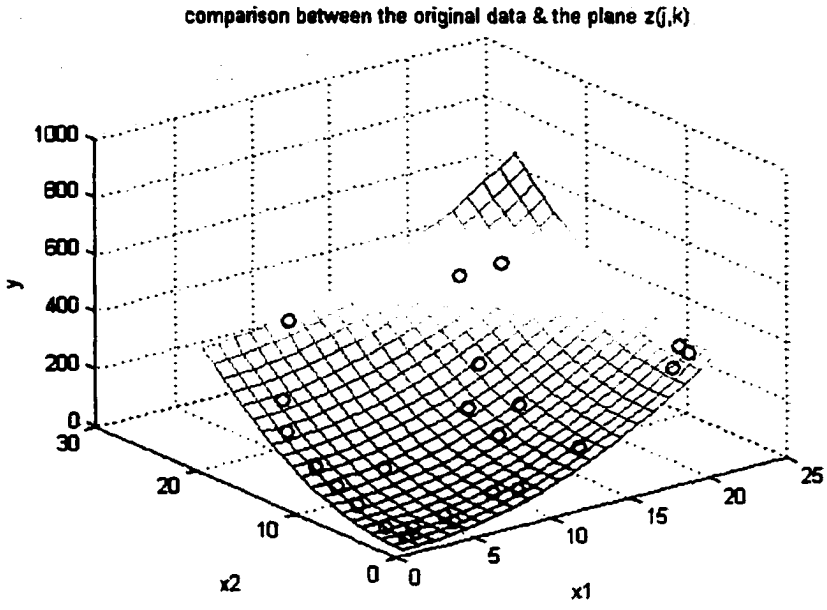


Figure6a- Synthetic Problem for Regression with Uncertainty (eita) = 0.5. Original Data.

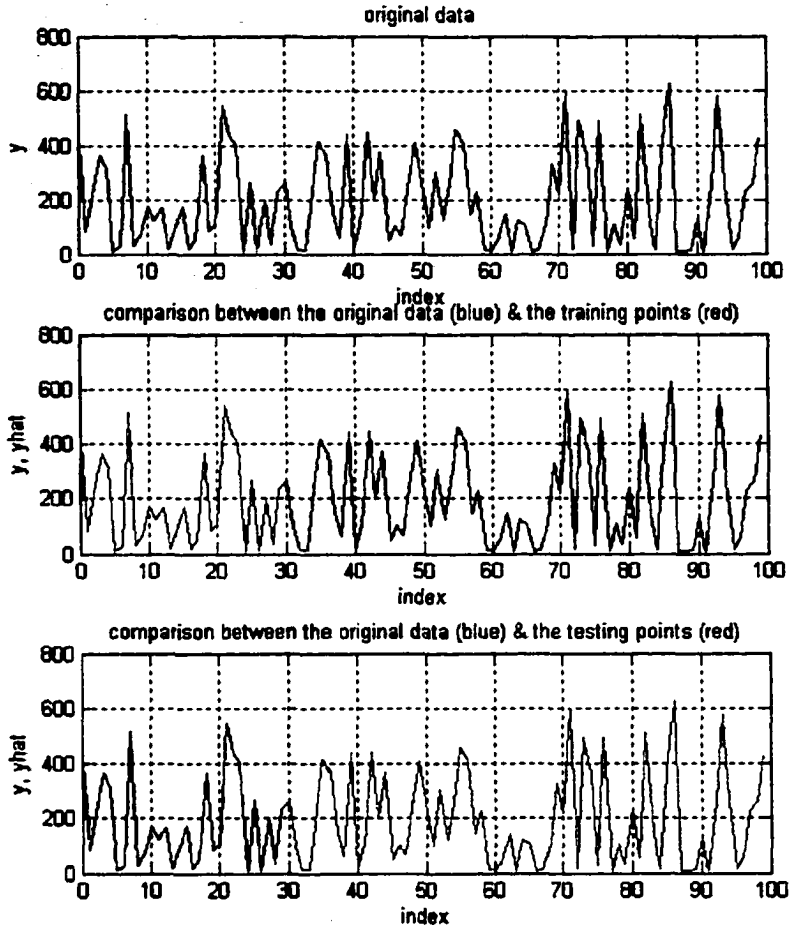


Figure6b- Synthetic problem for Regression

with Uncertainty eita) = 0.5.Original Data.

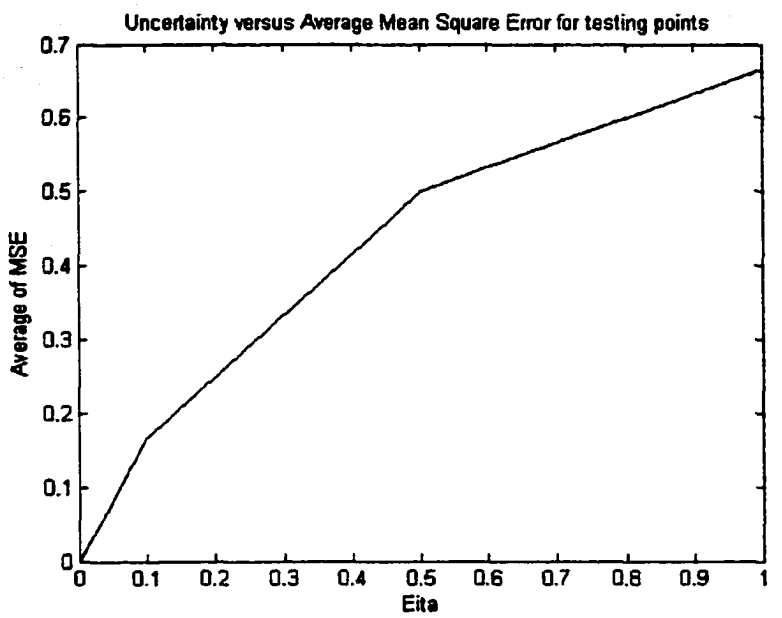


Figure7- The Uncertainty against the average mean square error of Testing points for the synthetic regression problem.

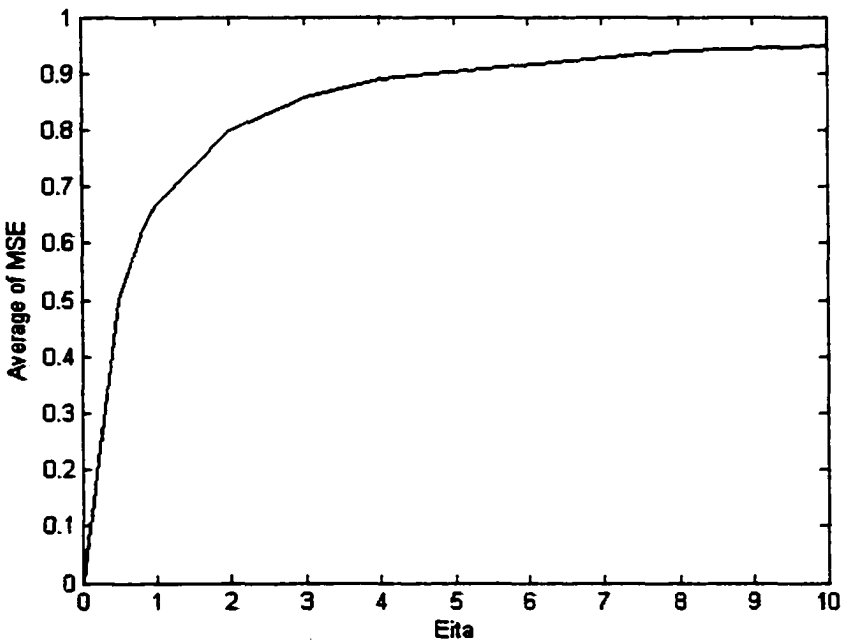


Figure8- Similar to figure7 above but with larger eita. It shows the Behavior of uncertainty and mean square error for the synthetic Regression problem.

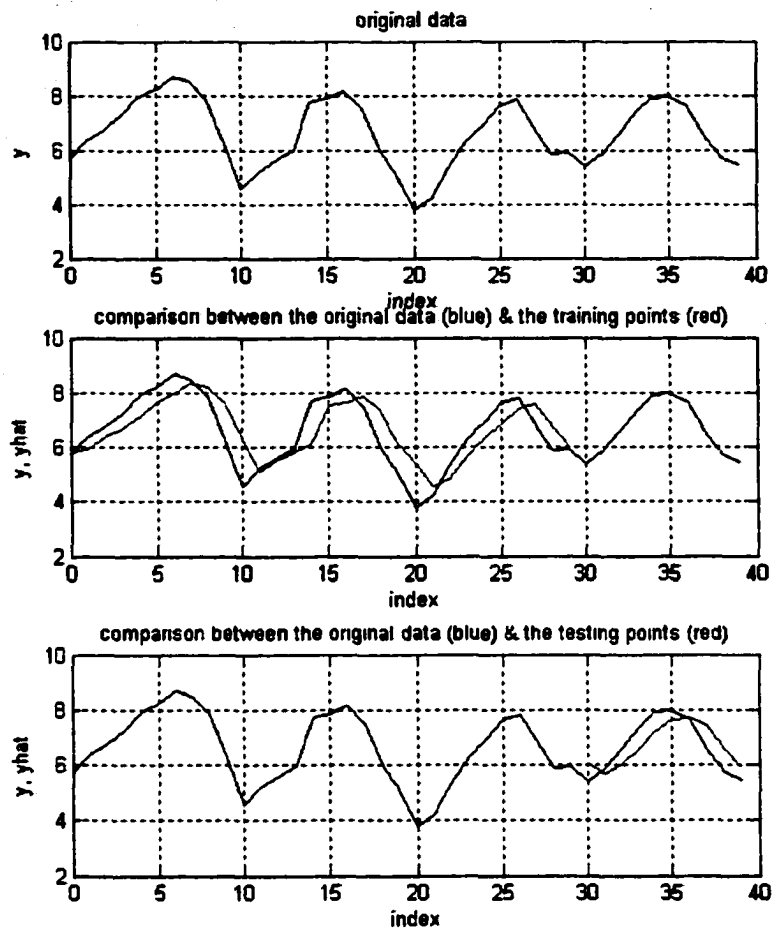


Figure9- Comparison Among Training and Testing points with the Original Data Points for the Lynx Regression Application with uncertainty equals to 0➔ Precise Data.

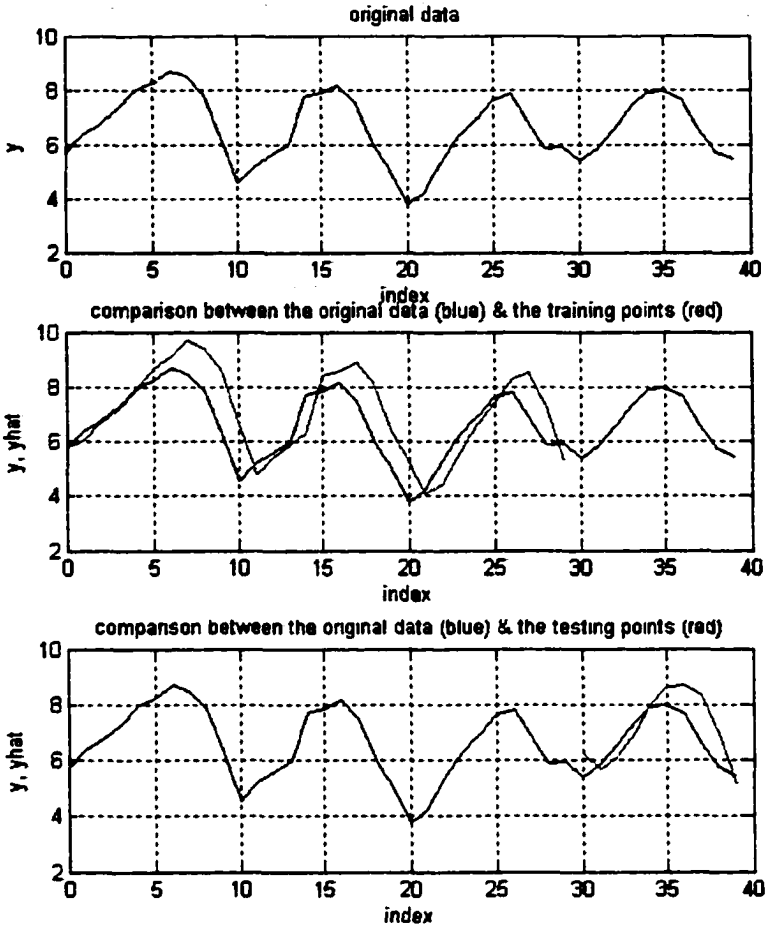


Figure10- Comparison Among Training and Testing points with the Original Data Points for the Lynx Regression Application with uncertainty equals to 0.0001.

References

[1] Ben-Tal A. and Nemirovski A., Robust Solutions to Uncertain Linear Programs Via Convex Programming, *Operations Research Letters*, Vol.25, No.1,1-17, 1996.

[2] Ben-Tal A. and Nemirovski A., Robust Convex Optimization, *Mathematics of Operations Research*, Vol.23, 4, 769-805, 1998.

[3] Ben-Tal A. and Nemirovski A., Robust, Truss Topology Design Via Semidefinite Programming, *SIAM J. of Optimization*, Vol. 7, No.4, 991-1016, 1997.

[4] Ben-Tal A., Tamar M., and Nemirovski A., Robust Modeling of Multi-Stage Portfolio Problems, *Applied Optimization*, Vol. 33, 303-328, 2000.

[5] Ben-Tal A. and Nemirovski A., Robust Convex Optimization. Technical report, faculty of Industrial Engineering and Management, Technion, December 1996.

[6] Beale E.M.L., Minimizing a Convex Function Subject to Linear Inequalities, *J. Royal Stat.Soc*, Vol.17, 173 –184, 1955.

[7] Bennett K.P. and Mangasarian O.L., Robust Linear Programming of Two Inseparable Sets, *Optimization Methods and Software* 1, 23 – 34, 1992.

[8] Bennett K.P. and Campbell C., Support Vector Machines: Hype or Hallelujah? *SIGKDD Explorations*, 2(2):1-6, 2000.

[8] Burges C.J.C., A Tutorial on Support Vector Machines for Pattern Classification, *Data mining and Knowledge Discovery*, 2(2): 121-167, 1998.

[9] Birge J.R., The Value of the Statistic Solution in Stochastic Linear Programs with Fixed Resources, *Math. Programming*, Vol.24, 314-325, 1982.

[10] Boyd S., Lobo M.S., and Vandenberghe L., Applications of Second-Order Cone Programming, *Linear Algebra and its Applications*, Vol.284, 193-226, 1998.

[11] Cristianini N. and Taylor J.S., *An Introduction to Support Vector machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, United Kingdom, 1999.

[12] Collobert R. and Bengio S., SVmtorch: Support Vector Machines for Large-scale Regression Problems, *Machine Learning Research*, 1:143-160, 2001.

[13] Chinneck J.W. and Ramadan K., Linear Programming with Interval Coefficients, *Journal of the Operation Research Society*, Vol.51, 209-220, 2000.

[14] Cherkassky V. and Mulier F., *Learning From Data: Concepts, Theory and Methods*, Wiley, 1998

[15] Dantzig G.B., Linear Programming Under Uncertainty, *Management Science*, Vol.1, 3 and 4,197-206, 1955.

[16] Dantzig G.B. and Infanger G., Multi-stage Stochastic Linear Programs for Portfolio Optimization, *Annals of Operations Research*, Vol.45, 59-76, 1993.

[17] El Ghaoui L. and Lebret H., Robust Solutions to Least- Squares Problems with Uncertain Data, *SIAM j. Optimization* Vol.18, 4, 1035-1064, 1997.

[18] Gunn S., Support Vector Machines for Classification and Regression, ISIS Technical Report, 1997.

[19] Grötschel, Lovasz L., and Schrijver A., *The Ellipsoid Method and Combinatorial Optimization*, Springer, Heidelberg, 1988.

[20] Kwon Hee Lee, In-Sup Eom, Gyung-Jin Park, and Wan-Ik Lee., Robust Design for Unconstrained Optimization Problems Using the Taguchi Method, *AIAA Journal*, Vol.34, No.5, 1059-1063, 1996.

[21] Kouvelis P. and Gang Yu., *Robust Discrete Optimization and Its Applications* Kluwer Academic Publishers, Boston, 1997.

[22] Mulvey J.M., Vanderbei R.J., and Zenios S.A., Robust Optimization of Large-Scale Systems, *Operations Research*, Vol.43, 2, 264-281,1995.

[23] Mangasarian O.L; Minimum- Support Solutions of Polyhedral Concave Programs, *Optimization*, 45, 149-162, 1999.

[24] Mangasarian O.L, and Musicant D.R., Robust Linear and Support Vector Regression, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22,950-955, 2000

[25] Murata N. and Pedroso J.P., Support Vector Machines for Linear Programming: Motivation and Formulations, Riken Brain Science Institute, BSIS Technical Report No.99-XXX, 1999.

[26] Oustry F., L.El Ghaoui, and H.Lebret, Robust Solutions to Uncertain Semidefinite Programs, *SIAM J. on Optimization*, 1998.

[27] Pontil M., Mukherjee. S., and Girosi F., On the Noise Model of Support Vector Machines Regression, Technical Report, Center for Biological and Computational Learning, MIT, 1999.

[28] Quan Zheng., Robust Analysis and Global Optimization, *Annals of Operations Research*, Vol.24, 1-4,273-286, 1990.

[29] Quan Zheng., Robust Analysis and Global optimization, *Computer and Mathematics With Application*, Vol.21, 6-7, 1991.

[30] Reklaitis G.V. Ravindran, A., and Ragsdell K.M., *Engineering Optimization Methods and Applications*, John Wiley and Sons, New York, 1983.

[31] Tsui K. –L., Robust Design Optimization For Multiple Characteristics Problems, *International Journal of Production Research*, Vol.37,2, 433-445, 1999.

[32] Tsutomu M., A Circumscribed Ellipsoid Method For Multi-Objective Programming And Applications To Robust Optimization, Ph.D. Dissertation, School of Industrial Engineering, The University of Oklahoma, 1996.

[33] Trafalis T., Robust Optimization In Support Vector Machines Learning, Technical Report, Optimization and Intelligent Systems Laboratory, School of Industrial Engineering, University of Oklahoma, 1999.

[34] UCI Machine Learning Data base ftp. Ics. Uci. Edu /pub/machine-learning databases/ echocardiogram.

[35] UCI Machine learning Database ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-

[36] Problem Source: Time Series Data Library at Monash University. http://www.maths.monash.edu.au/~hyndman/tseries/ecology.html.

[37] Vapnik V., *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.

[38] Vapnik V., *Statistical Learning Theory*, Wiley, 1998.