UNIVERSITY OF SOUTHAMPTON

Classification under Input Uncertainty with Support Vector Machines

by

Jianqiang Yang

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

July 31, 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Classification under Input Uncertainty with Support Vector Machines

by Jianqiang Yang

Uncertainty can exist in any measurement of data describing the real world. Many machine learning approaches attempt to model any uncertainty in the form of additive noise on the target, which can be effective for simple models. However, for more complex models, and where a richer description of anisotropic uncertainty is available, these approaches can suffer. The principal focus of this thesis is the development of advanced classification approaches that can incorporate the known input uncertainties into support vector machines (SVMs), which can accommodate isotropic uncertain information in the classification. This new method is termed as uncertainty support vector classification (USVC). Kernel functions can be used as well through the derivation of a novel kernelisation formulation to generalise this proposed technique to non-linear models and the resulting optimisation problem is a second order cone program (SOCP) with a unique solution. Based on the statistical models on the input uncertainty, Bi and Zhang (2005) developed total support vector classification (TSVC), which has a similar geometric interpretation and optimisation formulation to USVC, but chooses much lower probabilities that the corresponding original inputs are going to be correctly classified by the optimal solution than USVC. Adaptive uncertainty support vector classification (AUSVC) is then developed based on the combination of TSVC and USVC, in which the probabilities of the original inputs being correctly classified are adaptively adjusted in accordance with the corresponding uncertain inputs. Inheriting the advantages from AUSVC and the minimax probability machine (MPM), minimax probability support vector classification (MPSVC) is developed to maximise the probabilities of the original inputs being correctly classified. Statistical tests are used to evaluate the experimental results of different approaches. Experiments illustrate that AUSVC and MPSVC are suitable for classifying the observed uncertain inputs and recovering the true target function respectively since the contamination is normally unknown for the learner.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

$\boldsymbol{x}$	input vector
y	label of input vector
f	probability density function
$\mathcal{D}$	data set
g	target function
η	noise rate (probability) of inputs being contaminated
$EX(g, \mathcal{D}, \eta)$	noise model
$\boldsymbol{x}^{'}$	corrupted input vector
$\boldsymbol{w}$	weight vector
b	offset bias
g_v	function vertical to target function
Pr(.)	probability
$\phi(.)$	mapping function
$K(.)$	kernel function
$\mathbf{K}(.)$	covariance matrix
$k(.)$	covariance function
Cov	covariance
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean vector μ and covariance matrix Σ
$\boldsymbol{x}_k$	known features of $\boldsymbol{x}$
$\boldsymbol{x}_m$	missing features of $\boldsymbol{x}$
θ	parameter of conditional distribution
$\boldsymbol{x}_{io}$	original counterpart of $\boldsymbol{x}_i$
$\boldsymbol{x}_i$	corrupted input vector
$\boldsymbol{z}_i$	uncertain input vector
$\mathbf{M}_i$	covariance matrix of uncertain input
y_i	label scalar of uncertain input in training data sets
ρ	margin
$\mathcal{E}(\mathbf{A}, \boldsymbol{a})$	ellipsoid with centre $\boldsymbol{a}$ and covariance matrix $\mathbf{A}$
$\mathcal{S}(0, 1)$	unit ball
α	lower bound of probability of inputs being correctly classified
$\Phi(.)$	cumulative distribution function
ξ_i	penalty parameter

C	regularisation parameter
α_i	variable scalar in dual optimisation problem
$\boldsymbol{\beta}_i$	variable vector in dual optimisation problem
l	number of uncertain inputs
n	dimension of input space
m	dimension of feature space
$\mathbf{J}$	Jacobian matrix
$\mathbf{I}$	identity matrix
δ_i	uncertainty bound
$\triangle \boldsymbol{x}_i$	noise following a certain distribution
$\mathbb{V}$	set of support vectors
$O(.)$	order
r	probability confidence
r_i	individual probability confidence
d	minimal distance from uncertainty to boundary
$\kappa(\alpha)$	monotonically increasing function of probability α
erf	Gauss error function
p_i	minimal distance from the mean of uncertainty to its misclassified part
q_i	distance between the mean of uncertainty and the nearest edge of its distribution to the optimal solution
D_i	lower bound of individual probability confidence r_i
L_i	regularisation parameter
p	p-value
$\bar{r}_{.j}$	average rank for each algorithm
$\bar{r}$	average rank for all algorithms
N	number of data sets
k	number of algorithms compared
CD	critical difference
q_α	critical value
f_s	space factor
f_d	dimension factor
f_c	contamination factor
τ	parameter for percentage of contaminated inputs to all
ν	severe or light contamination controller
ζ	parameter for contamination according to the distribution of inputs or the target function

DECLARATION OF AUTHORSHIP

I, ... , [please print name]

declare that the thesis entitled [enter title]

. . . . . .

. . . . . .

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- none of this work has been published before submission, **or** [delete as appropriate] parts of this work have been published as: [please list references]

  Yang, J., Gunn, S.R.: Uncertain Input Classification in Support Vector Machines. Sheffield Machine Learning Workshop. (2004)

  Yang, J., Gunn, S.R.: Exploiting Uncertain Data in Support Vector Classification. KES2007 11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. (2007)

  Yang, J., Gunn, S.R.: Iterative Constraints in Support Vector Classification with Uncertain Information. International Workshop on Constraint-Based Mining and Learning, at ECML/PKDD 2007. (2007)

Signed:. . . . . .

Date:. . . . . .

Acknowledgements

Sincerely, I am deeply indebted to my supervisor Professor Steve Gunn, whose wise guidance and encouragement helped me in all the time of research for and writing of this thesis, especially after I moved to London, he always made some time to discuss the difficulties I met in my research every time he travelled to London for business. Also his creativity and enthusiasm enlightened me many times on the journey of Ph.D.

Besides, I would like to express my appreciation and gratitude to Professor Bob Damper and Dr Gavin Cawley, whose suggestions made me improve my work from different points of view.

Finally, a special gratefulness must be sent to many nice colleagues at ISIS which is such a cheerful working place.

Especially, I would like to give my special thanks to my beloved wife, Nan, whose patient love and continuously moral support enabled me to complete this work. This thesis is dedicated to my parents.

Chapter 1

Introduction

How can we forecast future events? What can we explain based on existing events? The answer is learning, through which future events can be forecasted by analysing existing events. Learning is

> "Changes in a system that enable it to do the same task or tasks drawn from the same population more efficiently and more effectively the next time."

defined by Simon (1983). A human can adapt and learn from their experience by using his/her biological systems. With the growth of algorithms and the development of computer technology, our cognitive abilities have been expanded to a new level. Computer systems can develop solutions for particular complex learning tasks that are too difficult or impossible to construct manually, can develop systems that automatically customise themselves to the needs of individual users through experience, and can develop database mining systems by discovering knowledge and patterns in databases. Machine learning is

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T as measured by P, improves with experience E."

defined by Mitchell (1997). Machine learning methods develop models by extracting principles, rules and patterns out of data sets. Generally, machine learning tasks can be divided into supervised, unsupervised, semi-supervised and reinforcement learning. The task of supervised learning is to provide an algorithm that the supervised learner uses to generate a function which maps inputs to desired outputs. The methods developed in this thesis are all categorised as supervised learning.

Theoretically, machine learning algorithms and their performance can be analysed in a related mathematical field, computational learning theory, which attempts to explain

the learning process from a statistical point of view. Normally, since training sets are finite and the future is uncertain, learning theory usually yield probabilistic bounds on the performance rather than absolute guarantees of the performance of the algorithms. On the other hand, different approaches to computational learning theory use varied inference principles and definitions of probability, finally leading to different machine learning algorithms. The different approaches include the probably approximately correct learning (PAC learning) proposed by Valiant (1984), Vapnik-Chervonenkis theory (VC theory) proposed by Vapnik and Chervonenkis (1971), Bayesian inference (Winkler, 2003) and the algorithmic learning theory proposed by Gold (1967).

VC theory is related to statistical learning theory and can be applied to empirical processes. VC theory covers at least four parts (Vapnik, 1999), including theory of consistency of learning processes, non-asymptotic theory of the rate of convergence of learning processes, theory of controlling the generalisation ability of learning processes and theory of constructing learning machines. VC theory contains important concepts such as the VC dimension and structural risk minimisation (SRM), from which, a well known learning algorithm, the support vector machine (SVM) was developed by Boser et al. (1992). As a maximum margin method, SVM maximises the margin between two data sets to minimise an upper bound on the VC dimension, which holds an upper bound for the generalisation error. In general, the larger the margin the lower the generalisation error. SVM receives much attention and becomes an active field of machine learning research. Applications include isolated handwritten digit recognition (Cortes and Vapnik, 1995; Gorgevik and Cakmakov, 2002), object recognition (Blanz et al., 1996; Boughorbel et al., 2004), speaker identification (Schmidt, 1996; Wan and Renals, 2003), face detection in images (Osuna et al., 1997; Huang et al., 2002), etc.

1.1 Problem Statement

Uncertainty is the lack of certainty, a state of having limited knowledge where it is impossible to exactly describe existing state or future outcome, more than one possible outcome (Hubbard, 2007). In machine learning, there exist different types of uncertainties, including uncertainty about variability, structural uncertainty, input uncertainty and target uncertainty. Variability is described by frequency distributions, whereas uncertainty is described by probability distributions (Morgan and Henrion, 1990). The uncertainty about variability is generated by estimating the unknown distribution of inputs in the population being studied. This uncertainty can be handled in classification problems by the minimax probability machine (MPM) proposed afterwards. The connections of the other uncertainties can be illustrated by Figure 1.1, where g represents the structure of a model that defines how characteristics y are determined from inputs x and $y = g(x)$, $\mathcal{D}_x$ denotes the plausibility region for model inputs and $\mathcal{D}_y$ denotes the uncertainty region for model targets. Structure uncertainty comes from model

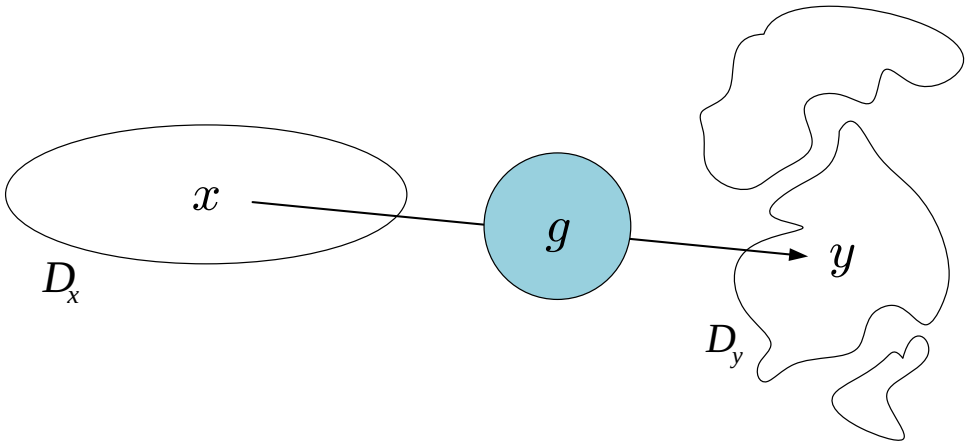


FIGURE 1.1: General description for input and target uncertainty.

structures, considers both model and parameter uncertainty, where the output is determined by a variety of statistical models consisting of a family of distributions indexed by the parameters. The Bayesian approach provides a natural and general probabilistic framework that accounts for model uncertainty as well as parameter uncertainty (Clyde and George, 2004). Input uncertainty comes from any uncertain input x defined by a probability distribution function and exists independently of the model. The probability function $f(x)$ describes input uncertainty and $f(y)$ describes target or output uncertainty, we have $x \sim f(x)$, $x \in \mathcal{D}_x$ and $y = g(x) \sim f(y)$, $y \in \mathcal{D}_y$.

Often, training sets may contain incomplete or incorrect information introduced by noise, which may exists in varied situations of data pre-processing and collection, obscuring the original inputs. As a result, processing the incompletion before learning can generate uncertainty for an input by estimating the missing attributes of this input under the assumption of a joint Gaussian distribution by which the missing and observed attributes of this input are distributed. On the other hand, incorrect or corrupted inputs that are observed can introduce uncertainties through chosen conditional distribution models related to their unknown original counterparts. No matter where these uncertainties are generated from, they contain the estimated amounts or information by which the observed or calculated inputs may differ from the original inputs. In this thesis, we will focus on the input uncertainty.

Here we have some examples of input uncertainty. For instance, in geographic information systems (GIS), there exists a raster-based model for structural landscape classification (Canters et al., 2002). This model uses two types of input data: a digital elevation model (DEM) and a land-cover map. Combining both data sources, for each cell in the landscape model, four structural indices (two from each type) are calculated. Based on the values of these indices for each cell, a landscape typology is derived, reflecting two important characteristics, openness of the landscape and the degree of landscape homogeneity. The land-cover map will be derived from remotely sensed data and some

input data can be perturbed during the data collection. For the elevation input, there are some observed differences in elevation between the DEM and an independent set of control points. The fact that the perturbation and the observed differences are present in data collection can introduce input uncertainties in the land-cover map and the DEM, and may affect the quality of the landscape classification model.

Another example is given by Cullen and Frey (1999). Probabilistic exposure assessments are carried out for exposure models and a diverse set of environmental hazards. Exposure models combine information about the frequency, intensity and duration of human contact with environmental contaminants and/or radionuclides through inhalation, ingestion, and dermal absorption. The model inputs can be categorised in accordance with the roles they serve. Some describe the degree of contamination of various environmental media, others depict the behaviours, activities, or demographics of exposure and exposed populations, and still others describe human physiology. Input uncertainties of these model inputs may appear as measures of the incompleteness of one's knowledge or information about unknown quantities whose true values could be established if the perfect measuring devices were available. And the quality of the exposure models may be affected by these input uncertainties as well.

Besides, input uncertainties may be introduced in many other aspects of machine learning research: in speech recognition, if the observed noisy input is assumed to be produced by noise corrupting the unknown clean input, the uncertainty for the clean input is a Gaussian whose mean is a linear transformation of the noisy input under the assumption that the clean input and the noisy input have a joint Gaussian distribution (Droppo et al., 2002; Liao and Gales, 2004). And the uncertainty and lack of confidence may also exist in the process of feature extraction from image and video (Wallace et al., 2006). In general, these input uncertainties can be introduced through data pre-processing or collection before learning when the incompleteness or errors appear to make the original inputs at least partially unknown to the learner. The obtained uncertain inputs are distributed by assumed distributions. Moreover, uncertainties, which are introduced through different ways, can lead to different approximations of these uncertainties. However, the uncertain information is typically ignored by traditional learning algorithms, but should be introduced to formulate new learning approaches. So the following questions are raised:

- How is it best to accommodate input uncertainty in supervised learning algorithms?
- What performance advantages are gained over approaches which do not directly include the "input uncertainty" information?
- Which kind of noise can boost or weaken the performance of algorithms which incorporate the "input uncertainty" information?

1.2 Motivation and Relationships to Existing Approaches

Traditionally, many approaches have considered either modelling and removing the uncertainties from inputs by approximating or estimating the input uncertainties, or directly incorporating the input uncertainties into the learner. The different input uncertainty models and the learning algorithms have been proposed in several papers, such as four sources of uncertainty are quantified in concentrations predicted by a multi-media fate model, including input uncertainty in substance properties is quantified using probabilistic modelling (Hauck et al., 2008). In another example, a Bayesian uncertainty framework presented by Huard and Mailhot (2006) allows one to account for input, output and structural uncertainties in the calibration of a model. Then the impact of input uncertainty on the parameters of the hydrological model is studied using this framework. Indeed, Bayesian approach is widely used not only to estimate the effects of model uncertainty (Mackay, 1992) but also to be extended to incorporate input uncertainty into learning systems (Wright, 1999). Draper (1995) discussed a Bayesian approach that can fully assess and propagate structural uncertainty, output uncertainty can then be assessed in accordance with input uncertainty and the structural uncertainty obtained. In Chick (2001), the Bayesian model average (BMA) approach can provide meaningful estimates of the mean of simulation output by quantifying the effects of input uncertainty on simulation output.

Recent advances in machine learning methods have seen significant contribution from kernel-based approaches. These have many advantages, including strong theory and convex optimisation formulation. As a maximum margin method derived from VC theory, SVM not only can accommodate different kernels, but more importantly can hold the upper bound of the generalisation error by maximising the margin can for the optimal solution in classification problems. However, the traditional support vector classification (SVC) can only accommodate isotropic uncertainty information in input space. When a richer description of anisotropic uncertainty is available, it can suffer.

In this work we aim to incorporate knowledge of input uncertainty into traditional SVM to provide more robust kernel-based algorithms in classification. Recent approaches have attempted to model input uncertainty in SVM within the kernel learning framework, such as Lanckriet et al. (2002a,b), Bi and Vapnik (2003); Bi and Zhang (2005), Bhattacharyya (2004); Bhattacharyya et al. (2005), Shivaswamy et al. (2006). They have derived some solutions for classification subject to input uncertainty, leading to quadratically constrained quadratic program (QCQP) and second order cone program (SOCP) formulations.

Comparing with the traditional SVC, the algorithm of the total support vector classification (TSVC) proposed by Bi and Zhang (2005) has an improved performance than SVC because the approach efficiently incorporates input uncertainties into the learner, where these input uncertainties come from the Gaussian-distributed uncertain inputs

which denote the unknown corresponding original inputs. Minimax probability machine (MPM) (Lanckriet et al., 2002b) is an algorithm developed from the moment problem and probability theory, and can be used to classify the contaminated classes of inputs with the assumption that no prior distribution of inputs is available. MPM also leads to another new algorithm proposed in (Bhattacharyya et al., 2005; Shivaswamy et al., 2006), whose structure is close to that of TSVC.

1.3 Contributions

The main contributions of this work are to introduce a new algorithm, uncertainty support vector classification (USVC) by introducing the information of uncertain inputs, and two new iterative algorithms based on USVC and other related methods. All these newly developed algorithms can be deemed as or transformed to maximum margin methods, well controlling the upper bound of the generalisation error and can be extended to accommodate different kernels through a novel kernelisation formulation. The primary contribution of this work has been to combine existing approaches along with new ones geometrically and statistically to provide robust solutions for classification subject to input uncertainty in both classifying the contaminated inputs available in observation, and recovering the original target function, which, together with the unknown original inputs, is indeed from the noiseless data set. More details of the contributions are specified as follows,

- Detailed discussion of different noise models defined in accordance with classification problems. Input uncertainty can be estimated from the observed corrupted inputs and their corresponding unknown original inputs.
- Development of a novel kernel-based maximum margin algorithm named the uncertainty support vector classification (USVC) (Yang and Gunn, 2004). The dual problem of USVC is derived and kernelised to accommodate non-linear case with the introduction of a novel kernelisation formulation.
- Development of an iterative algorithm named the adaptive uncertainty support vector classification (AUSVC) (Yang and Gunn, 2007a). AUSVC combines the characteristics of convex optimisation of USVC and TSVC statistically and geometrically.
- Development of a new algorithm, the minimax probability support vector classification (MPSVC) (Yang and Gunn, 2007b), which borrows the idea of the minimax probability machine (MPM) and combines MPM with other existing SVM-based approaches.

- Analysing and summarising the relationships statistically between USVC, TSVC, AUSVC, MPSVC and the traditional SVC. The algorithmic complexity of these algorithms are compared as well.

1.4 Outline of Thesis

This prelude has given some examples of where input uncertainty is generated under noisy situation and how input uncertainty can affect the inputs in training data set, and has introduced some difficulties that the traditional SVM can not accommodate anisotropic input uncertainty in classification problems. In this section, an outline of the remaining chapters for this thesis will be shown as follows:

Chapter 2 first introduces some noise models that can be used to contaminate original inputs. In classification problems, the applied noise models can be classified into four kinds based on two basic factors, the true target function (the original hyperplane) and the distribution of inputs in training set. The contamination can depend on neither, either or both of these two factors. Dependence on the true target function means that the chosen original inputs move according to the original hyperplane under the contamination. The contamination dependent on the distribution can move the chosen original inputs based on the distribution of inputs. Obviously, to make the classification more difficult, the chosen original inputs can be moved towards the opposite class on purpose. Then like the Gaussian process regression used to predict the missing labels, a characteristic of the Gaussian distribution can be used to estimate the missing attributes (or features) of an input under the assumption that the missing and available attributes of this input follow a joint Gaussian distribution, the resulting uncertain input is a Gaussian. A statistical model is also introduced to analyse the connections between the original and contaminated input under Gaussian noise, a conditional distribution of the contaminated input given the original input is a Gaussian as well. Finally, an input uncertainty model is proposed to provide a distribution for the unknown original input, whose distribution is assumed to be a Gaussian.

Chapter 3 describes the incorporation of input uncertainty to derive a new maximum margin method USVC based on the traditional SVC. The resulting optimisation problem is a SOCP with a unique solution. The primal and dual problems of USVC can be extended to a non-linear case by introducing a novel kernelisation formulation. USVC has the similar formulation as that of SVC except for the introduction of the covariance matrices and the dual variables α, $\boldsymbol{\beta}$ in the primal and dual problems respectively. Moreover, USVC behaves in a similar manner to SVC in the case of soft margin applied with the regularisation parameter. Different kernel functions are then tested with some synthetic data sets.

Chapter 4 first introduces the definition of TSVC, compares TSVC with USVC mathematically and geometrically, and analyses their fundamentally statistical difference which is actually based on the different assumptions of the probabilities of the unknown original inputs being classified correctly. Meanwhile, the optimisation problem of TSVC is extended to accommodate a more general case, anisotropic input uncertainty instead of originally proposed isotropic input uncertainty. Secondly, although USVC has the same primal problem as that of the introduced second order cone programming formulation (SOCPF), USVC is more likely to be strong duality than SOCPF because of their different dual transformation used. Finally, MPM can be applied in such classification problems in which inputs are considered together for each class without prior knowledge of distribution. The upper bound on the probability of misclassification of future data set to be minimised in MPM.

Chapter 5 describes the development of two new algorithms. AUSVC is an iterative algorithm based on USVC and TSVC, it aims to achieve a better classification performance by adaptively adjusting the probability of every single unknown original input being misclassified by future optimal hyperplane. The algorithm of MPSVC combines the statistical approach of USVC under the assumption of Gaussian prior and the idea of classifying inputs without prior knowledge in MPM. MPSVC maximises the sum of the probabilities of inputs being correctly classified with the assumption of Gaussian prior and the bounds of these probabilities constrained by the existing results from AUSVC. MPSVC aims to achieve a balance between classifying the corrupted inputs (classification) and recovering the original target function (restoration) when no further request is confirmed. The connections of all existing SVM-based algorithms are analysed geometrically and statistically as well as the comparison of their algorithmic complexities.

Chapter 6 is concerned with measuring and comparing the ability of classification and restoration of the SVM-based algorithms for classification subject to input uncertainty and some other existing methods on a number of real world data sets. Three different settings are applied to the training data sets. The training and test sets are contaminated by noise under three different settings which can be classified as severe contamination applied by Bi and Zhang's setting, moderate contamination applied by the general setting and light contamination applied by the reverse setting. The original and contaminated test sets are used to evaluate the optimal solutions of different learners which have been trained with the corrupted training sets. The measures obtained are compared with each other by a statistical comparison, the Friedman test, and its post-hoc analyses.

Chapter 7 draws a review of the overall contributions of this thesis, and brings forward some ideas for future work.

Chapter 2

Noise Models

Typically, data sets are not ideal in most real-life machine learning problems; different kinds of uncertain or incorrect information will be introduced when measuring or processing the input data: The noise (or errors) may exist in the observation of the real world, in the pre-processing operations applied on the data, in the discretisation of the data or in the representation of the data, which can lead to incorrectness or incompletion of the obtained data. All these adversities have opportunities to alter the original input data. As a result, processing the incompletion can generate uncertainties, which are based on the model selection and the parameter estimation of the chosen model. Generally, the analysis of input uncertainty problems focus on two aspects, the input uncertainty model and the propagation of all noise to the system output or target, the output uncertainty model. Different uncertainty models and their related learning algorithms have been studied in several papers, including Gaussian process implemented as uncertainty prediction (Williams and Rasmussen, 1996; Girard et al., 2002), Bayesian method assessment on input uncertainties (Chick, 1997; Williams and Barber, 1998; Zouaoui and Wilson, 2001; Aires et al., 2004).

2.1 Different Noise Models

Most of the noise models that are related to the degradation encountered in real application of classification under contamination can be classified into four kinds of noise models, which are determined by two basic components of classification problems: the original input data sets and the target function used to discriminate the original input data sets. Several examples of these noise models will be shown and illustrated in more details by introducing some notations. Let $EX(g, \mathcal{D}, \eta)$ denote the noise model and the learner for a classification problem is provided with $EX(g, \mathcal{D}, \eta)$ such that each call to $EX(g, \mathcal{D}, \eta)$ returns a labelled input data, $\{\boldsymbol{x}, y\}$, where $\boldsymbol{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ denotes the attributes, $y \in \{-1, 1\}$ denotes the label and $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ is a training data set,

where l is the size of this data set. $g(\boldsymbol{x}) = 0$ represents a target function, which is the original hyperplane of this classification problem. η represents the rate (or probability) that input data are contaminated by noise. In this thesis, the possible changes introduced by output noise are not considered, we only discuss problems with input data uncertainties introduced by input noise. Therefore, the inputs are corrupted by input noise affecting only the attributes, their labels are set unchanged as the default setup in contamination for all data sets thereafter. Before analysing noise models, we first introduce some simple additive noise, which can be exploited by noise models to contaminate input data.

2.1.1 Traditional Additive Noise

Noise is usually either additive or multiplicative. Additive noise is the noise additive to input data. Additive noise exists no matter whether input data exist or not. While multiplicative noise is strictly related to input data and it appears only when input data exist. This thesis only focus on additive noise which is zero-mean and white. White noise is spatially uncorrelated: the noise for each input data is independent and identically distributed (i.i.d.).

Example 2.1. *Some traditional multivariate noise models are given in this example.*

- ***Gaussian noise*** (Miller and Ruben, 1966) *is statistical noise that has a probability density function of Gaussian distribution. In other words, the values that the noise can take on are Gaussian-distributed. Gaussian noise has an important property: one can not do any better than the linear average to estimate the mean of a stationary Gaussian random variable. This makes Gaussian noise a worst-case scenario for non-linear estimators to restore the original inputs, in the sense that the improvement over linear estimators is least for Gaussian noise. To improve on the results obtained by linear estimator, non-linear estimators can exploit only the non-Gaussianity of the distributions of inputs. Its multivariate probability density function is:*

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}\right), \tag{2.1}$$

  where $\boldsymbol{x} \in \mathbb{R}^n$ is a random input that has a multivariate Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, $\mathbf{0}$ is $n \times 1$ zero mean vector, Σ is a $n \times n$ positive definite covariance matrix and $|\Sigma|$ is the determinant of Σ.

- ***Laplacian noise*** (Norton, 1984) *is statistical noise that has a probability density function of Laplace distribution, which is a continuous probability distribution and also called the double exponential distribution because it can be thought of as two exponential distributions spliced together back to back. The difference between two*

i.i.d. exponential values that the noise can take on is governed by a Laplace distribution. Generally, Laplace distribution is an asymmetric distribution $\boldsymbol{x} \sim \mathcal{L}(\mu, \Sigma)$ (Kotz et al., 2003)*, where* $\boldsymbol{x} \in \mathbb{R}^n$ *is a random input that has a multivariate Laplace distribution. Here, parameter* $\mu \in \mathbb{R}^n$ *controls both location and skewness, and* Σ *is a* $n \times n$ *non-negative definite symmetric matrix. When* $\mu = \mathbf{0}$*, asymmetric Laplace distributions degenerate to symmetric ones. The multivariate probability density function of symmetric Laplace distribution is presented by* Anderson (1992)*,*

$$f(\boldsymbol{x}) = \frac{2}{(2\pi)^{n/2}|\Sigma|^{1/2}} \left(\frac{\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}}{2}\right)^{\nu/2} K_\nu\left(\sqrt{2\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}}\right), \tag{2.2}$$

where $\nu = (2-n)/2$ *and* $K_\nu(u)$ *is the modified Bessel function of the third kind. Non-linear estimators can provide a much more accurate estimate of the mean of a stationary Laplacian random variable than the linear average.*

- ***Uniform noise*** (Weisstein) *is statistical noise that follows a uniform distribution, which, sometimes also known as a rectangular distribution, is a distribution that has constant probability. Uniform noise provides a useful comparison with Gaussian noise. The linear average is a comparatively poor estimator for the mean of a uniform distribution. This implies that non-linear estimators should be better at recovering the original inputs from uniform noise than from Gaussian noise. Its probability density function is given by:*

$$f(\boldsymbol{x}) = \begin{cases} \frac{1}{2\sqrt{3}\sigma}, & \text{for } |\boldsymbol{x}| < \sigma\sqrt{3}, \\ 0, & \text{else.} \end{cases} \tag{2.3}$$

To clearly show the difference between these noise, a univariate example is shown in Figure 2.1.

2.1.2 Noise Model Independent of the Function and the Distribution

This kind of noise model can generate contaminated input data from original input data by following the parameters that are not related to the target function or the distribution of input data. Traditional additive noise can be used in this noise model. Here we introduce a direct example.

Example 2.2. Malicious noise model (Valiant, 1985; Kearns and Li, 1993) *is an example with independent noise, in which the noise is added to the input data in some probability. The model* $EX(g, \mathcal{D}, \eta)$ *then randomly chooses between returning the noiseless data* $\{\boldsymbol{x}, y\}$*, with probability* $1-\eta$*, and returning the noisy data* $\{\boldsymbol{x}', y\}$*, with probability* η*, where* $\boldsymbol{x}' \in \mathbb{R}^n$ *are drawn uniformly at random over the unit interval.*

$$\{\boldsymbol{x}, y\} = \begin{cases} \{\boldsymbol{x}, y\}, & \text{with probability } 1-\eta, \\ \{\boldsymbol{x}', y\}, & \text{with probability } \eta. \end{cases} \tag{2.4}$$

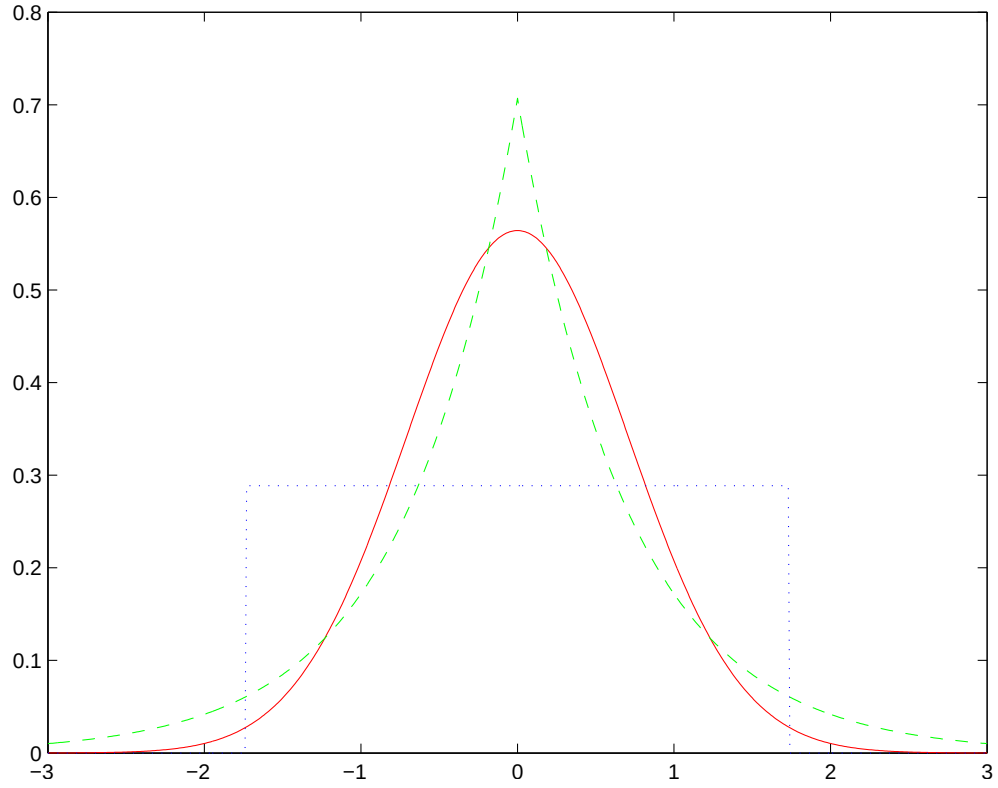
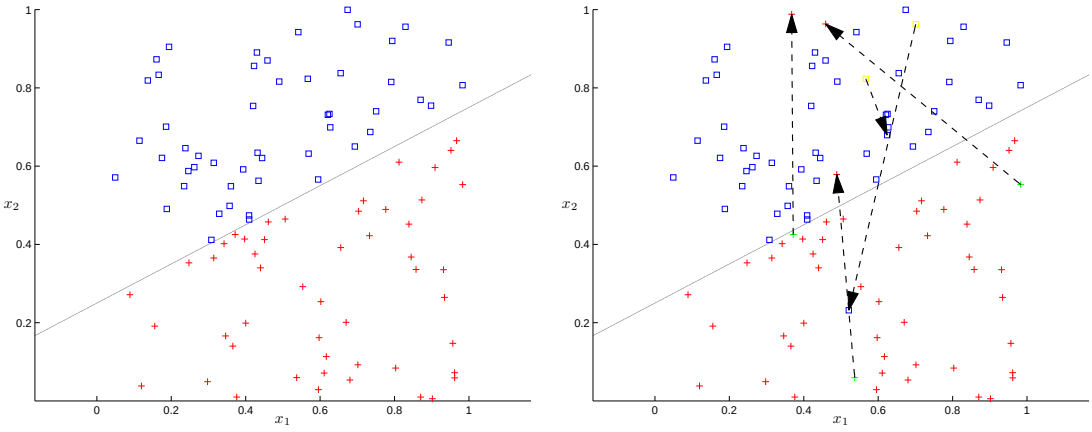


FIGURE 2.1: Univariate Gaussian noise, Laplacian noise and uniform noise.

This random noise (or called independent noise) can vary to particularly give an adversary the power to "distort" the perception of the target function no matter what the distribution of input data set $\mathcal{D}$ is and where the target function $g(\boldsymbol{x})$ is. Figure 2.2 *shows a two-dimensional example of malicious noise model.*



(a) (b)

FIGURE 2.2: 2-D example of a malicious noise model $EX(g, \mathcal{D}, \eta)$, in which $\eta = 0.05$, $\{\boldsymbol{x}, y\} \in \mathcal{D}$ and $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ denotes the target function shown as the solid line in the figure, where $\boldsymbol{w} = [0.5, -1]^T$ and $b = 0.25$. The original input data are displayed in (a), and (b) shows the contaminated inputs. Light green pluses and light yellow squares in Figure 2.2(b) represent the original inputs before being contaminated, dashed arrow shows how individual inputs are corrupted by this malicious noise model.

2.1.3 Noise Model Dependent on the Function

In this kind of noise model, the parameters are solely related to the target function. In other words, the noise model can add noise to input data with references to the original hyperplane obtained along with the input data.

Example 2.3. Random attribute noise model (Sloan, 1988, 1995; Goldman and Sloan, 1995) *adds noise to each input* $\boldsymbol{x}$ *by independently flipping each bit* x_i *of input* $\boldsymbol{x}$ *to* $\overline{x}_i$ *with probability* η *for* $1 \leq i \leq n$. *The noise model returns the altered input and the original label* y.

Strictly speaking, not only this so called random attribute noise model, but also all four kinds of classified noise models can be reckoned as attribute noise models by which the attributes of input data are contaminated. Random attribute noise model situations where the attributes of the examples are subject to noise, which actually mirrors the original examples based on the origin of coordinate axes. But this noise model is not close related to the target function so that we still want to develop an attribute noise model dependent on the target function.

Example 2.4. Function-dependent attribute noise model *generally provides such noise, which is imported to noise-free input data* $\{\boldsymbol{x}, y\}$ *by mapping* $\boldsymbol{x} \in \mathbb{R}^n$ *to* $g_v(\boldsymbol{x}) \in \mathbb{R}^n$ *with probability* η. *While the noiseless input data are returned by the noise model with probability* $1 - \eta$.

$$\{\boldsymbol{x}, y\} = \begin{cases} \{\boldsymbol{x}, y\}, & \text{with probability } 1 - \eta, \\ \{g_v(\boldsymbol{x}), y\}, & \text{with probability } \eta, \end{cases} \tag{2.5}$$

where g_v *is a function that contains the parameters provided by* g. $g_v(\boldsymbol{x})$ *is then obtained as the contaminated counterparts of the noiseless input* $\boldsymbol{x}$ *by moving along the traces which are strictly related to the obtained parameters of* g. *A two-dimensional example is shown in* Figure 2.3, *in which the target function is*

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = 0, \tag{2.6}$$

and the contaminated inputs are given by

$$g_v(\boldsymbol{x}) = \boldsymbol{x} + \operatorname{sgn}(r_1 - 0.5) r_2 \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \tag{2.7}$$

where r_1 *and* r_2 *are uniformly distributed random variables within the unit interval,* sgn *is the sign function. No matter which directions the contaminated inputs are going to move in* Figure 2.3, *the traces from* $\boldsymbol{x}$ *to* $g_v(\boldsymbol{x})$ *are always vertical to the target function* $g(\boldsymbol{x}) = 0$.

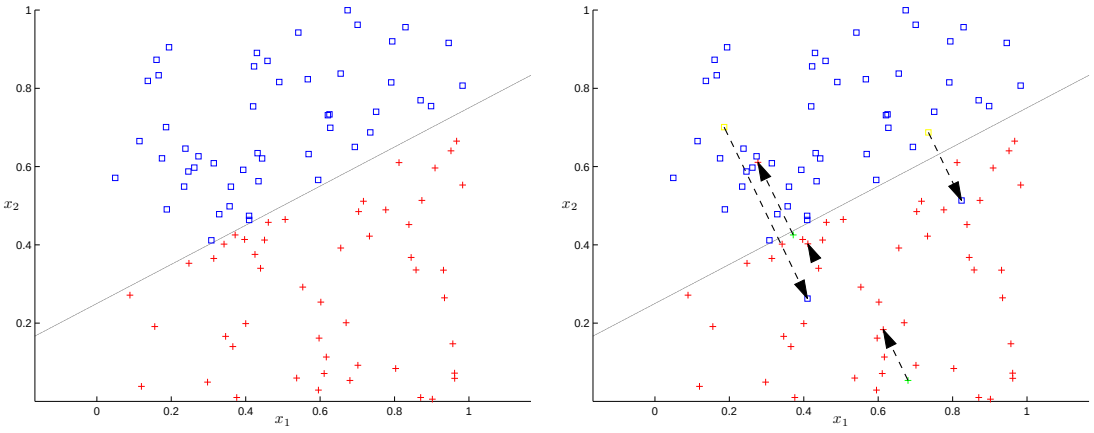


FIGURE 2.3: 2-D example of a function-dependent attribute noise model $EX(g, \mathcal{D}, \eta)$, in which $\eta = 0.05$, $\{\boldsymbol{x}, y\} \in \mathcal{D}$ and $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ denotes the target function shown as the solid line in the figure, where $\boldsymbol{w} = [0.5, -1]^T$ and $b = 0.25$. The original input data are displayed in (a), and (b) shows the contaminated inputs. Light green pluses and light yellow squares in Figure 2.3(b) represent the original inputs before being contaminated, dashed arrow shows how individual inputs are corrupted by this noise model.

2.1.4 Noise Model Dependent on the Distribution

The noise model dependent on the distribution of the input data is the third listed noise model, which normally includes two main types, parametric distribution-dependent noise model and non-parametric distribution-dependent noise model. The latter does not rely on assumptions that the data are drawn from a given probability distribution and the generated contaminated inputs can refer to a statistic whose interpretation does not depend on the population fitting any parametrised distributions. While in parametric distribution-dependent noise model, noise is introduced into input data with references to the parameters obtained from the distribution of input data. Generally, we can assume a proper distribution for the input data with its parameters being approximated by an expectation-maximisation (EM) algorithm for finding maximum likelihood estimates of the parameters. However, EM is an iterative method which alternates between performing an expectation step and a maximisation step by computing the expected value of the log likelihood function with respect to the conditional distribution under the current estimate of the parameters and computing the parameters which maximise the expected log likelihood respectively. Alternatively, some simple non-parametric noise models can contaminate input data with similar effects according to varied distributions of partial inputs. Here, we develop an example,

Example 2.5. *M-nearest neighbour distribution-dependent attribute noise model outputs the corrupted input for the selected input by evaluating the nearest M neighbours or M chosen inputs from either class that are related to the selected input.*

The specific procedure of generating such noise in M-nearest neighbour distribution-dependent attribute noise model is listed as follows, a two-dimensional example is shown in Figure 2.4.

1. *The model $EX(g, \mathcal{D}, \eta)$ randomly selects the input data $\{\boldsymbol{x}, y\} \in \mathcal{D}$ that is going to be contaminated by noise with probability η;*

2. *Choosing M inputs $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M} \in \mathcal{D}$ according to the selected input $\{\boldsymbol{x}, y\}$ and the inputs $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M}$ strictly belong to the same class. Therefore, if $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M}$ and $\{\boldsymbol{x}, y\}$ are from the same class, $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M}$ are the nearest M neighbours of $\{\boldsymbol{x}, y\}$. On the other hand, considering the possible move to the other class under contamination, we may have $y_j = -y$, $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M}$ are hereby the closest inputs from the other class to $\{\boldsymbol{x}, y\}$;*

3. *Searching the input with the highest sparsity among the M inputs $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M}$ chosen in step 2, the mean value $\boldsymbol{x}'$ of the attributes of the nearest M neighbours of this obtained input is computed;*

4. *Replacing the original attributes $\boldsymbol{x}$ with this obtained mean value $\boldsymbol{x}'$.*

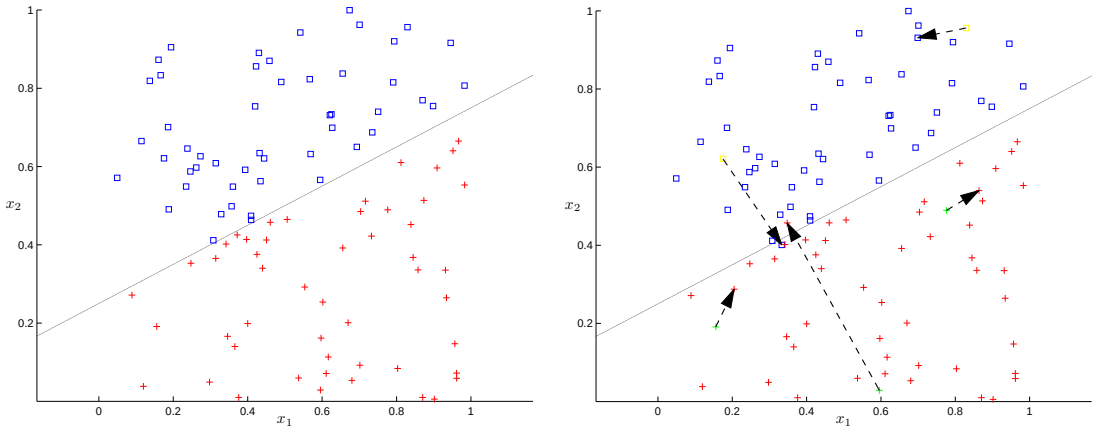


FIGURE 2.4: 2-D example of a five-nearest neighbour distribution-dependent attribute noise model $EX(g, \mathcal{D}, \eta)$, in which $\eta = 0.05$, $\{\boldsymbol{x}, y\} \in \mathcal{D}$ and $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}+b = 0$ denotes the target function shown as the solid line in the figure, where $\boldsymbol{w} = [0.5, -1]^T$ and $b = 0.25$. The original input data are displayed in (a), and (b) shows the contaminated inputs. Light green pluses and light yellow squares in Figure 2.4(b) represent the original inputs before being contaminated, dashed arrow shows how individual inputs are corrupted by this noise model.

2.1.5 Noise Model Dependent on the Function and the Distribution

With probability η, different noise models $EX(g, \mathcal{D}, \eta)$ return corrupted inputs about which different assumptions may be made, including the noise models dependent on the

target function or the distribution of the input data. In particular, the chosen input may be maliciously contaminated by an adversary who has infinite computing power, and has knowledge of the target function $g(\boldsymbol{x})$, the distribution $\mathcal{D}$ and the learning algorithm. Such a noise model called nasty noise model is designed by Bshouty et al. (2002) for proving the worst case bounds of accuracy on learning algorithms.

Example 2.6. *In* **nasty noise model**, *the adversary gets to see the whole data requested by the learning algorithm before it is given to the algorithm and modifies E of all l examples, where E represents the number of a fraction of all examples and is a random variable distributed by the binomial distribution with parameters η and l. The probability of selecting exactly E out of all l examples is given by the probability mass function,*

$$\Pr\{E = n\} = \begin{pmatrix} l \\ n \end{pmatrix} \eta^n (1-\eta)^{l-n} \tag{2.8}$$

This distribution makes the number of examples modified be the same as if it were determined by l independent tosses of an η-biased coin. The E inputs chosen by the adversary are removed from the data set and replaced by any other less "informative" and even misleading data. While the $l - E$ inputs not chosen by the adversary remain unchanged.

Example 2.7. *Generally, the adversary may only know part of all possible "informative" data important to the learning algorithm. For simplicity, a weaker variant of nasty noise model called* **nasty classification noise model** *is introduced by* Bshouty et al. (2002)*, in which the adversary modifies the chosen data according to the original classification. For classification problems subject to input uncertainty, especially when support vector machines (SVMs) and their related approaches are used as learning algorithms, nasty classification noise model can be extended to* **nasty classification function and distribution dependent attribute noise model**, *in which the support vectors are selected as "informative" data in contamination. Besides, the methods of generating corrupted examples in function-dependent or distribution-dependent noise models are reintroduced into this model as well. The specific procedure of generating noise in such a noise model is listed as follows,*

1. *Support vector classification is applied on the original examples with a proper chosen regularisation parameter;*

2. *The model $EX(g, \mathcal{D}, \eta)$ selects in total E input data $\{\boldsymbol{x}, y\} \in \mathcal{D}$ out of all obtained support vectors, where E is a random variable following the binomial distribution with parameters η and l;*

3. *The contamination dependent on the distribution or the target function is determined by two uniformly distributed random variables within the unit interval, r_3 and r_4, which are related to each $\{\boldsymbol{x}, y\}$. Due to the sequence of applying these two random variables in the procedure, the probability that $\{\boldsymbol{x}, y\}$ is contaminated*

solely based on the target function is 0.5*, the probability that* $\{\boldsymbol{x}, y\}$ *is contaminated based on the distribution of input data is* 0.25*, and the probability of* $\{\boldsymbol{x}, y\}$ *being contaminated according to both factors is* 0.25*;*

4. *If* $r_3 < 0.5$*, then go to step 5; otherwise, search the input with the highest sparsity among* $\{\boldsymbol{x}, y\}$*'s* M *nearest neighbours* $\{\boldsymbol{x}_j, y_j\}_{j=1}^{M}$ *which are chosen from the other class different to the class of* $\{\boldsymbol{x}, y\}$*. The mean value* $\boldsymbol{x}^{'}$ *of the attributes of the nearest* M *neighbours of this obtained input is computed and the original attributes* $\boldsymbol{x}$ *is replaced with* $\boldsymbol{x}^{'}$*;*

5. *If* $r_4 < 0.5$*, and* $r_3 < 0.5$ *obtained in step 4, or purely* $r_4 \geq 0.5$*, then* $\boldsymbol{x}$ *is corrupted to* $g_v(\boldsymbol{x})$ *by following* (2.7) *except that the sign function is not determined by the random variable* r_1 *but fixed to move examples from one class towards the other class.*

A two-dimensional example of nasty classification function and distribution dependent attribute noise model is shown in Figure 2.5*. There are in total* 8 *support vectors chosen*

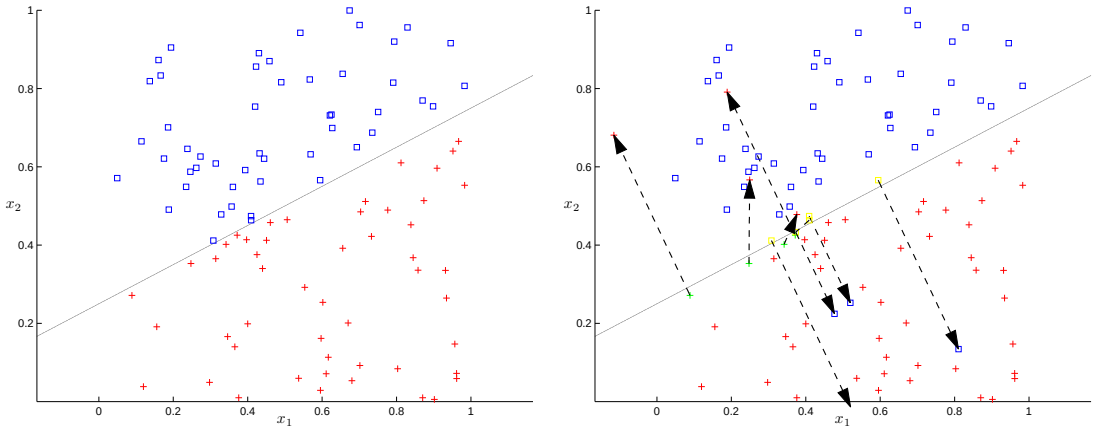


FIGURE 2.5: 2-D example of a nasty classification function and distribution dependent attribute noise model $EX(g, \mathcal{D}, \eta)$, in which $\eta = 0.05$, $\{\boldsymbol{x}, y\} \in \mathcal{D}$ and $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ denotes the target function shown as the solid line in the figure, where $\boldsymbol{w} = [0.5, -1]^T$ and $b = 0.25$. The original input data are displayed in (a), and (b) shows the contaminated inputs. Light green pluses and light yellow squares in Figure 2.5(b) represent the original inputs before being contaminated and the inputs in the middle of contamination process, dashed arrow shows how individual inputs are corrupted by this noise model.

as "informative" input data in Figure 2.5*, where five nearest neighbours are evaluated to replace the original data during the contamination dependent on the distribution. When the contamination dependent on the target function is applied on the input data, the traces from* $\boldsymbol{x}$ *to* $g_v(\boldsymbol{x})$ *are designed to be vertical to the target function.*

More generally, the adversary may need to contaminate data sets generated from non-linearly separable problems, in which linear function can no longer represent the target

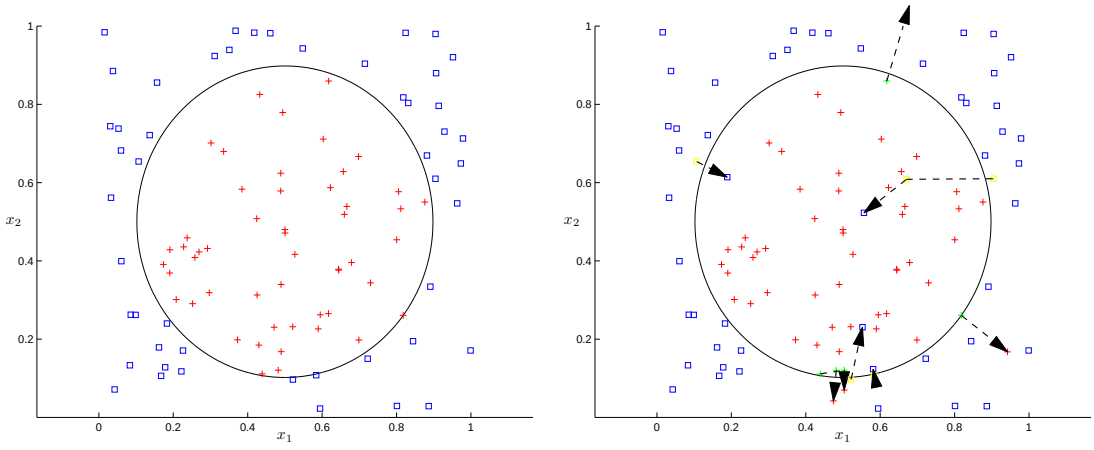
function especially when the input data are corrupted by the noise depending on the target function. In fact, we can borrow the kernel method proposed in SVMs to extend the contamination to non-linearly separable case.

Example 2.8. *Compared with that the contaminated inputs are directly computed by the weight vector $\boldsymbol{w}$ of the target function in linear case, the weight vector approximated through kernel functions in non-linear case is the main difference. Specifically speaking,*

1. *Support vector classification (SVC) is first applied for an optimal hyperplane if the exact expression of the non-linear target function is not available;*
2. *A small ball which has the same dimensions as the original input $\boldsymbol{x}$ is then created by fixing the original input as its centre;*
3. *We search all possible points on the ball surface to find an exact point $\boldsymbol{x}_s$ with maximal difference value between the distance d from $\boldsymbol{x}_s$ to the optimal hyperplane and the distance from $\boldsymbol{x}$ to the optimal hyperplane;*

$$d = y\left(\sum_{j=1}^{l} y_j \alpha_j K(\boldsymbol{x}, \boldsymbol{x}_j) + b\right) \tag{2.9}$$

The weight vector $\boldsymbol{w}$ can be obtained by computing $\boldsymbol{x}$ and $\boldsymbol{x}_s$. A two-dimensional example with polynomial kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^2$ used in SVC is shown in Figure 2.6*, where in total 8 support vectors chosen as "informative" input data according to the binomial distribution. The other settings remain the same as those used in* Figure 2.5.



(a) (b)

FIGURE 2.6: 2-D example of a nasty classification function and distribution dependent attribute noise model $EX(g, \mathcal{D}, \eta)$, in which $\eta = 0.05$, $\{\boldsymbol{x}, y\} \in \mathcal{D}$ and $g(\boldsymbol{x}) = \|\boldsymbol{x} - [0.5, 0.5]^T\| = 0.398$ denotes the polynomial target function shown as the solid line in the figure. The original input data are displayed in (a), and (b) shows the contaminated inputs. Light green pluses and light yellow squares in Figure 2.6(b) represent the original inputs before being contaminated and the inputs in the middle of contamination process, dashed arrow shows how individual inputs are corrupted by this noise model.

However, unlike the assumptions made that the learning algorithm is SVC in Example 2.7 and 2.8, the adversary has less opportunities of knowing the learning algorithm which is going to request the corrupted data. In this thesis, only the target function and the distribution of the input data are considered as influencing factors that can affect classification problems subject to input uncertainty. When the noise model is dependent on both the target function and the distribution, the contamination can possibly achieve more adversarial results in the theory.

2.2 Input Uncertainty

Although we have presented several noise models that can introduce different kinds of noise in their own ways and contaminate the input data sets, it is unclear for learning algorithms to exactly know how the original input data are corrupted by noise? Because all inputs that the learning algorithms have in the training session are contaminated input data or incomplete input data which may be brought in through the contamination introduced by adversaries. As a result, processing the incompletion generates input uncertainties. On the other hand, what we consider in classification problems includes not only the contaminated input data, but also their original counterparts which indeed affect the recovery of the unknown target function in adversarial circumstances, but is unknown to the learning algorithms. In this case, the information of an original input data can be provided in an estimated distribution, following which this original input data is deemed as an uncertain input near its corresponding contaminated input data. Therefore, a possible input uncertainty model that includes both the contaminated input data and the information of its original counterpart should be proposed first.

2.2.1 Gaussian Processes on Output Uncertainty Prediction

In the mathematical theory of probability, a Gaussian process is a generalisation of a Gaussian distribution, which describes a finite-dimensional random variable, to functions. Formally, the definition of Gaussian processes is given by Rasmussen and Williams (2006): A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. Here, the random variables represent the value of the function $g(\boldsymbol{x})$ at location $\boldsymbol{x}$. A Gaussian process is fully specified by its mean function and covariance function, which are defined as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbf{E}\left[g(\boldsymbol{x})\right], \\ k(\boldsymbol{x}, \boldsymbol{x}^{'}) &= \mathbf{E}[(g(\boldsymbol{x}) - m(\boldsymbol{x}))(g(\boldsymbol{x}^{'}) - m(\boldsymbol{x}^{'}))], \end{aligned} \tag{2.10}$$

and the Gaussian process is written as

$$g(\boldsymbol{x}) \sim \mathcal{N}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}^{'})\right). \tag{2.11}$$

The method of Gaussian processes on regression problems was first proposed by O'Hagan (1978). With a Gaussian prior placed over the function values, the covariance function of the function f is approximated instead of directly introducing f, the Gaussian process modelling framework can get the predictive distribution of the function values in accordance with new inputs.

Further details mainly come from Williams and Rasmussen (1996). Given a Gaussian prior on the function f and an observed data set $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$, and we have

$$y_i = g(\boldsymbol{x}_i) + \epsilon_i, \tag{2.12}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is an additive i.i.d. Gaussian noise of variance σ_ϵ^2, $\boldsymbol{x}_i \in \mathbb{R}^n$ denotes an input vector which represents the attributes or the features of an input, and y denotes a scalar output which is also called a target or a label. Without loss of generality, all l vector inputs can be aggregated in a $l \times n$ design matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l]^T$ and the targets are collected in a $l \times 1$ vector $\boldsymbol{y}$. The vector inputs and their outputs of the new inputs can also be represented by a $l_* \times n$ matrix $\boldsymbol{X}_*$ and a $l_* \times 1$ vector $\boldsymbol{y}_*$, where in total l_* new inputs need to predict their outputs and $\boldsymbol{X}_* = [\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_{l_*}^*]^T$. We can write the joint distribution of the observed target values and the function values at the new inputs under the prior as

$$\begin{bmatrix} \boldsymbol{y} \\ g(\boldsymbol{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \quad \begin{bmatrix} \mathbf{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_\epsilon^2 \mathbf{I} & \mathbf{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right), \tag{2.13}$$

where $\mathbf{K}(\boldsymbol{X}, \boldsymbol{X}_*)$ denotes the $l \times l_*$ matrix of the covariance functions evaluated at all pairs of the observed and the new inputs, and similarly for the other entries $\mathbf{K}(\boldsymbol{X}, \boldsymbol{X})$, $\mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X})$ and $\mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X}_*)$.

$$\mathbf{K}(\boldsymbol{X}, \boldsymbol{X}_*) = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1^*) & \ldots & \ldots & \ldots & k(\boldsymbol{x}_1, \boldsymbol{x}_{l_*}^*) \\ \vdots & \ddots & & & \vdots \\ \vdots & & k(\boldsymbol{x}_p, \boldsymbol{x}_q^*) & & \vdots \\ \vdots & & & \ddots & \vdots \\ k(\boldsymbol{x}_l, \boldsymbol{x}_1^*) & \ldots & \ldots & \ldots & k(\boldsymbol{x}_l, \boldsymbol{x}_{l_*}^*) \end{bmatrix}, \tag{2.14}$$

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q^*) = \mathrm{Cov}(g(\boldsymbol{x}_p), g(\boldsymbol{x}_q^*)), \qquad 1 \le p \le l, \quad 1 \le q \le l_*.$$

The zero mean functions and the covariance functions obtained in (2.13) and (2.14) are directly derived from equation (2.10). In a more general case, a mapping function $\phi(\boldsymbol{x})$ that can map a n-dimensional input vector $\boldsymbol{x}$ into a higher-dimensional feature space is introduced into function f with the weight parameter $\boldsymbol{w}$, we have

$$g(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \boldsymbol{w}, \tag{2.15}$$

and a zero mean Gaussian prior with covariance matrix $\boldsymbol{\Sigma}_p$ is put on $\boldsymbol{w}$, we have $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)$. Therefore, (2.10) can be computed in accordance with $\boldsymbol{x}_p$, $\boldsymbol{x}_q^*$ and the Gaussian prior of $\boldsymbol{w}$,

$$\begin{aligned} m(\boldsymbol{x}_p) &= \mathbf{E}[g(\boldsymbol{x}_p)] = \phi(\boldsymbol{x}_p)^T \mathbf{E}[\boldsymbol{w}] = 0, \\ m(\boldsymbol{x}_q^*) &= \mathbf{E}[g(\boldsymbol{x}_q^*)] = \phi(\boldsymbol{x}_q^*)^T \mathbf{E}[\boldsymbol{w}] = 0, \\ k(\boldsymbol{x}_p, \boldsymbol{x}_q^*) &= \mathbf{E}[(\phi(\boldsymbol{x}_p)^T \boldsymbol{w} - m(\boldsymbol{x}_p))(\phi(\boldsymbol{x}_q^*)^T \boldsymbol{w} - m(\boldsymbol{x}_q^*))] \\ &= \phi(\boldsymbol{x}_p)^T \mathbf{E}[\boldsymbol{w}\boldsymbol{w}^T] \phi(\boldsymbol{x}_q^*) = \phi(\boldsymbol{x}_p)^T \boldsymbol{\Sigma}_p \phi(\boldsymbol{x}_q^*). \end{aligned} \tag{2.16}$$

Thus $g(\boldsymbol{x}_p)$ and $g(\boldsymbol{x}_q^*)$ are jointly Gaussian with zero mean and covariance given by $\phi(\boldsymbol{x}_p)^T \boldsymbol{\Sigma}_p \phi(\boldsymbol{x}_q^*)$. Indeed, all function values corresponding to any number of the observed and new inputs are jointly Gaussian. Moreover, the covariance between the outputs is written as a function of the inputs. Like SVMs, the covariance function $k(.,.)$ can be treated as a kernel function. A very common Gaussian type of kernel is the squared exponential covariance function,

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q^*) = \mathrm{Cov}(g(\boldsymbol{x}_p), g(\boldsymbol{x}_q^*)) = \exp\left[-\frac{1}{2}(\boldsymbol{x}_p - \boldsymbol{x}_q^*)^T \boldsymbol{\Lambda}^{-1} (\boldsymbol{x}_p - \boldsymbol{x}_q^*)\right], \tag{2.17}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1^2, \ldots, \lambda_n^2]^T$ allows for various length scales in different input directions. The covariance between two targets $g(\boldsymbol{x}_p)$, $g(\boldsymbol{x}_q^*)$ is related to the distance between the two corresponding inputs $\boldsymbol{x}_p$ and $\boldsymbol{x}_q^*$ under the kernel metric. The covariance is almost unity between variables whose corresponding inputs are very close, and decreases as their distance in the input space increases.

Therefore, the predictive distribution of the function values of the new inputs is obtained by deriving the conditional distribution corresponding to conditioning the joint Gaussian prior distribution on the observations, we have,

$$\begin{aligned} g(\boldsymbol{X}_*)|\boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y} &\sim \mathcal{N}(\bar{\boldsymbol{g}_*}, \mathrm{Cov}(\boldsymbol{f}_*)), \qquad \text{where} \\ \bar{\boldsymbol{g}_*} &= \mathbf{E}[g(\boldsymbol{X}_*)|\boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y}] = \mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X})[\mathbf{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_\epsilon^2 \mathbf{I}]^{-\mathbf{1}} \boldsymbol{y}, \\ \mathrm{Cov}(\boldsymbol{f}_*) &= \mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \mathbf{K}(\boldsymbol{X}_*, \boldsymbol{X})[\mathbf{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_\epsilon^2 \mathbf{I}]^{-\mathbf{1}} \mathbf{K}(\boldsymbol{X}, \boldsymbol{X}_*). \end{aligned} \tag{2.18}$$

Thus the output uncertainties (or the target uncertainties) of the new inputs can be estimated by deriving the conditional distribution of the new inputs' function values $g(\boldsymbol{X}_*)$ given the new inputs $\boldsymbol{X}_*$, the observed inputs $\boldsymbol{X}$ and the function values of the observed inputs $g(\boldsymbol{X})$. This method is also suitable for us to estimate the output uncertainties of some inputs whose labels are missing under the contamination.

2.2.2 Input Uncertainty Prediction

Instead of missing labels, in this thesis, we focus on these cases, in which the attributes of any input are partially unknown, or some of all attributes are missing for any input

under the contamination. Let $\boldsymbol{x}$ denote the input that has missing attributes, $\boldsymbol{x}_k$ is the vector that denotes the known or observed attributes of $\boldsymbol{x}$, $\boldsymbol{x}_m$ is the vector that denotes the unknown or missing attributes of $\boldsymbol{x}$, and $\boldsymbol{x} = [\boldsymbol{x}_k^T, \boldsymbol{x}_m^T]^T$. A characteristic of the Gaussian distribution that is similar to the Gaussian processes regression method can be used to estimate the missing attributes $\boldsymbol{x}_m$ directly from the observed attributes $\boldsymbol{x}_k$ and a predefined joint Gaussian distribution by which $\boldsymbol{x}_k$ and $\boldsymbol{x}_m$ are distributed.

Several theorems that are related to the estimation of the missing attributes were proposed by Mardia et al. (1979). For continuity, the symbols originally used in Mardia et al. (1979) are replaced by our definitions.

Theorem 2.1. (Theorem 3.2.3 in Mardia et al. (1979)) *If* $\boldsymbol{x} = [\boldsymbol{x}_k^T, \boldsymbol{x}_m^T]^T \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, *then* $\boldsymbol{x}_k$ *and* $\boldsymbol{x}_{m.k} = \boldsymbol{x}_m - \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}\boldsymbol{x}_k$ *have the following distributions and are statistically independent:*

$$\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_{kk}), \qquad \boldsymbol{x}_{m.k} \sim \mathcal{N}(\boldsymbol{\mu}_{m.k}, \boldsymbol{\Sigma}_{mm.k}),$$

where

$$\boldsymbol{\mu}_{m.k} = \boldsymbol{\mu}_m - \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}\boldsymbol{\mu}_k, \qquad \boldsymbol{\Sigma}_{mm.k} = \boldsymbol{\Sigma}_{mm} - \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}\boldsymbol{\Sigma}_{km}, \tag{2.19}$$
$$\boldsymbol{\mu} = [\boldsymbol{\mu}_k^T, \boldsymbol{\mu}_m^T]^T, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{kk} & \boldsymbol{\Sigma}_{km} \\ \boldsymbol{\Sigma}_{mk} & \boldsymbol{\Sigma}_{mm} \end{bmatrix}.$$

Theorem 2.2. (Theorem 3.2.4 in Mardia et al. (1979)) *Using the assumptions and notation of* Theorem 2.1*, the conditional distribution of* $\boldsymbol{x}_m$ *for a given value of* $\boldsymbol{x}_k$ *is*

$$\boldsymbol{x}_m|\boldsymbol{x}_k \sim \mathcal{N}\left(\boldsymbol{\mu}_m + \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu}_k), \boldsymbol{\Sigma}_{mm} - \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}\boldsymbol{\Sigma}_{km}\right) \tag{2.20}$$

Theorem 2.2 can be simply proved by using Theorem 2.1's result that $\boldsymbol{x}_{m.k}$ is independent of $\boldsymbol{x}_k$ and its conditional distribution for a given value of $\boldsymbol{x}_k$ is the same as its marginal distribution. On the other hand, $\boldsymbol{x}_m = \boldsymbol{x}_{m.k} + \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}\boldsymbol{x}_k$ and this term is constant when $\boldsymbol{x}_k$ is given. Therefore, the conditional distribution of $\boldsymbol{x}_m|\boldsymbol{x}_k$ is Gaussian, and its conditional mean is

$$\mathbf{E}[\boldsymbol{x}_m|\boldsymbol{x}_k] = \boldsymbol{\mu}_{m.k} + \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}\boldsymbol{x}_k = \boldsymbol{\mu}_m + \boldsymbol{\Sigma}_{mk}\boldsymbol{\Sigma}_{kk}^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu}_k). \tag{2.21}$$

The conditional covariance matrix of $\boldsymbol{x}_m$ is the same as that of $\boldsymbol{x}_{m.k}$, namely $\boldsymbol{\Sigma}_{mm.k}$. Thus the input uncertainties of the missing attributes of any input can be estimated through (2.19) and (2.20). An EM algorithm is used to estimate the parameters of the joint Gaussian distribution of the observed and missing attributes of any input, the detailed procedure was also proposed by Shivaswamy et al. (2006),

1. Initialise the joint Gaussian distribution's parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for input $\boldsymbol{x}$;
2. Estimate $\boldsymbol{x}_m|\boldsymbol{x}_k$ by using (2.19) and (2.20);

3. Recompute and collect the new values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ by using the completed data. If $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ converge, then the obtained values are what we expect, otherwise, return to step 2;

2.2.3 Statistical Models on Input Uncertainty

Besides the case that the input uncertainties are generated from estimating the missing or unknown attributes of inputs, a statistical model that gives a profile of the original data and their contaminated counterparts can describe the input uncertainties as well by using the maximum-likelihood method to estimate the unknown parameters.

The derivatives of this statistical model of the uncertainty mainly come from Bi and Zhang (2005). For convenience in notation understanding, some notations are replaced by some new expressions, which will be used as default notations for all data sets used in this thesis thereafter. Consider a set of training examples $\{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is corrupted with noise and $y_i \in \mathbb{R}$ is a label which is not contaminated by the noise. Let $\boldsymbol{x}_{io}$ denote the unobserved original counterpart of $\boldsymbol{x}_i$. We assume the following data generating process: first $(\boldsymbol{x}_{io}, y_i)$ is generated according to a distribution $\Pr(\boldsymbol{x}_{io}, y_i|\theta)$, where θ is an unknown parameter of the conditional distribution $\Pr(\boldsymbol{x}_{io}, y_i|\theta)$; next, given $(\boldsymbol{x}_{io}, y_i)$, we assume that $\boldsymbol{x}_i$ is generated from $\boldsymbol{x}_{io}$ but independent of y_i in accordance with a distribution $\Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io})$, where θ' is another assumably unknown parameter that are used in the conditional distribution $\Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io})$, and θ' along with θ can be estimated from the data by using the maximum-likelihood estimate. σ_i is a known parameter which is dependent on our estimate of the uncertainty (e.g. variance) for $\boldsymbol{x}_i$. The joint probability distribution of $(\boldsymbol{x}_{io}, \boldsymbol{x}_i, y_i)$ can be written as:

$$\Pr(\boldsymbol{x}_{io}, \boldsymbol{x}_i, y_i) = \Pr(\boldsymbol{x}_{io}, y_i|\theta) \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}). \tag{2.22}$$

The joint probability distribution of $(\boldsymbol{x}_i, y_i)$ is obtained by integrating out the unobserved original quantity $\boldsymbol{x}_{io}$:

$$\Pr(\boldsymbol{x}_i, y_i) = \int \Pr(\boldsymbol{x}_{io}, y_i|\theta) \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}) d\boldsymbol{x}_{io}. \tag{2.23}$$

This model can be reckoned as a mixture model in which each mixture component corresponds to a possible original input $\boldsymbol{x}_{io}$ not observed. The unknown parameters (θ, θ') can be estimated from the data using the maximum-likelihood estimate as:

$$\max_{\theta,\theta'} \sum_i \ln \Pr(\boldsymbol{x}_i, y_i|\theta, \theta') = \max_{\theta,\theta'} \sum_i \ln \int \Pr(\boldsymbol{x}_{io}, y_i|\theta) \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}) d\boldsymbol{x}_{io}. \tag{2.24}$$

Equation (2.24) is a principled approach under the current data generation process. It often leads to a complicated formulation which is difficult to solve. Alternatively, the previous formulation (2.24) can be transformed to an approximation formulation (2.25),

in which each unobserved $\boldsymbol{x}_{io}$ can be simply regarded as a parameter in the probability model, so the maximum-likelihood becomes (Bi and Zhang, 2005):

$$\max_{\theta,\theta'} \sum_i \ln \Pr(\boldsymbol{x}_i, y_i|\theta, \theta') \approx \max_{\theta,\theta'} \sum_i \ln \sup_{\boldsymbol{x}_{io}} [\Pr(\boldsymbol{x}_{io}, y_i|\theta) \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io})]. \tag{2.25}$$

Since large values of $\Pr(\boldsymbol{x}_{io}, y_i|\theta) \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io})$ dominate the summation in the integration $\int \Pr(\boldsymbol{x}_{io}, y_i|\theta) \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}) d\boldsymbol{x}_{io}$, similar effects are derived from equation (2.24) and (2.25), both of which prefer a parameter configuration such that the product of $\Pr(\boldsymbol{x}_{io}, y_i|\theta)$ and $\Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io})$ is large for some $\boldsymbol{x}_{io}$.

We assume that $\Pr(\boldsymbol{x}_{io}, y_i|\theta)$ has a form $\Pr(\boldsymbol{x}_{io}, y_i|\theta) = \Pr(\boldsymbol{x}_{io}) \Pr(y_i|\theta, \boldsymbol{x}_{io})$ and consider regression problems with Gaussian noise:

$$\Pr(\boldsymbol{x}_{io}, y_i|\theta) \sim \Pr(\boldsymbol{x}_{io}) \exp\left(-\frac{(\theta^T \boldsymbol{x}_{io} - y_i)^2}{2\sigma^2}\right),$$
$$\Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}) \sim \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{io}\|^2}{2\sigma_i^2}\right).$$

The method in (2.25) becomes

$$\theta = \arg\min_{\theta} \sum_i \inf_{\boldsymbol{x}_{io}} \left[\frac{(\theta^T \boldsymbol{x}_{io} - y_i)^2}{2\sigma^2} + \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{io}\|^2}{2\sigma_i^2}\right]. \tag{2.26}$$

This formulation is closely related to the total least squares method (Golub et al., 1999; Bi and Zhang, 2005), which is derived from a numerical computation point of view. The resulting formulation of the total least squares algorithm is similar to (2.26), but its solution can be conveniently described by a matrix singular value decomposition. Thus the formulation (2.26) can be regarded as the underlying statistical model for total least squares.

For binary classification, we consider a logistic conditional probability model for y_i, while still assuming Gaussian noise for the input:

$$\Pr(\boldsymbol{x}_{io}, y_i|\theta) \sim \Pr(\boldsymbol{x}_{io}) \frac{1}{1 + \exp(-\theta^T \boldsymbol{x}_{io} y_i)}, \quad \Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}) \sim \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{io}\|^2}{2\sigma_i^2}\right).$$

Similar to (2.26), the following formulation is obtained:

$$\theta = \arg\min_{\theta} \sum_i \inf_{\boldsymbol{x}_{io}} \left[\ln(1 + e^{-\theta^T \boldsymbol{x}_{io} y_i}) + \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{io}\|^2}{2\sigma_i^2}\right]. \tag{2.27}$$

2.2.4 Input Uncertainty Model

For generality and practicality, Gaussian noise is applied to contaminate the inputs throughout the thesis. According to (2.26) and (2.27), we know that the conditional

probability distribution of the corrupted input for a given value of its original counterpart can be presented as

$$\Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io}) \sim \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{io}\|^2}{2\sigma_i^2}\right). \tag{2.28}$$

From (2.28), we see that the conditional distribution is inversely proportional to the quotient of dividing the Euclidean distance between the contaminated input $\boldsymbol{x}_i$ and its original counterpart $\boldsymbol{x}_{io}$ by a variance. Geometrically, $\boldsymbol{x}_i$ has fairish probability of staying within a σ_i-radius sphere with $\boldsymbol{x}_{io}$ being the centre. Additionally, $\boldsymbol{x}_{io}$ can also be deemed as a point that stays within the same σ_i-radius sphere but with $\boldsymbol{x}_i$ appointed the centre. In other words, the distribution of the original input can be estimated through the information of $\boldsymbol{x}_i$ when the conditional probability distribution of $\boldsymbol{x}_i$ is available.

Moreover, the conditional distribution in (2.28) can be extended to accommodate other noise models by simply introducing other noise's distribution assumptions. When no prior assumption of this conditional distribution is available, under the same probability used in Gaussian noise, $\boldsymbol{x}_i$ stays within a sphere well larger than the sphere generated from Gaussian distribution. $\Pr(\boldsymbol{x}_i|\theta', \sigma_i, \boldsymbol{x}_{io})$ can also be derived from the original input $\boldsymbol{x}_{io}$ and the variance σ_i (Please note that the probability of $\boldsymbol{x}_i$ within the same σ_i-radius sphere without prior distribution assumption is lower than the probability with Gaussian prior). Further details about the situation of no prior distribution assumption will be shown in Chapter 4 and Chapter 5 with the introduction of the minimax probability machine (MPM).

Generally, the equidensity contour shape of a Gaussian distribution for any input in a multi-dimensional space is not a prefect sphere, but an ellipsoid centred at mean. Therefore, the one-dimensional Gaussian distribution used in (2.28) should be generalised to a multivariate Gaussian distribution, which is presented as,

$$\Pr(\boldsymbol{x}_i|\theta', \mathbf{M}_i, \boldsymbol{x}_{io}) \sim \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_{io})^T \mathbf{M}_i^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_{io})\right), \tag{2.29}$$

where $\mathbf{M}_i$ is the covariance matrix. The directions of the principal axes of the contour ellipsoid are given by the eigenvectors of the covariance matrix $\mathbf{M}_i$ and the squared relative lengths of the principal axes are given by the corresponding eigenvalues. Similar to the previous discussion in one-dimensional case, the original input $\boldsymbol{x}_{io}$ can also be estimated as an uncertain input $\boldsymbol{z}_i$ that follows a multivariate Gaussian distribution centred at its corrupted counterpart $\boldsymbol{x}_i$, and this multivariate Gaussian distribution can be assumed as $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{x}_i, \mathbf{M}_i)$. A two-dimensional example is shown in Figure 2.7. In the figure, the original input $\boldsymbol{x}_{io}$ is set to move to two possible locations under the contamination. Indeed, all possible locations chosen within the red dashed contour can be the destination of the corrupted input $\boldsymbol{x}_i$ with varied probabilities. The distribution with the same covariance matrix as that of the conditional distribution $\Pr(\boldsymbol{x}_i|\theta', \mathbf{M}_i, \boldsymbol{x}_{io})$

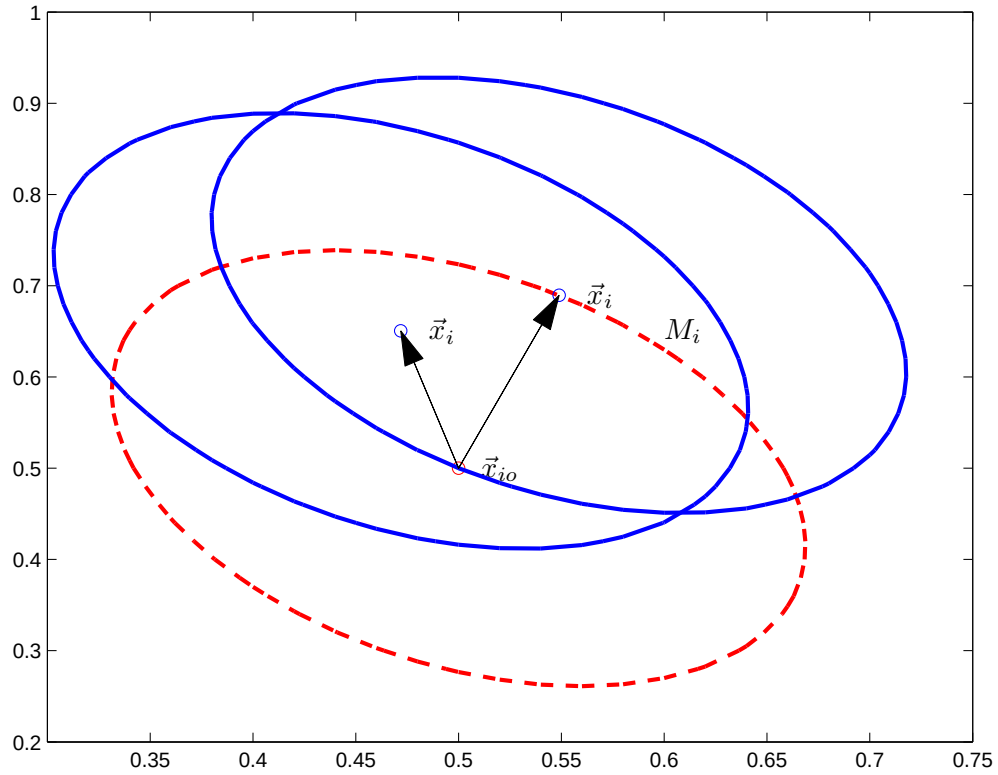


FIGURE 2.7: A two-dimensional example of input uncertainty model, in which $\boldsymbol{x}_{io}$ is the original input and $\boldsymbol{x}_i$ is corrupted input at two possible locations, covariance matrix $\mathbf{M}_i$ represents the uncertainty at $\boldsymbol{x}_{io}$. The red dashed line contours the conditional distribution of $\boldsymbol{x}_i$ for a given value of $\boldsymbol{x}_{io}$, while the blue solid lines contour two possible conditional distributions of $\boldsymbol{x}_{io}$.

is applied to these possible locations to give $\boldsymbol{x}_{io}$'s estimated conditional distributions centred at $\boldsymbol{x}_i$.

2.3 Summary

A main target of machine learning research is to develop algorithms which learn predictive relationships from data. However, the task will become difficult when training data are corrupted by noise or partially unknown under data processing errors. In order to generate varied contamination in classification problems, several basic noise models were first explored in this chapter with only the attributes of inputs being contaminated. Besides the traditional additive noise model, there are in total four types noise models can be classified: the noise model independent of the target function and the distribution of input data, the function-dependent noise model, the distribution-dependent noise model, the noise model dependent on both the target function and the distribution. All of them can output corrupted inputs with given probabilities in accordance with the hyperplane and distribution of the original inputs.

The input uncertainties can then be obtained either by processing the incompletion or by finding out the statistical connection between the original inputs and their corrupted counterparts. Like the Gaussian processes used in output uncertainty prediction, the unknown or missing attributes can be estimated from the existing attributes under the assumption that both observed and missing attributes follow a joint Gaussian distribution. On the other hand, a statistical model is introduced to illustrate and analyse the relationship of the unknown original input, the observed contaminated input and the model parameters. A conditional distribution of the contaminated input for a given value of its original counterpart is then obtained as a Gaussian distribution. As a result, the unknown original input can be estimated as an uncertain input that follows a Gaussian distribution centred at the observed corrupted input. This Gaussian distribution can be easily extended to a more general multivariate case by introducing a more general covariance matrix.

Chapter 3

Uncertainty Support Vector Classification

A disadvantage of logistic model (2.27) proposed in Section 2.2.3 for binary classification is that it does not model deterministic conditional probability very well. This problem can be remedied by using the support vector machine (SVM) formulation, which has attractive intuitive geometric interpretations for linearly separable problems. In this chapter, input uncertainties will be incorporated into this traditional method.

3.1 Incorporating Input Uncertainty in Support Vector Machines

As we have known from Section 2.2.4, the unknown original input $\boldsymbol{x}_{io}$ can be deemed as an uncertain input that follows a multivariate Gaussian distribution centred at its observed corrupted counterpart under the contamination of Gaussian noise. The following definition is then obtained,

Definition 3.1. Let $\mathcal{D} = \{\boldsymbol{z}_i, y_i\}_{i=1}^{l}$ denote the observed data, where $y_i \in \{-1, +1\}$, $\boldsymbol{z}_i \in \mathbb{R}^n$ and $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{x}_i, \mathbf{M}_i)$, in which $\mathcal{N}$ denotes a Gaussian distribution with mean $\boldsymbol{x}_i \in \mathbb{R}^n$ and covariance matrix $\mathbf{M}_i \in \mathbb{R}^{n \times n}$.

Without loss of generality, we can assume that $\boldsymbol{x}_{io}$ can not be observed when it is contaminated by noise and vice versa. Therefore, if the original input is not corrupted by noise, $\boldsymbol{z}_i = \boldsymbol{x}_{io}$; otherwise, $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{x}_i, \mathbf{M}_i)$. Figure 3.1 gives an example of two-dimensional linearly separable classification over uncertain inputs. Here two solutions are listed from many possible linear classifiers that can separate the data, but only one is the optimal solution (the thick solid line in Figure 3.1) that can maximise the margin ρ, which depicts the distance between the optimal classifier and the nearest uncertain inputs of each class.

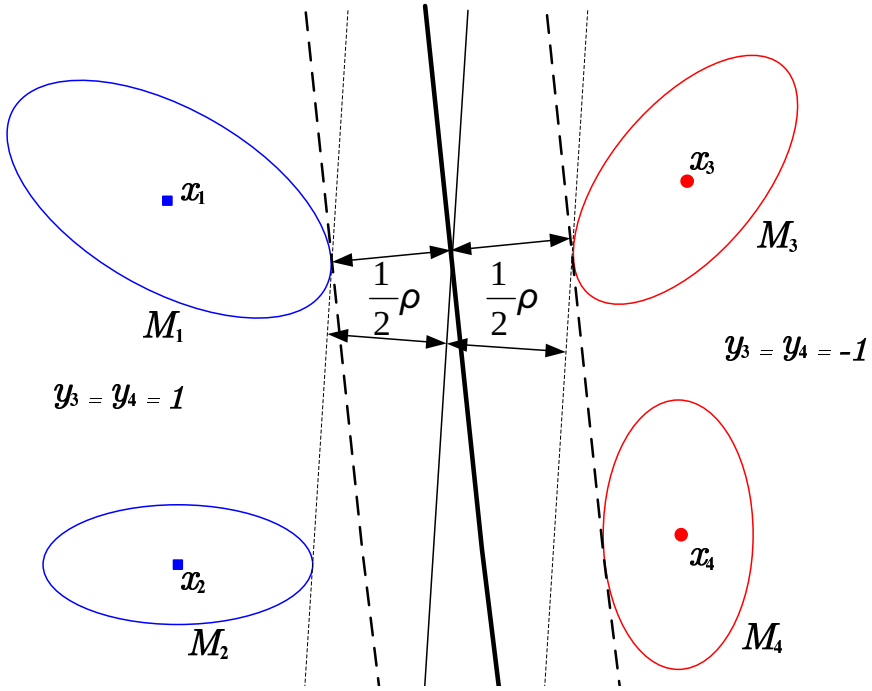


FIGURE 3.1: Two-dimensional linearly separable classification subject to input uncertainty. The elliptical contours represent the uncertain inputs $\boldsymbol{z}_i$ which follow the Gaussian distributions centred at $\boldsymbol{x}_i$. The thin and thick solid lines denote two possible solutions of the classification, the dashed lines describe the loci of the margin.

3.1.1 Geometric Interpretation

Definition 3.1 gives the description of uncertain inputs, whose data structures in classification subject to input uncertainty are similar to those of input data in the standard support vector classification (SVC) except for introducing the covariance matrix $\mathbf{M}_i$ as extra parameters. Like SVC, the optimal classifier here is constructed by the support vectors, whose distribution contours are tangential to the margin. Since the original inputs $\boldsymbol{x}_{io}$ are unknown to the learner, and only their corresponding uncertain inputs $\boldsymbol{z}_i$ are available, the margin of the optimal classifier is geometrically chosen to be tangential to the nearest edges of the distribution contours to make sure that the corresponding uncertain inputs are correctly classified, or the unknown original inputs are likely to be classified correctly. This is illustrated by depicting two support vectors in details in a two-dimensional linearly separable classification shown in Figure 3.2 (Yang and Gunn, 2004), where ρ represents the margin, $\boldsymbol{z}_{\max}$ and $\boldsymbol{z}_{\min}$ denote those points, at which the lines parallel to the optimal classifier are tangential to the ellipses.

The optimisation problem of achieving the maximal ρ is $\max\{\boldsymbol{w}^T\boldsymbol{z}_i|\boldsymbol{z}_i \in \mathcal{E}(\mathbf{M}_i, \boldsymbol{x}_i)\}$, where $\mathcal{E}(\mathbf{M}_i, \boldsymbol{x}_i) \subseteq \mathbb{R}^n$ is an n-dimensional ellipsoid, $\boldsymbol{x}_i \in \mathbb{R}^n$, $\mathbf{M}_i \in \mathbb{R}^{n\times n}$ and $\mathcal{E}(\mathbf{M}_i, \boldsymbol{x}_i) \coloneqq \{\boldsymbol{z}_i \in \mathbb{R}^n \,|\, (\boldsymbol{z}_i - \boldsymbol{x}_i)^T\mathbf{M}_i^{-1}(\boldsymbol{z}_i - \boldsymbol{x}_i) \leq 1\}$.

Theorem 3.2. (Grötschel et al., 1993) *For every positive definite matrix $\boldsymbol{A}$ there exists a unique positive definite matrix, denoted by $\boldsymbol{A}^{1/2}$, such that $\boldsymbol{A} = \boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}$. It follows*

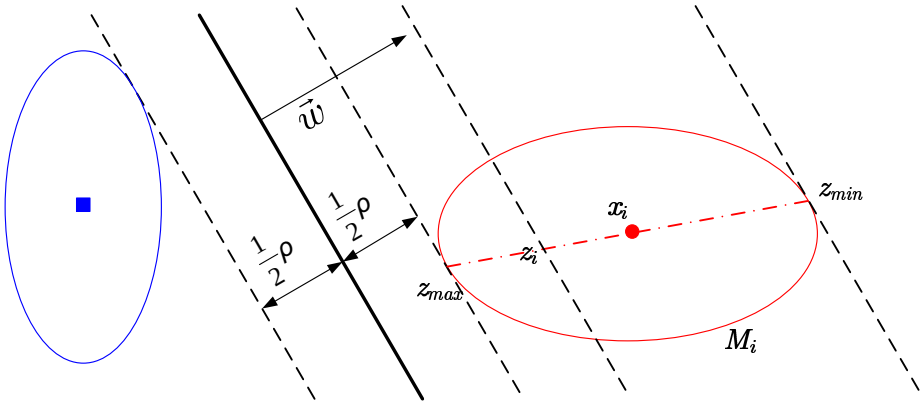


FIGURE 3.2: Geometric interpretation of a two-dimensional classification subject to input uncertainty. The elliptical contours represent the uncertain inputs $\boldsymbol{z}_i$ which follow the Gaussian distributions centred at $\boldsymbol{x}_i$. The solid line denotes the optimal classifier, which is parallel to all dashed lines. The dashed line passing through $\boldsymbol{z}_i$ illustrates how $\boldsymbol{z}_i$ is determined by the weight vector $\boldsymbol{w}$ and the covariance matrix $\mathbf{M}_i$. The dash-dot line illustrates the track on which $\boldsymbol{z}_i$ varies between $\boldsymbol{z}_{\min}$ and $\boldsymbol{z}_{\max}$.

by a simple calculation that

$$\mathcal{E}(\boldsymbol{A}, \boldsymbol{a}) = \boldsymbol{A}^{1/2}\mathcal{S}(0,1) + \boldsymbol{a}, \tag{3.1}$$

where $\mathcal{S}(0,1)$ *is the unit ball around zero and thus every ellipsoid is the image of the unit ball under a bijective affine transformation.*

Proof. $\forall \boldsymbol{x} \in \mathcal{E}(\mathbf{A}, \boldsymbol{a})$, we have $(\boldsymbol{x}-\boldsymbol{a})^T\mathbf{A}^{-1}(\boldsymbol{x}-\boldsymbol{a}) \leq 1$ from the definition of ellipsoid, there exist $\boldsymbol{y} = \mathbf{A}^{-1/2}(\boldsymbol{x}-\boldsymbol{a})$ and $\boldsymbol{y}^T\boldsymbol{y} \leq 1$, $\boldsymbol{y} \in \mathcal{S}(0,1)$, so we have $\mathbf{A}^{-1/2}(\mathcal{E}(\mathbf{A},\boldsymbol{a}) - \boldsymbol{a}) \subseteq \mathcal{S}(0,1)$, contrariwise, we can get $\mathbf{A}^{1/2}\mathcal{S}(0,1) + \boldsymbol{a} \subseteq \mathcal{E}(\mathbf{A},\boldsymbol{a})$, thus $\mathcal{E}(\mathbf{A},\boldsymbol{a}) = \mathbf{A}^{1/2}\mathcal{S}(0,1) + \boldsymbol{a}$ is finally obtained. □

According to the definition and theorem above, set $\mathbf{Q} \coloneqq \mathbf{M}_i^{1/2}$ and recall from (3.1) that $\mathbf{Q}^{-1}\mathcal{E}(\mathbf{M}_i, \boldsymbol{x}_i) = \mathcal{S}(0,1) + \mathbf{Q}^{-1}\boldsymbol{x}_i = \mathcal{S}(\mathbf{Q}^{-1}\boldsymbol{x}_i, 1)$ (Grötschel et al., 1993), we have

$$\begin{aligned}
\max\{\boldsymbol{w}^T\boldsymbol{z}_i \mid \boldsymbol{z}_i \in \mathcal{E}(\mathbf{M}_i, \boldsymbol{x}_i)\} &= \max\{\boldsymbol{w}^T\mathbf{Q}\mathbf{Q}^{-1}\boldsymbol{z}_i \mid \mathbf{Q}^{-1}\boldsymbol{z}_i \in \mathbf{Q}^{-1}\mathcal{E}(\mathbf{M}_i, \boldsymbol{x}_i)\} \\
&= \max\{\boldsymbol{w}^T\mathbf{Q}\boldsymbol{p}_i \mid \boldsymbol{p}_i \in \mathcal{S}(\mathbf{Q}^{-1}\boldsymbol{x}_i, 1)\} \\
&= \boldsymbol{w}^T\mathbf{Q}\frac{1}{\|\mathbf{Q}\boldsymbol{w}\|}\mathbf{Q}\boldsymbol{w} + \boldsymbol{w}^T\mathbf{Q}\mathbf{Q}^{-1}\boldsymbol{x}_i \\
&= \boldsymbol{w}^T\frac{1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\mathbf{M}_i\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{x}_i.
\end{aligned} \tag{3.2}$$

So we have $\boldsymbol{z}_{\max} = \boldsymbol{x}_i - \frac{1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\mathbf{M}_i\boldsymbol{w}$, $\boldsymbol{z}_{\min} = \boldsymbol{x}_i + \frac{1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\mathbf{M}_i\boldsymbol{w}$ and $\boldsymbol{z}_i = \boldsymbol{x}_i - r\frac{1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\mathbf{M}_i\boldsymbol{w}$. z_i can be treated as a point, through which a hyperplane parallel to the optimal hyperplane divides the corresponding ellipsoid into two parts, and the further part to the optimal hyperplane contains less probability of the corresponding unknown original input being correctly classified than $\mathbf{M}_i$ does. $\boldsymbol{z}_i$ moves along the

track determined by the weight vector $\boldsymbol{w}$ and the covariance matrix $\mathbf{M}_i$. Therefore, r is indeed a parameter that tunes $\boldsymbol{z}_i$ between $\boldsymbol{z}_{\min}$ and $\boldsymbol{z}_{\max}$. The larger r is, the higher the probability of the original input being correctly classified can reach. Reintroducing $\boldsymbol{z}_i$ into the constraint of the standard SVC, we have

$$\begin{aligned} y_i(\boldsymbol{w}^T\boldsymbol{z}_i + b) &= y_i\left(\boldsymbol{w}^T\left(\boldsymbol{x}_i - r\frac{1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\mathbf{M}_i\boldsymbol{w}\right) + b\right) \\ &= y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - y_i r\|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \geq 1. \end{aligned} \tag{3.3}$$

Since the inequality in (3.3) is supposed to satisfy any $y_i \in \{-1, 1\}$, formulation (3.3) can be rewritten as:

$$\begin{aligned} \sup_{\boldsymbol{w},b}\left(y_i r\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|\right) &\leq y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \\ r\|\mathbf{M}_i^{1/2}\boldsymbol{w}\| &\leq y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1. \end{aligned} \tag{3.4}$$

3.1.2 Statistical Approach

Alternatively, the Gaussian distribution in Definition 3.1 allows us to derive tighter bounded probabilities of misclassification through statistical approach. The constraint of the standard SVC becomes:

$$\begin{aligned} \inf_{\boldsymbol{z}_i\sim\mathcal{N}(\boldsymbol{x}_i,\mathbf{M}_i)} \Pr\{-y_i\boldsymbol{w}^T\boldsymbol{z}_i \leq y_i b - 1\} &= \Pr\left\{\mathcal{N}(0,1) \geq \frac{-y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + 1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\right\} \\ &= 1 - \Phi\left(\frac{-y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + 1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\right) \\ &= \Phi\left(\frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1}{\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}}\right) \geq \alpha, \end{aligned} \tag{3.5}$$

where α is the lower bound of the probability that the uncertain inputs are correctly classified by the optimal hyperplane, $\Phi(v)$ is the cumulative distribution function for a standard normal Gaussian distribution,

$$\Phi(v) = \Pr\{\mathcal{N}(0,1) \leq v\} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{v}\exp(-s^2/2)ds. \tag{3.6}$$

Since $\Phi(v)$ is monotonically increasing, we can write (3.5) as:

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geq \Phi^{-1}(\alpha)\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}. \tag{3.7}$$

Comparing (3.4) and (3.7), we can set $r = \Phi^{-1}(\alpha)$ to control the probability of misclassification, when the assumption is a Gaussian distribution. Here $r \in \mathbb{R}^n$ is named as the probability confidence. Although the distribution is assumed as a Gaussian distribution

in this thesis, (3.4) provides a feasible way to exploit other distributions of the uncertain inputs when the prior knowledge of those distributions is available.

In classification subject to input uncertainty, classifying the uncertainty ellipsoids as much as possible through the optimal hyperplane is what we expect the learner to finish. In accordance with the geometric interpretation in Figure 3.2, the probability that $\boldsymbol{x}_{io}$ is to be classified correctly by the optimal hyperplane is one-sided probability. If the unknown $\boldsymbol{x}_{io}$ has 95% probability of being correctly classified, the probability confidence $r = 1.645$ in one-dimensional Gaussian distribution. And in multivariate Gaussian distribution, r should be larger than that in one-dimensional Gaussian distribution under the same probability. However, introducing a large r or a large α may be difficult for the learner to classify the uncertain inputs, especially in the non-separable case. On the contrary, $r = 1$ is used to mildly introduce the input uncertainty information into classification. Moreover, $r = 1$ has explicit geometric interpretation illustrated in Figure 3.2. Therefore, instead of $\boldsymbol{z}_i$, the obtained optimal points $\boldsymbol{z}_{\max}$ is introduced into the constraints of the standard SVC. We have

$$\|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \leq y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1, \qquad i = 1, \ldots, l. \tag{3.8}$$

Like SVC, this optimal hyperplane is given by maximising the margin ρ, subject to the constraints of (3.8). The margin is given by,

$$\begin{aligned}\rho(\boldsymbol{w}, b) &= \min_{y_i=-1} \frac{\left|\boldsymbol{w}^T\boldsymbol{x}_i + \sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}} + b\right|}{\|\boldsymbol{w}\|} + \min_{y_i=1} \frac{\left|\boldsymbol{w}^T\boldsymbol{x}_i + \sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}} + b\right|}{\|\boldsymbol{w}\|} \\ &= \frac{2}{\|\boldsymbol{w}\|},\end{aligned} \tag{3.9}$$

and the primal problem of this approach is

$$\begin{aligned}&\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|^2}{2} \\ &s.t.\ \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \leq y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1, \qquad i = 1, \ldots, l.\end{aligned} \tag{3.10}$$

3.1.3 Second Order Cone Program

Second order cone programming (SOCP) problems are convex optimisation problems in which a linear function in minimised over the intersection of an affine linear manifold with the Cartesian product of second order cones. Standard SOCP problem is shown as follows (Boyd and Vandenberghe, 2004),

$$\begin{aligned}&\min_{\boldsymbol{x}} \ \boldsymbol{f}^T\boldsymbol{x} \\ &s.t.\ \|\mathbf{A}_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^T\boldsymbol{x} + d_i, \qquad i = 1, \ldots, m \\ &\quad \mathbf{F}\boldsymbol{x} = \boldsymbol{g},\end{aligned} \tag{3.11}$$

where

$$\mathcal{C}_{k_i} = \left\{ \left[\begin{array}{c} \boldsymbol{u} \\ t \end{array} \right] \middle| \boldsymbol{u} \in \mathbb{R}^{k_i-1}, t \in \mathbb{R}, \|\boldsymbol{u}\| \leq t \right\},$$

is standard k_i-dimensional second order cone and

$$\|\mathbf{A}_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^T\boldsymbol{x} + d_i \Longleftrightarrow \begin{bmatrix} \mathbf{A}_i \\ \boldsymbol{c}_i^T \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} \boldsymbol{b}_i \\ d_i \end{bmatrix} \in \mathcal{C}_{k_i},$$

are second order cone constraints of dimension k_i, which are the same as requiring the affine functions $(\mathbf{A}_i\boldsymbol{x} + \boldsymbol{b}_i, \boldsymbol{c}_i^T\boldsymbol{x} + d_i)$ to lie in the second order cone $\mathcal{C}_{k_i}$. $\boldsymbol{x} \in \mathbb{R}^n$ is the optimisation variable, $\mathbf{A}_i \in \mathbb{R}^{(k_i-1)\times n}$, $\boldsymbol{b}_i \in \mathbb{R}^{k_i-1}$, $\boldsymbol{c}_i \in \mathbb{R}^n$, $d \in \mathbb{R}$ and $\mathbf{F} \in \mathbb{R}^{p\times n}$, $\boldsymbol{g} \in \mathbb{R}^p$.

SOCP problems also include linear programs (LP), quadratic programs (QP) and quadratically constrained quadratic programs (QCQP) as special cases, which can all be formulated as SOCP problems. When $\boldsymbol{c}_i = \mathbf{0}, i = 1, \ldots, m$, the problem (3.11) is equivalent to a QCQP by squaring each of the constraints. And QP is a special case of QCQP. Similarly, if $\mathbf{A}_i = \mathbf{0}, i = 1, \ldots, m$, (3.11) reduces to a (general) LP. (Boyd and Vandenberghe, 2004).

3.2 Dual Problem

In the last section, we have seen how the introduction of uncertain inputs into the SVM technique can be achieved. The resulting optimisation problem is shown to be a convex SOCP. We name this algorithm uncertainty support vector classification (USVC). Comparing the form of the standard SVC with (3.10), it can be found that the ellipsoid matrices $\mathbf{M}_i,\ i = 1, \ldots, l$ are introduced to formulate the margin of classification subject to input uncertainty, while the margin of the standard SVC is fixed by the data points, which is the reason that USVC can accommodate the input uncertainties. This difference will cause a higher computational cost for USVC than SVC, especially with a large training set. The reason for the different margins in USVC and SVC will be discovered by deriving the dual problem of USVC. In the next two sections, the Lagrangian dual method will be used to obtain the dual problem of (3.10), along with a kernelised version.

3.2.1 Dual Problem for the Linearly Separable Case

Like the standard SVC, after the derivation of the dual problem of USVC, the dual variables can be directly used to obtain the parameters of the optimal hyperplane. Moreover, the dual problem can help us extend USVC to the non-linear case. In order to follow the definition of SOCP in (3.11), an auxiliary parameter t is introduced into the optimisation problem to transform formula (3.10) to the standard SOCP form. Let $\frac{\|\boldsymbol{w}\|^2}{2} \leq t$,

which then can be transformed to standard SOCP form

$$\left\| \begin{bmatrix} \boldsymbol{w} \\ \frac{t-1}{\sqrt{2}} \end{bmatrix} \right\| \leq \frac{t+1}{\sqrt{2}}.$$

Therefore, we have

$$\begin{aligned} &\min_{t,\boldsymbol{w},b} \ t \\ &s.t. \ \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \leq y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1, \\ &\left\| \begin{bmatrix} \boldsymbol{w} \\ \frac{t-1}{\sqrt{2}} \end{bmatrix} \right\| \leq \frac{t+1}{\sqrt{2}}. \end{aligned} \tag{3.12}$$

An auxiliary parameter $\boldsymbol{w}_1$ is introduced to simplify (3.10) further, by letting $\boldsymbol{w}_1 = \begin{bmatrix} t & \boldsymbol{w}^T & b \end{bmatrix}^T \in \mathbb{R}^{n+2}$, thus we obtain:

$$\begin{aligned} &\min_{\boldsymbol{w}_1} \ \boldsymbol{f}^T\boldsymbol{w}_1 \\ &s.t. \ \|\mathbf{A}_i\boldsymbol{w}_1 + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^T\boldsymbol{w}_1 + d_i, \qquad i = 1, \ldots, l+1, \end{aligned} \tag{3.13}$$

where l is the size of the input data set and the other parameters in (3.13) are listed as follows:

$$\boldsymbol{f} = \begin{bmatrix} 1 & \mathbf{0}^T & 0 \end{bmatrix}^T, \qquad i = 1, \ldots, l,$$

$$\mathbf{A}_i = \left[\begin{array}{c|c|c} 0 & & 0 \\ \vdots & \mathbf{M}_i^{1/2} & \vdots \\ 0 & & 0 \end{array}\right], \qquad \boldsymbol{b}_i = \mathbf{0}, \quad \boldsymbol{c}_i = \left[\begin{array}{c} 0 \\ \hline y_i\boldsymbol{x}_i \\ \hline y_i \end{array}\right], \qquad d_i = -1,$$

$$\mathbf{A}_{l+1} = \left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \mathbf{0}^T & 0 \\ \hline 0 & & 0 \\ \vdots & \mathbf{I} & \vdots \\ 0 & & 0 \end{array}\right], \qquad \boldsymbol{b}_{l+1} = \left[\begin{array}{c} -\frac{1}{\sqrt{2}} \\ \hline \mathbf{0} \end{array}\right], \quad \boldsymbol{c}_{l+1} = \left[\begin{array}{c} \frac{1}{\sqrt{2}} \\ \hline \mathbf{0} \\ \hline 0 \end{array}\right], \qquad d_{l+1} = \frac{1}{\sqrt{2}},$$

where $\mathbf{A}_i \in \mathbb{R}^{n\times(n+2)}$, $\boldsymbol{b}_i \in \mathbb{R}^n$, $\boldsymbol{c}_i \in \mathbb{R}^{n+2}$, $d_i \in \mathbb{R}$, $i = 1, \ldots, n$, $\mathbf{A}_{l+1} \in \mathbb{R}^{(n+1)\times(n+2)}$, $\boldsymbol{b}_{l+1} \in \mathbb{R}^{n+1}$, $\boldsymbol{c}_{l+1} \in \mathbb{R}^{n+2}$, $d_{l+1} \in \mathbb{R}$, and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix. In order to derive the dual problem, extra auxiliary parameters are introduced and set as follows,

$$\begin{aligned} \boldsymbol{u}_i &= \mathbf{A}_i\boldsymbol{w}_1 + \boldsymbol{b}_i, \\ t_i &= \boldsymbol{c}_i^T\boldsymbol{w}_1 + d_i, \\ t_i &\geq \|\boldsymbol{u}_i\|, \qquad i = 1, \ldots, l+1. \end{aligned}$$

The Lagrangian multipliers used in the dual problem include α_i, $\boldsymbol{\beta}_i$ and γ_i, while $\alpha_i, \gamma_i \in \mathbb{R}$ and $\boldsymbol{\beta}_i \in \mathbb{R}^n$, $i = 1, \ldots, l$, and $\alpha_{l+1} \in \mathbb{R}$, $\boldsymbol{\beta}_{l+1} \in \mathbb{R}^{n+1}$ are derived from the auxiliary

variable t introduced in (3.12), so the dual problem of (3.12) is written as follows:

$$\begin{aligned} \max_{\alpha_i,\boldsymbol{\beta}_i} \quad & -\sum_{i=1}^{l+1}(\boldsymbol{b}_i^T\boldsymbol{\beta}_i + d_i\alpha_i) \\ s.t. \quad & \sum_{i=1}^{l+1}(\mathbf{A}_i^T\boldsymbol{\beta}_i + \alpha_i\boldsymbol{c}_i) = \boldsymbol{f}, \\ & \|\boldsymbol{\beta}_i\| \le \alpha_i, \qquad i = 1,\ldots,l+1. \end{aligned} \tag{3.14}$$

If we introduce $\mathbf{A}_i$, $\boldsymbol{b}_i$, $\boldsymbol{c}_i$, d_i, $i = 1,\ldots,l+1$ and $\boldsymbol{f}$ from formula (3.13) into (3.14) and set $\boldsymbol{\beta}_{l+1} = \left[\beta' \,\vdots\, \boldsymbol{\beta}'_{l+1}\right]^T$, the constraint function $\sum_{i=1}^{l+1}(\mathbf{A}_i^T\boldsymbol{\beta}_i + \alpha_i\boldsymbol{c}_i) = \boldsymbol{f}$ can be expanded as:

$$\begin{bmatrix} 0 \\ \hdashline \vdots \\ \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta}_i \\ \vdots \\ \hdashline 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \hdashline \vdots \\ \sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i \\ \vdots \\ \hdashline \sum_{i=1}^{l}\alpha_i y_i \end{bmatrix} + \begin{bmatrix} \frac{1}{\sqrt{2}}\beta' \\ \hdashline \vdots \\ \boldsymbol{\beta}'_{l+1} \\ \vdots \\ \hdashline 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{\sqrt{2}}\alpha_{l+1} \\ \hdashline \vdots \\ \mathbf{0} \\ \vdots \\ \hdashline 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \hdashline \vdots \\ \mathbf{0} \\ \vdots \\ \hdashline 0 \end{bmatrix}.$$

Thus, the dual problem of the linearly separable case can be obtained as:

$$\begin{aligned} \max_{\alpha_i,\boldsymbol{\beta}_i} \quad & \sum_{i=1}^{l}\alpha_i + \frac{1}{\sqrt{2}}\beta' - \frac{1}{\sqrt{2}}\alpha_{l+1} \\ s.t. \quad & \|\boldsymbol{\beta}_i\| \le \alpha_i, \qquad i = 1,\ldots,l+1, \\ & \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta}_i + \sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i + \boldsymbol{\beta}'_{l+1} = \mathbf{0}, \\ & \sum_{i=1}^{l}\alpha_i y_i = 0, \\ & \frac{1}{\sqrt{2}}\beta' + \frac{1}{\sqrt{2}}\alpha_{l+1} = 1. \end{aligned} \tag{3.15}$$

In order to compare this with the dual form of the traditional SVC, extra work is still needed to transform the dual optimisation problem of USVC. Here, t in the primal problem is an auxiliary parameter, consequently, the α_{l+1} and $\boldsymbol{\beta}_{l+1}$ are auxiliary parameters in the dual problem as well. These auxiliary parameters can be removed (see Appendix A). After combining the constraints of (3.15), the exact expression of $\boldsymbol{w}$ can be obtained as follows,

$$\boldsymbol{w} = -\boldsymbol{\beta}_{l+1} = \sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta}_i. \tag{3.16}$$

Comparing (3.16) with the expression of $\boldsymbol{w}$ from the standard SVC, the uncertainties are introduced into the expression of $\boldsymbol{w}$ of USVC. A new parameter $\mathbf{M}_i$ and a new

variable $\boldsymbol{\beta_i}$ have been used to introduce the influence of the sizes and directions of the uncertainty ellipsoids in the data set, with α_i behaving in a similar manner to SVC, to introduce the influence of the positions of the data points. Reintroducing (3.16) and the constraints of (3.15) back to the objective function of (3.15), we have

$$\begin{aligned}
\max_{\alpha_i,\boldsymbol{\beta}_i} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i \boldsymbol{x}_i^T(\mathbf{M}_j^{1/2})^T\boldsymbol{\beta_j}\right. \\
& \left. + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_j y_j \boldsymbol{\beta_i}^T\mathbf{M}_i^{1/2}\boldsymbol{x}_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\boldsymbol{\beta_i}^T\mathbf{M}_i^{1/2}(\mathbf{M}_j^{1/2})^T\boldsymbol{\beta_j}\right) \\
s.t. \quad & \|\boldsymbol{\beta_i}\| \le \alpha_i, \qquad i = 1,\dots,l \\
& \sum_{i=1}^{l}\alpha_i y_i = 0 \\
& \alpha_i \ge 0, \qquad i = 1,\dots,l,
\end{aligned} \tag{3.17}$$

where $\alpha_1, \alpha_2, \dots, \alpha_l \in \mathbb{R}$ and $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \dots, \boldsymbol{\beta}_l \in \mathbb{R}^n$. The difference of the dual problems between USVC and SVC is the introduction of uncertainty ellipsoids, which leads to a higher algorithmic complexity in USVC than in SVC because of the increase in memory requirements and computational cost of the algorithm. Following Definition 3.1, with l input data and n features, the algorithmic complexities can be derived from (3.17). The objective function of (3.17) can be transformed to (3.18).

Hence, the memory requirement of USVC is $l^2(n+1)^2$, which is of order $O\left(l^2n^2\right)$. The computational cost of USVC is dominated by two aspects, the cost of constructing matrix $\mathbf{R}$ in the objective function and the optimisation complexity of SOCP solver. The latter is determined by the optimisation implementation and is shown to be bounded by $O\left(l^{3/2}n^3\right)$ in Section 3.4.1. The former is related to the amount of multiplication required in constructing $\mathbf{R}$, which is a symmetric matrix. Therefore, the cost of constructing $\mathbf{R}$ is $\frac{l(l+1)}{2}n + l^2n^2 + \frac{l(l+1)}{2}n^3$, which is of order $O\left(l^2n^3\right)$ because this cost is determined by the lower symmetric sub-matrix consisting of $\mathbf{M}_i^{1/2}(\mathbf{M}_i^{1/2})^T$.

Therefore, the overall complexity of USVC is $O\left(l^2n^3\right)$, which may cost the learner a lot to solve some non-trivial problems. However, there are some situations where only a few points are known to be contaminated. Consequently, the complexity of USVC can be significantly reduced. For instance, let l_m denote the number of contaminated inputs, so the other $l - l_m$ inputs are uncorrupted. Then the memory requirement of USVC is of order $O\left((l + l_m n)^2\right)$, the optimisation complexity, which is determined by the number of second order cones, is $O\left(l_m^{3/2}n^3\right)$, and the cost of constructing $\mathbf{R}$ is of order $O\left(l_m^2n^3\right)$.

$$\max_{\boldsymbol{\gamma}} \quad \boldsymbol{p}^T\boldsymbol{\gamma} - \boldsymbol{\gamma}^T\mathbf{R}\boldsymbol{\gamma} \qquad \text{where} \quad \boldsymbol{p} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \hdashline \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{\gamma} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_l \\ \hdashline \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_l \end{bmatrix}, \tag{3.18}$$

$$\mathbf{R} = \left[\begin{array}{cccc:ccc:c:c} y_1y_1\boldsymbol{x}_1^T\boldsymbol{x}_1 & y_1y_2\boldsymbol{x}_1^T\boldsymbol{x}_2 & \dots & y_1y_l\boldsymbol{x}_1^T\boldsymbol{x}_l & y_1\boldsymbol{x}_1^T(\mathbf{M}_1^{1/2})^T & y_1\boldsymbol{x}_1^T(\mathbf{M}_2^{1/2})^T & \dots & y_1\boldsymbol{x}_1^T(\mathbf{M}_l^{1/2})^T \\ y_2y_1\boldsymbol{x}_2^T\boldsymbol{x}_1 & y_2y_2\boldsymbol{x}_2^T\boldsymbol{x}_2 & \dots & y_2y_l\boldsymbol{x}_2^T\boldsymbol{x}_l & y_2\boldsymbol{x}_2^T(\mathbf{M}_1^{1/2})^T & y_2\boldsymbol{x}_2^T(\mathbf{M}_2^{1/2})^T & \dots & y_2\boldsymbol{x}_2^T(\mathbf{M}_l^{1/2})^T \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_ly_1\boldsymbol{x}_l^T\boldsymbol{x}_1 & y_ly_2\boldsymbol{x}_l^T\boldsymbol{x}_2 & \dots & y_ly_l\boldsymbol{x}_l^T\boldsymbol{x}_l & y_l\boldsymbol{x}_l^T(\mathbf{M}_1^{1/2})^T & y_l\boldsymbol{x}_l^T(\mathbf{M}_2^{1/2})^T & \dots & y_l\boldsymbol{x}_l^T(\mathbf{M}_l^{1/2})^T \\ \hdashline y_1\mathbf{M}_1^{1/2}\boldsymbol{x}_1 & y_2\mathbf{M}_1^{1/2}\boldsymbol{x}_2 & \dots & y_l\mathbf{M}_1^{1/2}\boldsymbol{x}_l & \mathbf{M}_1^{1/2}(\mathbf{M}_1^{1/2})^T & \mathbf{M}_1^{1/2}(\mathbf{M}_2^{1/2})^T & \dots & \mathbf{M}_1^{1/2}(\mathbf{M}_l^{1/2})^T \\ \hdashline y_1\mathbf{M}_2^{1/2}\boldsymbol{x}_1 & y_2\mathbf{M}_2^{1/2}\boldsymbol{x}_2 & \dots & y_l\mathbf{M}_2^{1/2}\boldsymbol{x}_l & \mathbf{M}_2^{1/2}(\mathbf{M}_1^{1/2})^T & \mathbf{M}_2^{1/2}(\mathbf{M}_2^{1/2})^T & \dots & \mathbf{M}_2^{1/2}(\mathbf{M}_l^{1/2})^T \\ \hdashline \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hdashline y_1\mathbf{M}_l^{1/2}\boldsymbol{x}_1 & y_2\mathbf{M}_l^{1/2}\boldsymbol{x}_2 & \dots & y_l\mathbf{M}_l^{1/2}\boldsymbol{x}_l & \mathbf{M}_l^{1/2}(\mathbf{M}_1^{1/2})^T & \mathbf{M}_l^{1/2}(\mathbf{M}_2^{1/2})^T & \dots & \mathbf{M}_l^{1/2}(\mathbf{M}_l^{1/2})^T \end{array}\right],$$

Furthermore, when no inputs are contaminated by noise, the uncertain information is no longer available, then all $\mathbf{M}_i$ are zero matrices, USVC will degenerate to SVC, with

$$\begin{aligned}
\max_{\boldsymbol{\gamma}} \quad & \boldsymbol{p}^T\boldsymbol{\gamma} - \boldsymbol{\gamma}^T\mathbf{R}\boldsymbol{\gamma} \\
\text{where} \quad & \boldsymbol{p} = [11\ldots1]^T, \\
& \boldsymbol{\gamma} = \left[\alpha_1\alpha_2\ldots\alpha_l\right]^T, \\
& \mathbf{R} = \begin{bmatrix} y_1y_1\boldsymbol{x}_1^T\boldsymbol{x}_1 & y_1y_2\boldsymbol{x}_1^T\boldsymbol{x}_2 & \ldots & y_1y_l\boldsymbol{x}_1^T\boldsymbol{x}_l \\ y_2y_1\boldsymbol{x}_2^T\boldsymbol{x}_1 & y_2y_2\boldsymbol{x}_2^T\boldsymbol{x}_2 & \ldots & y_2y_l\boldsymbol{x}_2^T\boldsymbol{x}_l \\ \vdots & \vdots & \ddots & \vdots \\ y_ly_1\boldsymbol{x}_l^T\boldsymbol{x}_1 & y_ly_2\boldsymbol{x}_l^T\boldsymbol{x}_2 & \ldots & y_ly_l\boldsymbol{x}_l^T\boldsymbol{x}_l \end{bmatrix}.
\end{aligned}$$

Therefore, the memory requirement of SVC is of order $O\left(l^2\right)$, the cost of constructing the symmetric matrix $\mathbf{R}$ has order $O\left(l^2n\right)$, and the optimisation problem is a quadratic program with corresponding optimisation complexity of $O\left(l^{3/2}\right)$.

3.2.2 Dual Problem for the Linearly Non-Separable Case

USVC can be extended to the linearly non-separable case in the same manner as SVC by introducing the penalty parameters ξ_i. The primal optimisation problem now becomes:

$$\begin{aligned}
\min_{t,\boldsymbol{w},b,\xi_i} \quad & t + C\sum_{i=1}^{l}\xi_i \\
s.t. \quad & \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \le y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i, \\
& \left\| \begin{bmatrix} \boldsymbol{w} \\ \frac{t-1}{\sqrt{2}} \end{bmatrix} \right\| \le \frac{t+1}{\sqrt{2}}, \\
& \xi_i \ge 0, \qquad i = 1,\ldots,l.
\end{aligned} \tag{3.19}$$

The dual form of the linearly non-separable case is given by

$$\begin{aligned}
\max_{\alpha_i,\boldsymbol{\beta}_i} \quad & \sum_{i=1}^{l}\alpha_i + \frac{1}{\sqrt{2}}\beta' - \frac{1}{\sqrt{2}}\alpha_{l+1} \\
s.t. \quad & \|\boldsymbol{\beta_i}\| \le \alpha_i, \qquad i = 1,\ldots,l+1, \\
& \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta_i} + \sum_{i=1}^{l}\alpha_i y_i\boldsymbol{x}_i + \boldsymbol{\beta'_{l+1}} = \mathbf{0}, \\
& \sum_{i=1}^{l}\alpha_i y_i = 0, \\
& \frac{1}{\sqrt{2}}\beta' + \frac{1}{\sqrt{2}}\alpha_{l+1} = 1, \\
& 0 \le \alpha_i \le C, \qquad i = 1,\ldots,l.
\end{aligned} \tag{3.20}$$

Similar to the separable case, (3.20) can be rewritten in a more familiar form,

$$\begin{aligned}
\max_{\alpha_i,\boldsymbol{\beta}_i} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i \boldsymbol{x}_i^T(\mathbf{M}_j^{1/2})^T\boldsymbol{\beta_j}\right. \\
& \left. + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_j y_j \boldsymbol{\beta_i}^T\mathbf{M}_i^{1/2}\boldsymbol{x}_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\boldsymbol{\beta_i}^T\mathbf{M}_i^{1/2}(\mathbf{M}_j^{1/2})^T\boldsymbol{\beta_j}\right) \\
s.t. \quad & \|\boldsymbol{\beta_i}\| \le \alpha_i, \qquad i = 1,\ldots,l, \\
& \sum_{i=1}^{l}\alpha_i y_i = 0, \\
& 0 \le \alpha_i \le C, \qquad i = 1,\ldots,l.
\end{aligned} \tag{3.21}$$

The algorithmic complexity of (3.21) is the same as that of (3.17).

3.3 Extension to Non-Linear Models

Often, data to be classified will require a non-linear separation. We now consider extending the approach to this non-linear scenario. The theory of kernels was first developed by Mercer (1909), which states that any continuous, symmetric, positive semidefinite function $K(x, y)$ can be expressed as a dot product in a high-dimensional space. Another ancestral field of the kernel approach is called "Reproducing kernel Hilbert space theory", which is a subfield of Hilbert space theory and was developed by Aronszajn (1950). Mercer's theorem for interpreting kernels was introduced into machine learning by Aizermann et al. (1964) on the method of potential functions, but its possibilities were not widely understood until it was first used in the support vector method (Boser et al., 1992). The idea of the kernel method is to use a linear classifier algorithm to solve a non-linear problem by mapping the original non-linear inputs in the input space into a higher-dimensional feature space. Although kernel method makes a linear classification in the feature space equivalent to non-linear classification in the original input space, operations are performed in the input space rather than the potentially higher-dimensional feature space. Hence, the dot product does not need to be evaluated in the feature space. When mapping the data of input space $\mathbb{R}^n$ to the higher-dimensional Euclidean space, the feature space $\mathbb{R}^m$, a mapping function ϕ is used.

$$\phi: \ \mathbb{R}^n \mapsto \mathbb{R}^m.$$

So $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$ of the input space are mapped to $\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \in \mathbb{R}^m$ of the feature space. If the decision function would only depend on an inner product, which is called the kernel function K, then no explicit computation of $\phi(\mathbf{x})$ is required.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j).$$

Although the input $\boldsymbol{x}_i$ of the input space can be directly mapped to $\phi(\boldsymbol{x}_i)$ of the feature space by the mapping function ϕ, it is difficult for ϕ to map the uncertainty ellipsoid $\mathbf{M}_i$ to the feature space because the mapped region may correspond to an irregular shape in the feature space. Thus an approximation strategy is proposed to try and find an ellipsoid in the feature space that corresponds as closely as possible to the non-linear projector of an ellipsoid in the input space performed using ϕ.

A Taylor series expansion is then introduced from Graepel and Herbrich (2003) in order to find the connection of mapping input uncertainty between the input space and the feature space. The Taylor series can be expanded at the training input $\boldsymbol{x}_i$ in the input space, and $\boldsymbol{x}_j - \boldsymbol{x}_i$ is an infinitesimal vector between $\boldsymbol{x}_i$ and its neighbour $\boldsymbol{x}_j$. Let

$$\phi(\boldsymbol{x}_i) = \boldsymbol{z}_i = \begin{bmatrix} z_{i1} \\ \vdots \\ z_{im} \end{bmatrix} \qquad \boldsymbol{x}_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{in} \end{bmatrix},$$
$$\phi(\boldsymbol{x}_j) = \boldsymbol{z}_j = \begin{bmatrix} z_{j1} \\ \vdots \\ z_{jm} \end{bmatrix} \qquad \boldsymbol{x}_j = \begin{bmatrix} x_{j1} \\ \vdots \\ x_{jn} \end{bmatrix},$$

we assume when an infinitesimal vector $\triangle\boldsymbol{x}_j = \boldsymbol{x}_j - \boldsymbol{x}_i$ be continuous in the input space, its mapping infinitesimal vector $\triangle\boldsymbol{z}_j = \boldsymbol{z}_j - \boldsymbol{z}_i = \phi(\boldsymbol{x}_j) - \phi(\boldsymbol{x}_i)$ must be continuous in the feature space. And all partial derivatives must be evaluated at $\boldsymbol{x}_i$. The Taylor expansion at $\boldsymbol{x}_i$ is shown as follows,

$$\begin{aligned}
\begin{bmatrix} z_{j1} \\ \vdots \\ z_{jm} \end{bmatrix} &= \begin{bmatrix} z_{i1} \\ \vdots \\ z_{im} \end{bmatrix} + \begin{bmatrix} \frac{\partial z_1}{\partial x_1}(x_{j1} - x_{i1}) & + \ \ldots \ + & \frac{\partial z_1}{\partial x_n}(x_{jn} - x_{in}) \\ \vdots & + \ \vdots \ + & \vdots \\ \frac{\partial z_m}{\partial x_1}(x_{j1} - x_{i1}) & + \ \ldots \ + & \frac{\partial z_m}{\partial x_n}(x_{jn} - x_{in}) \end{bmatrix} \\
&\quad + O\left(\frac{1}{2}\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{x}^2}(\boldsymbol{x}_j - \boldsymbol{x}_i) + \ldots\right) \\
\begin{bmatrix} z_{j1} \\ \vdots \\ z_{jm} \end{bmatrix} &= \begin{bmatrix} z_{i1} \\ \vdots \\ z_{im} \end{bmatrix} + \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_m}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} x_{j1} - x_{i1} \\ \vdots \\ x_{jn} - x_{in} \end{bmatrix} + O\left(\frac{1}{2}\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{x}^2}(\boldsymbol{x}_j - \boldsymbol{x}_i) + \ldots\right) \\
\phi(\boldsymbol{x}_j) &= \phi(\boldsymbol{x}_i) + \mathbf{J}(\boldsymbol{x}_j - \boldsymbol{x}_i) + O\left(\frac{1}{2}\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{x}^2}(\boldsymbol{x}_j - \boldsymbol{x}_i) + \ldots\right),
\end{aligned} \tag{3.22}$$

where $\mathbf{J}$ is the Jacobian matrix made up of the first order partial derivatives and

$$\mathbf{J} = \frac{\partial \phi(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_m}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix}.$$

The Taylor expansion at $\boldsymbol{x}_i$ can be approximated by truncating the second and higher order partial derivatives,

$$\triangle\phi(\boldsymbol{x}_j) = \mathbf{J}\triangle\boldsymbol{x}_j. \tag{3.23}$$

Furthermore, the expression (3.23) can be extended to accommodate the geometric polygonal mapping between the input space and the feature space,

$$\begin{aligned}
\begin{bmatrix} \triangle\phi(\boldsymbol{x}_1^T) \\ \vdots \\ \triangle\phi(\boldsymbol{x}_l^T) \end{bmatrix} &= \begin{bmatrix} \triangle\boldsymbol{x}_1^T \\ \vdots \\ \triangle\boldsymbol{x}_l^T \end{bmatrix} \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_1}{\partial x_n} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix} \\
\begin{bmatrix} \triangle\phi(\boldsymbol{x}_1^T) \\ \vdots \\ \triangle\phi(\boldsymbol{x}_l^T) \end{bmatrix} &= \begin{bmatrix} \triangle\boldsymbol{x}_1^T \\ \vdots \\ \triangle\boldsymbol{x}_l^T \end{bmatrix} \mathbf{J}^T,
\end{aligned} \tag{3.24}$$

where $\triangle\phi(\boldsymbol{x}_i) \in \mathbb{R}^m$ and $\triangle\boldsymbol{x}_i \in \mathbb{R}^n$. The uncertainty ellipsoid matrix $\mathbf{M}_i$ is a description of the uncertainty in the input space, from Definition 3.1 and (3.2), we realise that $\mathbf{M}_i$ represents the covariance matrix of a multivariate Gaussian distribution centred at $\boldsymbol{x}_i$ and the infinitesimal vector $\triangle\boldsymbol{x}_i$ in the input space is related to $O\left(\mathbf{M}_i^{1/2}\right)$, which can be represented as

$$\triangle\boldsymbol{x}_i\triangle\boldsymbol{x}_i^T \propto \mathbf{M}_i = (\mathbf{M}_i^{1/2})^T\mathbf{M}_i^{1/2}, \qquad i = 1, \ldots, l. \tag{3.25}$$

Therefore, the related geometric mapping of $\mathbf{M}_i$ in feature space can be formed by the Jacobian matrix and its counterpart in input space in accordance with the derivation result of (3.24),

$$\phi(\mathbf{M}_i^{1/2}) = \mathbf{M}_i^{1/2}\mathbf{J}^T. \tag{3.26}$$

Moreover, according to the definition of the kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$, $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ can be seen as independent functions during the derivatives over kernel functions because of their different function variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, so the first and second derivatives of kernel function can be retrieved by the inner product of the mapping function ϕ and its derivative respectively. Therefore, we have

$$\frac{\partial\Phi(\boldsymbol{x}_i)}{\partial\boldsymbol{x}_i} \cdot \Phi(\boldsymbol{x}_j) = \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial\boldsymbol{x}_i}. \tag{3.27}$$

$$\Phi(\boldsymbol{x}_i) \cdot \frac{\partial\Phi(\boldsymbol{x}_j)}{\partial\boldsymbol{x}_j} = \left[\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial\boldsymbol{x}_j}\right]^T. \tag{3.28}$$

$$\frac{\partial\Phi(\boldsymbol{x}_i)}{\partial\boldsymbol{x}_i} \cdot \frac{\partial\Phi(\boldsymbol{x}_j)}{\partial\boldsymbol{x}_j} = \frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial\boldsymbol{x}_i\partial\boldsymbol{x}_j}. \tag{3.29}$$

In the general case, the optimisation expression of USVC is as follows:

$$\begin{aligned}
\max_{\alpha_i,\boldsymbol{\beta}_i} \ & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)\right. \\
& + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i \left[\frac{\partial K(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\partial \boldsymbol{x}_j}\right]^T (\mathbf{M}_j^{1/2})^T \boldsymbol{\beta_j} \\
& + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_j y_j \boldsymbol{\beta_i}^T \mathbf{M}_i^{1/2}\frac{\partial K(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} \\
& \left. + \sum_{i=1}^{l}\sum_{j=1}^{l}\boldsymbol{\beta_i}^T \mathbf{M}_i^{1/2}\frac{\partial^2 K(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}(\boldsymbol{M}_j^{1/2})^T \boldsymbol{\beta_j}\right) \\
s.t. \ & \|\boldsymbol{\beta_i}\| \le \alpha_i \qquad i = 1,\ldots,l, \\
& \sum_{i=1}^{l}\alpha_i y_i = 0, \\
& 0 \le \alpha_i \le C \qquad i = 1,\ldots,l. \qquad (\textbf{Non-separable case})
\end{aligned} \tag{3.30}$$

Since the cost of computing (3.29) in the non-linear case will not exceed the complexity of computing the multiplication of two n-dimensional matrices $\mathbf{M}_i^{1/2}(\mathbf{M}_i^{1/2})^T$, the algorithmic complexity of (3.30) is the same as that of (3.17) and (3.21).

3.4 Experiments on Uncertainty Support Vector Classification

In this section, several experimental results of USVC will be shown for linear and non-linear classification with some synthetic data sets generated. It should be noted that the experiments in this section are only intended as illustrations to convey the nature of the algorithms. The experimental code is based on the MATLAB SVM toolbox (Gunn, 1998). SeDuMi (Sturm, 1999) is implemented as a freely available SOCP optimisation toolkit.

3.4.1 SeDuMi

SeDuMi is an add-on interior-point method for MATLAB, which solves convex optimisation problems with linear, quadratic and semidefinite constraints. Furthermore, SeDuMi can take full advantage of sparsity, leading to significant speed benefits and have large scale optimisation problems solved efficiently, by exploiting sparsity.

SeDuMi stands for Self-Dual-Minimisation: it implements the self-dual embedding technique for optimisation over self-dual homogeneous cones (Sturm, 1999). The self-dual

embedding technique was proposed by Ye et al. (1994), essentially making it possible to solve certain optimisation problems in a single phase, leading either to an optimal solution, or to a certificate of infeasibility. Optimisation over self-dual homogeneous cones, concisely, optimisation over symmetric cones, was first studied by Nesterov and Todd (1997).

In order to solve such a SOCP problem, it is necessary to introduce the form needed for the quadratic constraints and the quadratic cone in SeDuMi. A quadratic cone is by definition a cone of the form,

$$\mathbb{L}^n := \{(x_1, \boldsymbol{x}_2) \in \mathbb{R} \times \mathbb{R}^{n-1} | x_1 \geq \|\boldsymbol{x}_2\|\}. \tag{3.31}$$

and SOCP optimisation problem can be transformed as the following QP problem:

$$\min\{y_1 + y_2 | y_1 \geq \|\boldsymbol{q} - \mathbf{P}\boldsymbol{y}_3\|, y_2 \geq \sqrt{1 + \|\boldsymbol{y}_3\|^2}\}, \tag{3.32}$$

where $\mathbf{P}$ is a given matrix, and $\boldsymbol{q}$ is a given vector. Problem (3.32) is a robust least squares problem (Ghaoui and Lebret, 1997). The decision variables are the scalars y_1 and y_2, and the vector $\boldsymbol{y}_3$. So this problem has two quadratic constraints (Sturm, 1999),

$$(y_1, \boldsymbol{q} - \mathbf{P}\boldsymbol{y}_3) \in \text{Qcone}, \quad \left(y_2, \begin{bmatrix} 1 \\ \boldsymbol{y}_3 \end{bmatrix}\right) \in \text{Qcone}, \tag{3.33}$$

where Qcone denotes the quadratic cones.

In SeDuMi, the number of iterations for the interior-point method to decrease the duality gap to a constant fraction of itself is bounded with time complexity $O\left(\sqrt{l}\log(1/\epsilon)\right)$ (Sturm, 1999), where l denotes the number of second order cones in (3.30) and ϵ provides termination control. SeDuMi will terminate successfully if it finds a solution that violates feasibility and optimality requirements by no more than ϵ.

In each iteration of the interior-point method, the algorithmic complexity is $O\left(ln^3\right)$ (Lobo et al., 1998) for the SOCP problem (3.30), where n is the number of the attributes of the inputs. Therefore, the optimisation complexity of USVC is bounded by $O\left(l^{3/2}n^3\log(1/\epsilon)\right)$, which can be written as $O\left(l^{3/2}n^3\right)$ because we do not change the default SeDuMi setting $\epsilon = 10^{-9}$ (Sturm, 1999) in our experiments. SVC reduces to a quadratic program which initially can be solved with an optimiser in quadratic time $O\left(l^2\right)$. In this work, SVC is solved by SeDuMi and its algorithmic complexity per iteration is $O\left(l\right)$, thus the optimisation complexity of SVC is bounded by $O\left(l^{3/2}\right)$.

3.4.2 Linear Case

Introducing the linear mapping function $\phi(\boldsymbol{x}) = \boldsymbol{x}$ to formula (3.27), (3.28) and (3.29) in the case of feature space, we have

$$\begin{aligned} \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} &= \boldsymbol{x}_j, \\ \left[\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_j}\right]^T &= \boldsymbol{x}_i^T, \\ \frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} &= \mathbf{I}, \end{aligned} \tag{3.34}$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix. If replacing the counterpart above in (3.30),

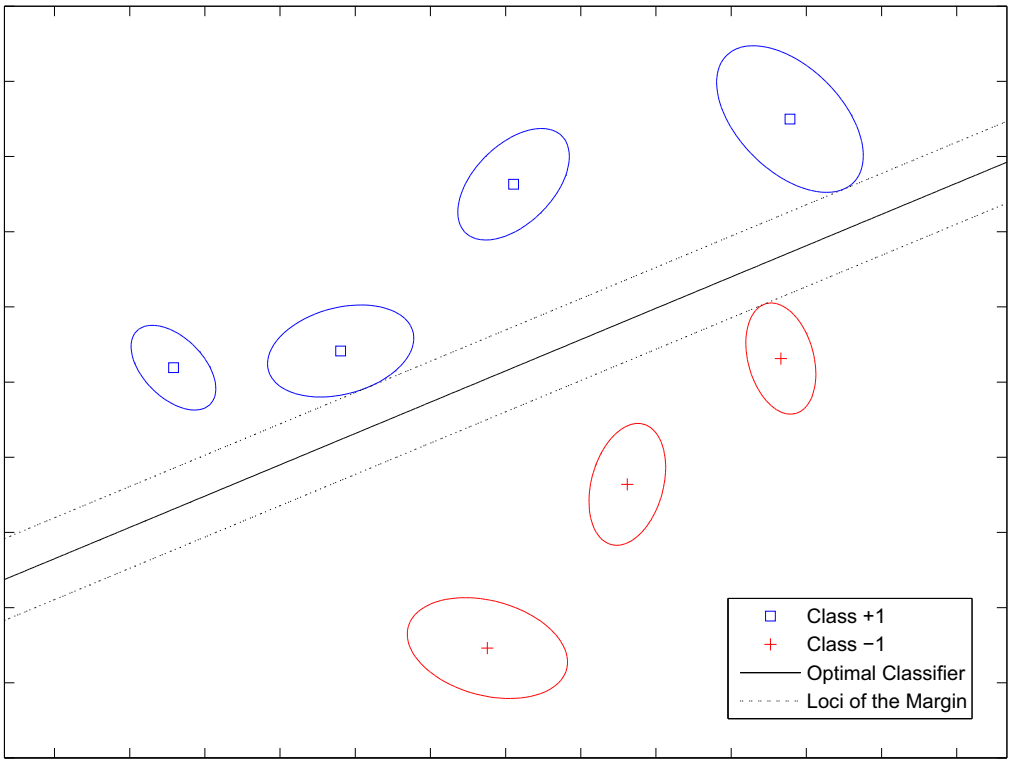
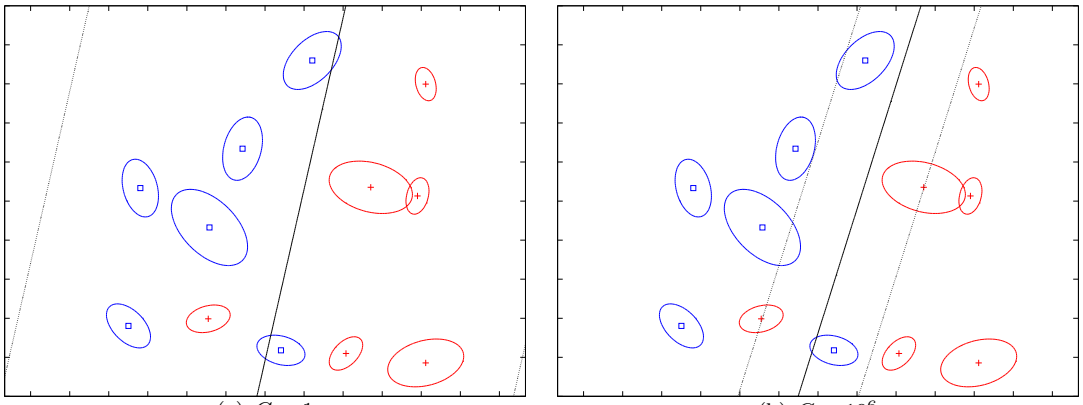


FIGURE 3.3: Linearly separable case of USVC, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin.

the problem same as (3.21) can be retrieved. The separable and non-separable case are shown in Figures 3.3 and 3.4 respectively.

In Figure 3.3, the uncertain inputs from one class do not mix with those from the other class. Therefore, the optimal linear classifier can clearly separate the inputs of two classes. Like SVC, the loci of the margin are tangential to three of all uncertainty ellipses, and these three uncertain inputs are support vectors, which provide information for constructing the optimal separating line in the classification. This result is consistent with the values of α_i.



(a) $C = 1$ (b) $C = 10^6$

FIGURE 3.4: Linearly non-separable case of USVC, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin. (a) high misclassification tolerance determined by a small $C = 1$. (b) low misclassification tolerance determined by a large $C = 10^6$.

While in Figure 3.4, some uncertain inputs from one class mix with those from the other class, which is non-separable case for the linear classifier. With the introduction of the penalty parameters ξ_i, the USVC is applied in this linearly non-separable case, where the same data set is trained with different values of the regularisation parameter C, which behaves in a similar manner to that of SVC. Different C determine different tolerance of misclassification, the smaller the C is, the larger the margin ρ becomes and more uncertain inputs contribute as support vectors.

3.4.3 Non-Linear Case

In this subsection, some examples are illustrated with non-linear kernel functions applied to validate the non-linear theory in USVC.

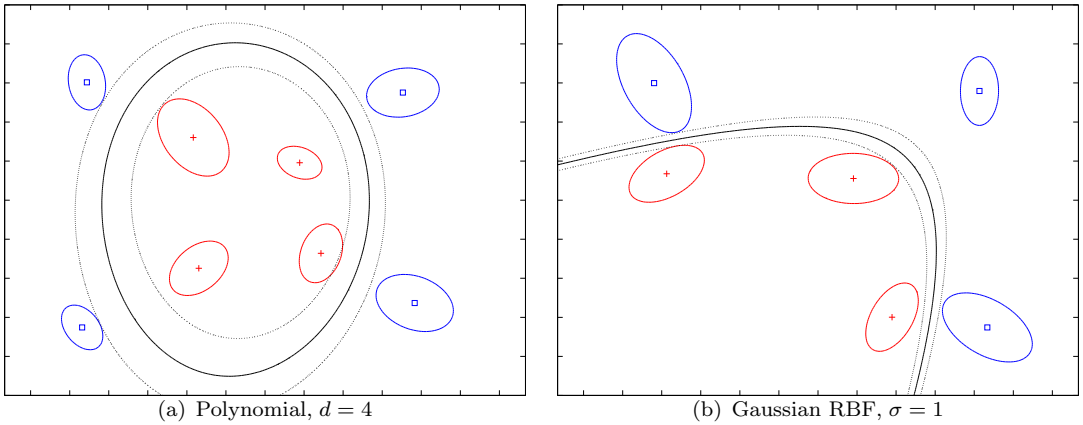
3.4.3.1 Polynomial Kernel

The polynomial kernels have been widely implemented in non-linear applications of SVM. Since a polynomial kernel function is directional, all inputs with the same direction of the support vector will have a high output from the kernel. Polynomial kernels are suited for problems where all the training data is normalised. According to its kernel function, the first and second derivatives can be derived to retrieve the optimisation

problem with the polynomial kernel function,

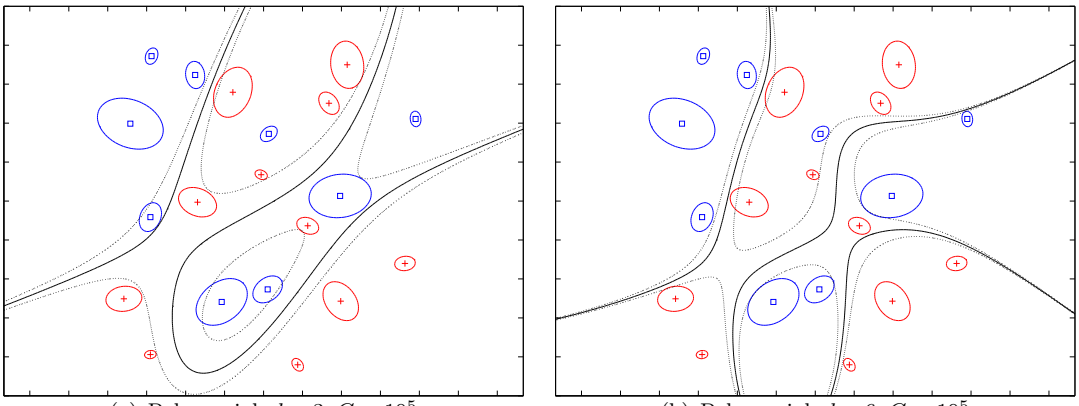
$$\begin{aligned} K(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^d \\ \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} &= d(\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^{d-1} \boldsymbol{x}_j \\ \left[\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_j}\right]^T &= d(\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^{d-1} \boldsymbol{x}_i^T \\ \frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} &= d(d-1)(\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^{d-2} \boldsymbol{x}_j \boldsymbol{x}_i^T + d(\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^{d-1} \mathbf{I}, \end{aligned} \tag{3.35}$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix with the same size as $\boldsymbol{x}_i \boldsymbol{x}_j^T$. The separable case of USVC with the polynomial kernel is shown in Figure 3.5(a).



(a) Polynomial, $d = 4$ (b) Gaussian RBF, $\sigma = 1$

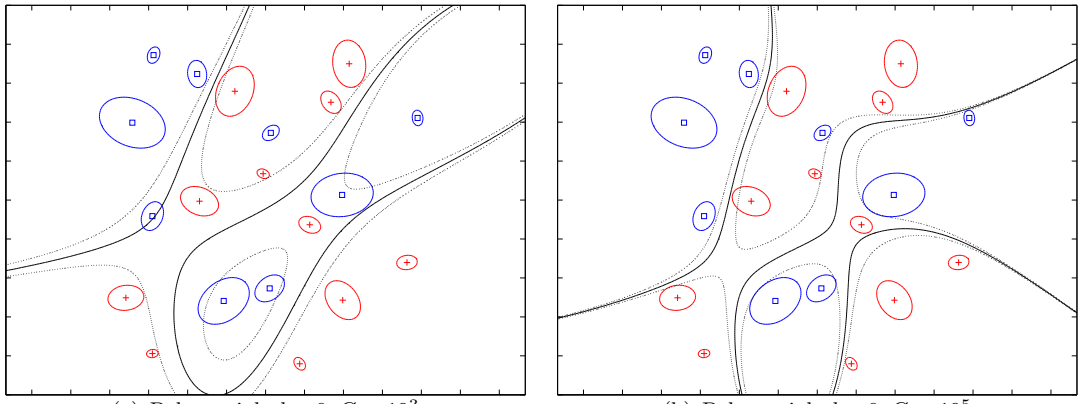
FIGURE 3.5: Non-linearly separable case of USVC, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin. (a) non-linear classification with polynomial kernel function $(\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^3$. (b) non-linear classification with Gaussian radial basis kernel function $\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2}\right)$.



(a) Polynomial, $d = 3$, $C = 10^5$ (b) Polynomial, $d = 6$, $C = 10^5$

FIGURE 3.6: Non-linearly non-separable case of USVC by implementing the polynomial kernels with different parameters d, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin.

In Figure 3.6, the inputs obtain higher outputs from the polynomial kernel with a larger parameter d, which makes the decision boundary not as smooth as the optimal classifier derived by the polynomial kernel with a smaller d. Like the linear case, the regularisation parameter C controls the misclassification tolerance. As a consequence, the obtained optimal hyperplane have some inevitable inputs misclassified in the classification with a smaller C in Figure 3.7(a), where more support vectors are included to construct a smoother decision boundary.



(a) Polynomial, $d = 6$, $C = 10^3$ (b) Polynomial, $d = 6$, $C = 10^5$

FIGURE 3.7: Non-linearly non-separable case of USVC by implementing the polynomial kernels with different regularisation parameter C, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin.

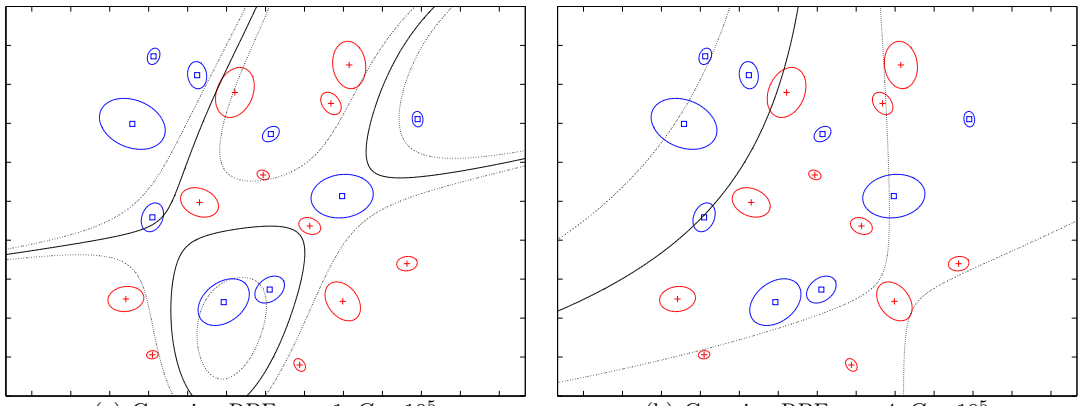
3.4.3.2 Gaussian Radial Basis Function Kernel

Radial basis functions (RBFs) have received significant attention, most commonly with a Gaussian of the form,

$$\begin{aligned}
K(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \\
\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} &= \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\sigma^2} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \\
\left[\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_j}\right]^T &= \frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T}{\sigma^2} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \\
\frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} &= \frac{1}{\sigma^2}\mathbf{I} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) - \frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T}{\sigma^4} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right),
\end{aligned} \tag{3.36}$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix. The output of the kernel is dependent on the Euclidean distance of the support vector $\boldsymbol{x}_j$ from the input $\boldsymbol{x}_i$. The support vector is the centre of the Gaussian RBF and σ determines the area of influence this support vector has over the input space. The corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularised, so the infinite dimension

does not spoil the results. The separable case of USVC with the Gaussian RBF kernel is shown in Figure 3.5(b). Figure 3.8 shows the non-separable case of USVC with the Gaussian RBF kernel. The character of the Gaussian RBF kernel function is illustrated



(a) Gaussian RBF, $\sigma = 1$, $C = 10^5$ (b) Gaussian RBF, $\sigma = 4$, $C = 10^5$

FIGURE 3.8: Non-linearly non-separable case of USVC by implementing the Gaussian radial basis function kernels with different parameters σ, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin.

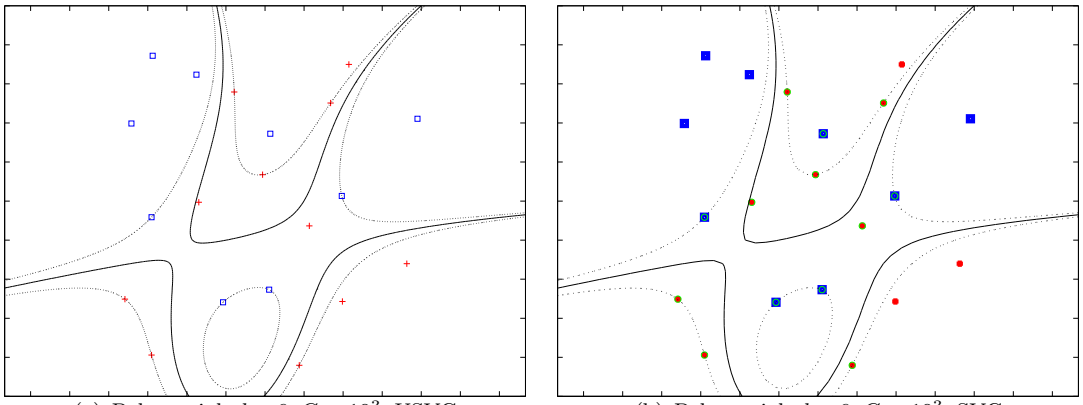
in Figure 3.8, where the large value of σ gives a smoother decision boundary. This is because a RBF with large σ will allow a support vector to have a strong influence over a larger area. A large σ value also increases the value of the Lagrange multiplier α for the classifier. When one support vector influences a larger area, all other support vectors in the area will increase in α value to counter the influence. Consequently, all α values will reach a balance at a larger magnitude.

3.4.4 Result of Degenerate Case

When no uncertainty information is available, which means $\mathbf{M}_i$ are zero matrices, the dual problem of SVC can be retrieved from problem (3.21). In this case, $\boldsymbol{\beta}_i = \mathbf{0}$ and $\|\boldsymbol{\beta_i}\| \leq \alpha_i$ can be rewritten as $\alpha_i \geq 0$ in the constraints of (3.17), (3.21) and (3.30). The resulting problem is the SVC optimisation,

$$\begin{aligned} \max \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ s.t. \quad & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l. \qquad (\textbf{Non-separable case}) \end{aligned} \tag{3.37}$$

It can be seen from Figure 3.9 that Figure 3.9(a) is quite similar to Figure 3.9(b). The slight difference between Figure 3.9(a) and Figure 3.9(b) can be explained by the different optimisation toolboxes used in the implementations, SeDuMi in USVC and LOQO (Vanderbei, 2006) in SVC.

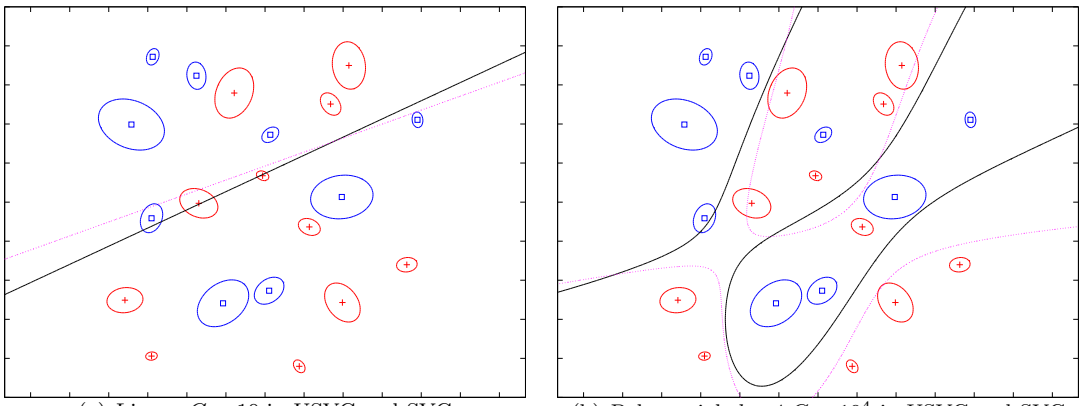


(a) Polynomial, $d = 6$, $C = 10^3$, USVC (b) Polynomial, $d = 6$, $C = 10^3$, SVC

FIGURE 3.9: Degenerate case of USVC by implementing the polynomial kernels, where the solid line denotes the optimal classifier and the dotted lines mark the loci of the margin.

3.4.5 Experimental Comparison of USVC and SVC

The same synthetic data set used for non-linearly non-separable case can be applied to both USVC and SVC to show their different characteristics in the classification when the uncertainty information is available. To compare the algorithms objectively, the value of the regularisation parameter C leading to the best performance must be determined first. Since the size of the data set is very small, the leave-one-out cross-validation is chosen to search the proper result by minimising the cross-validation error of the misclassified $\boldsymbol{x}_i$. The optimal classifiers of USVC and SVC are shown in Figure 3.10.



(a) Linear, $C = 10$ in USVC and SVC (b) Polynomial, $d = 4$,$C = 10^4$ in USVC and SVC

FIGURE 3.10: Experimental comparison of USVC and SVC over the same data set, where USVC is represented by the solid line and SVC is represented by the dashed line. (a) linear classification. (b) non-linear classification with polynomial kernels.

Following the characteristics of support vectors, USVC is prone to strictly follow the nearest edges of the uncertainties to achieve its optimal classifier, which leads to the result that if some inputs' ellipsoids have been correctly classified, their corresponding

unknown original inputs will likely be classified correctly by following their own distributions. Without ellipsoids representing the uncertainties, the classifier of SVC goes through some of the uncertainties, where the corresponding original inputs following their distributions will possibly be misclassified by the optimal hyperplane obtained for SVC, though the means of these inputs may have been correctly classified.

3.5 Summary

In this chapter, a new method of classification subject to input uncertainty, USVC, is derived from SVC by incorporating uncertain inputs. Kernel functions can be used by a novel kernelisation formulation to generalise this proposed technique to non-linear models and the resulting optimisation problem is a SOCP with a unique solution. Different kernel functions along with USVC and SVC are applied to the experiments in this section, which are intended as illustrations to convey the nature of the algorithms.

To obtain their optimal separating hyperplanes, both the new approach USVC and the traditional method SVC implement soft margin classification to formulate their optimisation problems. The solutions are determined by the support vectors, which are a subset of the input data. Comparing USVC with SVC, it can be found that USVC follows the formulation of SVC by inheriting SVC's geometric and statistical characteristics except for the introduction of uncertainty matrices, whose sizes and directions along with the positions of the uncertain inputs determine the position and the direction of the optimal classifier. Moreover, SVC can be retrieved from USVC in the degenerate case of no uncertainties.

Chapter 4

Other Related Methods

The classification with information of input uncertainty has started to receive some attention in recent machine learning research. Known as a maximum margin classifier, the SVM can be regarded as one of the attractive implements of classification process subject to input uncertainty. As derived in Chapter 3, SVC can be extended to USVC by incorporating noise-specific covariance information as additional soft constraints. USVC not only can accommodate the inputs with certain values and the uncertain inputs following isotropic Gaussian distribution like SVC can do, but also can accommodate the uncertain inputs following general anisotropic Gaussian distribution. In dichotomy classification, the margin needs to be maximised is no longer the nearest distance between inputs from both classes, but the nearest distance between uncertain inputs with their distributions. Theoretically, USVC makes no assumption about original inputs before contamination, thus classifying uncertain inputs with their distributions tends to make it certain that original inputs are correctly classified as well. Some recently developed methods of classification subject to input uncertainty will be analysed in this chapter.

4.1 Total Support Vector Classification

Like USVC, Bi and Zhang (2005) proposed a novel formulation derived from SVC, which can accommodate uncertainties in input data as well. The algorithm is named the total support vector classification (TSVC).

4.1.1 Linear Case

Considering the previous definition in Section 2.2.3, a set of training input data $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, l$ are given, where $\boldsymbol{x}_i$ is corrupted from the original uncorrupted input $\boldsymbol{x}_{io}$ by Gaussian noise, σ_i is the standard deviation of the Gaussian noise model and it is the estimate

of the uncertainty (e.g. variance) for $\boldsymbol{x}_i$. In Bi and Zhang (2005), it is assumed that the inputs are subject to an additive noise, $\boldsymbol{x}_{io} = \boldsymbol{x}_i + \triangle\boldsymbol{x}_i$, where noise $\triangle\boldsymbol{x}_i$ follows a certain distribution, which is designed to be bounded by an uncertainty model $\|\triangle\boldsymbol{x}_i\| \leq \delta_i$ with uniform priors. The bound δ_i has a similar effect of the standard deviation σ_i in the Gaussian noise model, whereas the squared penalty term $\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{io}\|^2}{2\sigma_i^2}$ is replaced by $\|\triangle\boldsymbol{x}_i\| \leq \delta_i$. In linear case, the parameter θ in (2.26) and (2.27) is replaced by the weight vector $\boldsymbol{w}$ and the offset bias b. The TSVC solution in linearly separable case is shown as follows,

$$\begin{aligned} \min_{\boldsymbol{w},b,\triangle\boldsymbol{x}_i} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i(\boldsymbol{w}^T(\boldsymbol{x}_i + \triangle\boldsymbol{x}_i) + b) \geq 1, \\ & \|\triangle\boldsymbol{x}_i\| \leq \delta_i, \qquad i = 1, \ldots, l. \end{aligned} \tag{4.1}$$

Soft margin method is also used in TSVC by the introduction of slack variables $\xi_i = \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\}$, the square loss $\frac{(\theta^T\boldsymbol{x}_{io} - y_i)^2}{2\sigma^2}$ in (2.26) or the logistic loss $\ln(1 + e^{-\theta^T\boldsymbol{x}_{io}y_i})$ in (2.27) is simply replaced by ξ_i in linearly non-separable case,

$$\begin{aligned} \min_{\boldsymbol{w},b,\triangle\boldsymbol{x}_i,\xi_i} \quad & C\sum_{i=1}^{l}\xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i(\boldsymbol{w}^T(\boldsymbol{x}_i + \triangle\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \\ & \|\triangle\boldsymbol{x}_i\| \leq \delta_i, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{4.2}$$

Lemma 4.1. (Bi and Zhang, 2005) *For any given hyperplane* $(\boldsymbol{w}, b)$*, the solution* $\triangle\hat{\boldsymbol{x}}_i$ *of problem* (4.2) *is* $\triangle\hat{\boldsymbol{x}}_i = y_i\delta_i\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$, $i = 1, \ldots, l$.

$\triangle\hat{\boldsymbol{x}}_i$ is the optimal solution of $\triangle\boldsymbol{x}_i$ and we derive the fact that when the optimal $\hat{\boldsymbol{w}}$ is obtained, the optimal $\triangle\hat{\boldsymbol{x}}_i$ can be represented in terms of $\hat{\boldsymbol{w}}$.

Therefore, we can define $S_{\boldsymbol{w}}(\boldsymbol{x}) = \{\boldsymbol{x}_i + y_i\delta_i\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, i = 1, \ldots, l\}$. $S_{\boldsymbol{w}}(\boldsymbol{x})$ is a set of points that are obtained by shifting the original points labeled $+1$ along $\boldsymbol{w}$ and points labeled -1 along $-\boldsymbol{w}$ respectively, to its individual uncertainty boundary.

Theorem 4.2. (Bi and Zhang, 2005) *The optimal hyperplane* $(\hat{\boldsymbol{w}}, \hat{b})$ *obtained by the* TSVC *problem* (4.1) *separates* $S_{\hat{\boldsymbol{w}}}(\boldsymbol{x})$ *with the maximal margin. The optimal hyperplane* $(\hat{\boldsymbol{w}}, \hat{b})$ *obtained by the* TSVC *problem* (4.2) *separates* $S_{\hat{\boldsymbol{w}}}(\boldsymbol{x})$ *with the maximal soft margin.*

Using Lemma 4.1 and Theorem 4.2, $\triangle\hat{\boldsymbol{x}}_i = y_i\delta_i\frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|}$ can be introduced to remove $\triangle\hat{\boldsymbol{x}}_i$ in the optimisation problem. The problems (4.1) and (4.2) can be converted to the optimisation over variable $\boldsymbol{w}$, b, ξ_i, the linearly non-separable problem (4.2) can be

rewritten as

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & C\sum_{i=1}^{l}\xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \delta_i\|\boldsymbol{w}\| \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{4.3}$$

Moreover, without loss of generality, the simple bounded uncertainty model $\|\triangle\boldsymbol{x}_i\| \leq \delta_i$ can be replaced by a more general noise-specific covariance matrix $\mathbf{M}_i$ according to Definition 3.1. As the result, $\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|$ introduced by ellipsoidal uncertainties replace $\delta_i\|\boldsymbol{w}\|$ in (4.3), we have

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & C\sum_{i=1}^{l}\xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, \ldots, l, \end{aligned} \tag{4.4}$$

and the optimisation problem of USVC (3.19) can be rewritten as,

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & C\sum_{i=1}^{l}\xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, \ldots, l. \end{aligned} \tag{4.5}$$

The first constraints of (4.4) and (4.5) come from (3.4) with the probability confidence r set to ∓ 1 respectively. Figure 4.1 shows a figure originating from Bi and Zhang (2005)

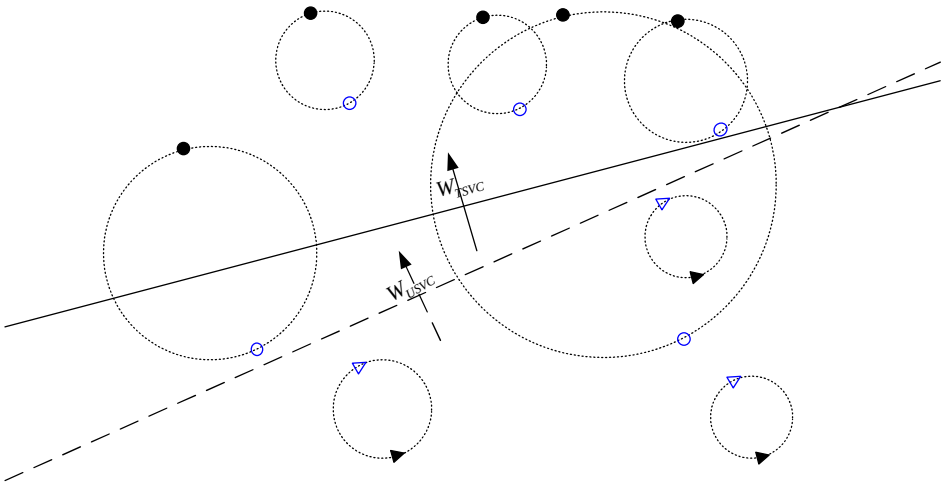


FIGURE 4.1: Geometric interpretation of TSVC and USVC.

with the USVC solution superimposed to illustrate the different geometric interpretation

between TSVC and USVC. In Figure 4.1, dotted circles represent spherical uncertainties. The circles with circular points and the circles with triangular points respectively represent the uncertain inputs from either class where $y_i = \pm 1$. In classification, TSVC uses the farthest points (solid points, obtained by Lemma 4.1 and realised by shifting the original points labelled $+1$ along $\boldsymbol{w}$ and the points labelled -1 along $-\boldsymbol{w}$ respectively) in the distributions of the uncertain inputs as reference to achieve the optimal hyperplane ($\boldsymbol{w}_{\text{TSVC}}$, solid line), while USVC uses the nearest points (hollow points, obtained by Lemma 4.1 and realised by shifting the original points labelled $+1$ along $-\boldsymbol{w}$ and the points labelled -1 along $\boldsymbol{w}$) in the distributions of the uncertain inputs to the optimal hyperplane ($\boldsymbol{w}_{\text{USVC}}$, dashed line) to compute the classifier.

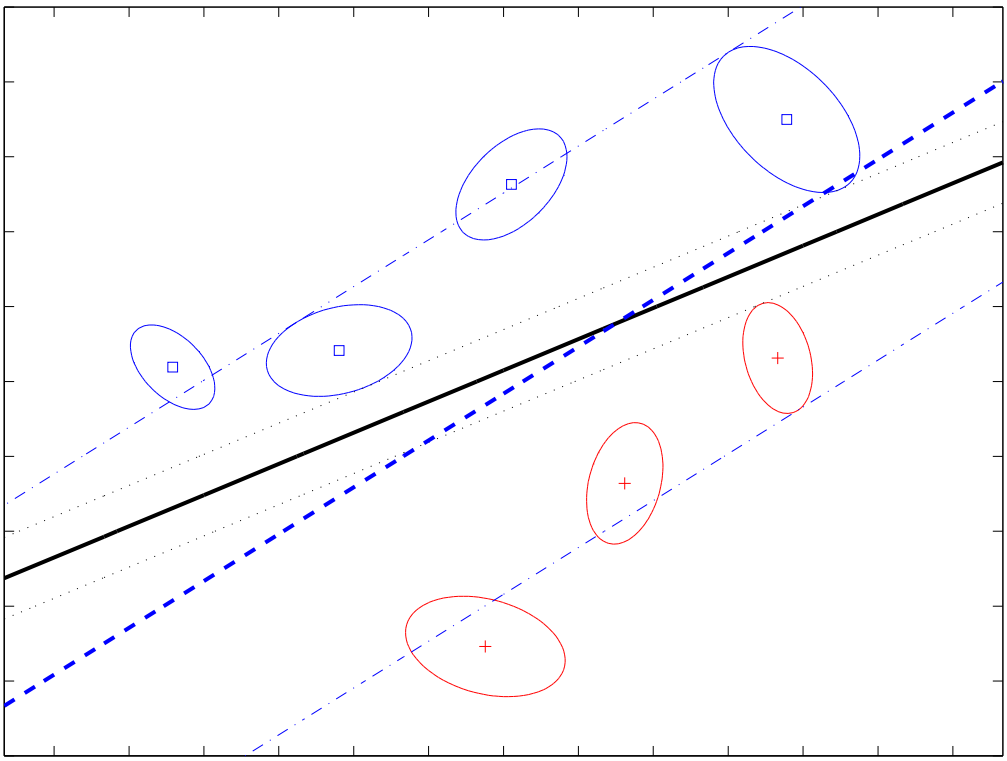


FIGURE 4.2: Linearly separable case of TSVC and USVC, where the solid line denotes the optimal classifier of USVC and the dotted lines mark the loci of the margin of USVC solution. The dashed line denotes the optimal classifier of TSVC and the loci of the margin of TSVC solution are marked by the dash-dot lines.

A linearly separable case of TSVC and USVC with hard margin is shown in Figure 4.2 by reproducing the classification in Figure 3.3. In the figure, three out of all inputs are chosen as the support vectors for each algorithm. The totally different decision boundaries of TSVC and USVC are dependent on their support vectors which are chosen by TSVC and USVC based on their dramatically opposite strategies. The objective of TSVC is to restore the true target boundary from contaminated data sets which are created under the assumption that training inputs contaminated by Gaussian noise tend to move towards the other class by attempting to cross the original boundary. TSVC can well accommodate this assumption to reduce the effect from the noise in the classification

by selecting the farthest points as reference to get the optimal solution. USVC choosing the nearest points is to guarantee that uncertain inputs are correctly classified along with their original inputs. Consequently, the obtained optimal solution of USVC may be far different from the target function under some kinds of contamination.

4.1.2 Limitations

According to the definition of SOCP in (3.11), problem (4.5) is a SOCP problem and certainly it is a convex optimisation. We need to know whether (4.4) is a SOCP problem or not.

Definition 4.3. (Boyd and Vandenberghe, 2004) A convex optimisation problem is on of the form

$$\begin{aligned} \min \ & f_0(\boldsymbol{x}) \\ s.t. \ & f_i(\boldsymbol{x}) \leq b_i, \qquad i = 1, \ldots, m, \end{aligned}$$

where the functions $f_0, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex, *i.e.*, satisfy

$$f_i(\omega \boldsymbol{x}_1 + (1-\omega)\boldsymbol{x}_2) \leq \omega f_i(\boldsymbol{x}_1) + (1-\omega) f_i(\boldsymbol{x}_2),$$

$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$ and for any $\omega \in \mathbb{R}$ with $0 \leq \omega \leq 1$.

Theorem 4.4. *Problem* (4.4)

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi_i} \ & C \sum_{i=1}^{l} \xi_i + \frac{1}{2} \|\boldsymbol{w}\|^2 \\ s.t. \ & y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) + \|\mathbf{M}_i^{1/2} \boldsymbol{w}\| \geq 1 - \xi_i \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l, \end{aligned}$$

is not a convex optimisation

Proof. $\forall \{\boldsymbol{w}_1, b_1, \xi_1\}, \{\boldsymbol{w}_2, b_2, \xi_2\} \in \{\mathbb{R}^n, \mathbb{R}, \mathbb{R}\}$, $\forall \{\boldsymbol{x}, y, \mathbf{M}\} \in \mathbb{V}$ (the set of support vectors). According to the characteristics of support vectors, the first constraint can be rewritten as follows by introducing $\{\boldsymbol{w}_1, b_1, \xi_1\}$ and $\{\boldsymbol{w}_2, b_2, \xi_2\}$,

$$f(\boldsymbol{w}_1, b_1, \xi_1): \quad y(\boldsymbol{w}_1^T \boldsymbol{x} + b_1) + \|\mathbf{M}^{1/2} \boldsymbol{w}_1\| = 1 - \xi_1. \tag{4.6}$$

$$f(\boldsymbol{w}_2, b_2, \xi_2): \quad y(\boldsymbol{w}_2^T \boldsymbol{x} + b_2) + \|\mathbf{M}^{1/2} \boldsymbol{w}_2\| = 1 - \xi_2. \tag{4.7}$$

For any $0 \leq \omega \leq 1$, we have,

$$\begin{aligned}(1-\omega)f(\boldsymbol{w}_1, b_1, \xi_1) + \omega f(\boldsymbol{w}_2, b_2, \xi_2): \quad & 1 - [(1-\omega)\xi_1 + \omega\xi_2] \\ & = y[(1-\omega)\boldsymbol{w}_1 + \omega\boldsymbol{w}_2]^T\boldsymbol{x} + y[(1-\omega)b_1 + \omega b_2] \\ & \quad + (1-\omega)\|\mathbf{M}^{1/2}\boldsymbol{w}_1\| + \omega\|\mathbf{M}^{1/2}\boldsymbol{w}_2\|.\end{aligned} \tag{4.8}$$

Since $(1-\omega)\|\mathbf{M}^{1/2}\boldsymbol{w}_1\| + \omega\|\mathbf{M}^{1/2}\boldsymbol{w}_2\| \geq \|(1-\omega)\mathbf{M}^{1/2}\boldsymbol{w}_1 + \omega\mathbf{M}^{1/2}\boldsymbol{w}_2\|$, it is able to find a ω and let

$$\begin{aligned}& 1 - [(1-\omega)\xi_1 + \omega\xi_2] \\ > & \, y[(1-\omega)\boldsymbol{w}_1 + \omega\boldsymbol{w}_2]^T\boldsymbol{x} + y[(1-\omega)b_1 + \omega b_2] \\ & + \|\mathbf{M}^{1/2}[(1-\omega)\boldsymbol{w}_1 + \omega\boldsymbol{w}_2]\|.\end{aligned} \tag{4.9}$$

Therefore, from Definition 4.3, we know that the functions $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \geq 1 - \xi_i$, $i = 1, \ldots, l$ are not convex, then the problem is not a convex optimisation. □

Corollary 4.5. *Since problem* (4.4) *is not a convex optimisation,* (4.4) *is not a* SOCP *problem.*

Fortunately, Tikhonov regularisation (Golub et al., 1999; Bi and Vapnik, 2003; Bi and Zhang, 2005) min $C\sum\xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2$ has an important equivalent formulation as min $\sum\xi_i$, subject to $\|\boldsymbol{w}\| \leq \gamma$, where γ is a positive constant. It can be shown that if $\gamma \leq \|\boldsymbol{w}^*\|$ where $\boldsymbol{w}^*$ is the solution to (4.2) with $\frac{1}{2}\|\boldsymbol{w}\|^2$ removed, then the optimal solution is identical to the one of the Tikhonov regularisation problem for an appropriately chosen C. In this case, the constraint $\|\boldsymbol{w}\| \leq \gamma$ at optimality, which leads to $\|\hat{\boldsymbol{w}}\| = \gamma$. Consequently, the TSVC problem (4.3) can be converted to a simple SOCP with the constraint $\|\boldsymbol{w}\| \leq \gamma$ or a QCQP as follows if equivalently using $\|\boldsymbol{w}\|^2 \leq \gamma^2$.

$$\begin{aligned}\min_{\boldsymbol{w},b,\xi_i} \quad & \sum_{i=1}^{l}\xi_i \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \gamma\delta_i \geq 1 - \xi_i, \\ & \|\boldsymbol{w}\|^2 \leq \gamma^2, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l,\end{aligned} \tag{4.10}$$

and the dual formulation of (4.10) in dual variables α_i is given by Bi and Zhang (2005) as follows,

$$\begin{aligned}\min_{\alpha_i} \quad & \gamma\sqrt{\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j} - \sum_{i=1}^{l}(1-\gamma\delta_i)\alpha_i \\ s.t. \quad & \sum_{i=1}^{l}\alpha_i y_i = 0, \qquad 0 \leq \alpha_i \leq 1, \qquad i = 1, \ldots, l.\end{aligned} \tag{4.11}$$

Although TSVC can be recast to a SOCP problem, directly implementing this SOCP problem will be computationally expensive. Therefore, an iterative approach, is proposed by Bi and Zhang (2005) to achieve the optimal solution of TSVC. According to Definition 3.1, this iterative approach can be reformulated as follows in linear case. As the result,

Algorithm 1 TSVC's Iterative Algorithm for Linear Case

Initialise $\triangle \boldsymbol{x}_i = 0$, repeat the following steps until $\sum_{i=1}^{l} \xi_i$ reaches a local minimum:
1. Fix $\triangle \boldsymbol{x}_i$, $i = 1, \ldots, l$ to the current value, solve problem (4.2) without its constraints $\|\triangle \boldsymbol{x}_i\| \leq \delta_i$ for parameters $\boldsymbol{w}$ and b. With fixed $\triangle \boldsymbol{x}_i$, (4.2) is then transformed to a SVC primal problem;
2. Use the obtained $\boldsymbol{w}$, b to calculate $\triangle \boldsymbol{x}_i$, $i = 1, \ldots, l$, and $\sum_{i=1}^{l} \xi_i$, where $\triangle \boldsymbol{x}_i = y_i \frac{\mathbf{M}_i \boldsymbol{w}}{\|\mathbf{M}_i^{1/2} \boldsymbol{w}\|}$ and $\xi_i = \max\left\{0, 1 - \left[y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) + \|\mathbf{M}_i^{1/2} \boldsymbol{w}\|\right]\right\}$.

this iterative approach may not achieve the global optimum of $\sum_{i=1}^{l} \xi_i$ but only reach many local optimum by a termination criterion because $\sum_{i=1}^{l} \xi_i$ obtained in each step is the optimal solution of SVC depending upon $\boldsymbol{x}_i + \triangle \boldsymbol{x}_i$, which vary around the farthest points during the iterations. This has been supported by the experimental results. In the experiment of TSVC, if no fixed termination criterion is determined in Algorithm 1, the more iterations, the better results we will have.

4.1.3 Non-Linear Case

TSVC can also be extended to the non-linear case by using a kernel function K. In non-linear case, the optimisation problem obtained is similar to that of (4.2) only with $\boldsymbol{x}_i$ replaced by $\phi(\boldsymbol{x}_i)$ and $\triangle \boldsymbol{x}_i$ replaced by $\triangle \phi(\boldsymbol{x}_i)$, where ϕ denotes a mapping function. The uncertainties in the input space can also be introduced in the feature space. However, like USVC, when the uncertainty bounded spheres $\|\triangle \boldsymbol{x}_i\| \leq \delta_i$ in the input space are mapped to the feature space, the mapped uncertainty may correspond to an irregular shape in the feature space, which makes the optimisation of TSVC difficult. Thus an approximation strategy is introduced based on the first order Taylor expansion of the kernel function $K(\boldsymbol{x}, \boldsymbol{z})$

$$K(\boldsymbol{x}_i + \triangle \boldsymbol{x}_i, \boldsymbol{z}) = K(\boldsymbol{x}_i, \boldsymbol{z}) + \triangle \boldsymbol{x}_i^T \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{z})}{\partial \boldsymbol{x}_i},$$

where $\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{z})}{\partial \boldsymbol{x}_i}$ is the gradient of $K(\boldsymbol{x}, \boldsymbol{z})$ with respect to $\boldsymbol{x}$ at point $\boldsymbol{x}_i$. Applying the Taylor expansion, we have

$$\begin{aligned} & y_i \left(\sum_j y_j \alpha_j K(\boldsymbol{x}_i + \triangle \boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j) + b \right) \\ \cong\ & y_i \left(\sum_j y_j \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j) + b \right) + y_i \triangle \boldsymbol{x}_i^T \sum_j y_j \alpha_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i}. \end{aligned} \tag{4.12}$$

Definition 3.1, along with the kernelisation method developed in Chapter 3, can be applied to (4.12) to extend Algorithm 1 to non-linear case. More details can be found in Appendix B. This iterative approach of TSVC is designed as follows:

Algorithm 2 TSVC's Iterative Algorithm for Non-Linear Case

Initialise $\triangle \boldsymbol{x}_i = 0$, repeat the following steps until $\sum_{i=1}^{l} \xi_i$ reaches a local minimum:
1. Fix $\triangle \boldsymbol{x}_j$, $j = 1, \ldots, l$ to the current value of $\triangle \boldsymbol{x}_i$, $i = 1, \ldots, l$, solve the non-linear dual problem of SVC for α_j and b;
2. Use the obtained α_j, b to calculate $\triangle \boldsymbol{x}_i$, $i = 1, \ldots, l$, and $\sum_{i=1}^{l} \xi_i$, where
$\triangle \boldsymbol{x}_i = y_i (\mathbf{M}_i^{1/2})^T \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|}$, $\quad \boldsymbol{v}_i = \mathbf{M}_i^{1/2} \sum_j \alpha_j y_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i}$ and
$\xi_i = \max \left\{ 0, 1 - \left[y_i \left(\sum_j \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j) + b \right) + \|\boldsymbol{v}_i\| \right] \right\}$.

The memory requirement of TSVC is of order $O\left(l^2\right)$, the same as that of SVC because every single iteration of the TSVC algorithm is a SVC problem. Since TSVC is an iterative algorithm, the computational cost is related to the number of iterations before $\sum_{i=1}^{l} \xi_i$ reaches its local minimum. In practice, it is not straightforward to determine an optimal termination criterion. For convenience, the number of iterations is preset to a constant and the optimal solution of TSVC is chosen alongside the minimum of all $\sum_{i=1}^{l} \xi_i$ in all iterations. If a tie happens at the minimum of all $\sum_{i=1}^{l} \xi_i$, the latest one will be selected. Hence, in TSVC, the number of iterations is fixed to L_t. In every single iteration, the computational cost includes the cost of constructing auxiliary parameters $\triangle \boldsymbol{x}_i$ and $\boldsymbol{v}_i$, besides, the cost of constructing the matrix $\mathbf{R}$ with order $O\left(l^2 n\right)$ and the optimisation complexity bounded by $O\left(l^{3/2}\right)$ are the same as those of SVC. Following Algorithm 2, the cost of constructing auxiliary parameters in every single iteration is of order $O\left(l l_{\text{sv}} n^2\right)$, where l_{sv} is the number of support vectors. Since l_{sv} is part of l, this cost is bounded with order $O\left(l^2 n^2\right)$. In general, the computational cost of TSVC has the cost of constructing the matrix $\mathbf{R}$ with order $O\left(L_t l^2 n\right)$, the optimisation complexity bounded by $O\left(L_t l^{3/2}\right)$ and the cost of constructing auxiliary parameters with order $O\left(L_t l^2 n^2\right)$.

4.1.4 Experimental Comparison of TSVC and USVC

In this section, we exploit experimental comparison between TSVC and USVC by reproducing the experiment from Bi and Zhang (2005). The data in the experiment are generated by following the exact prescription described in Bi and Zhang (2005), hence, the data differs solely by random input generator which follows the uniform distribution. In the experiments with synthetic data in two dimensions, $l = 50, 100$ training inputs $\boldsymbol{x}_i$ are generated randomly from $[-5, 5]^2$. Two binary classification problems are created with target separating functions $x_1 - x_2 = 0$ and $x_1^2 + x_2^2 = 9$. TSVC and USVC are trained with linear functions and quadratic kernel $(\boldsymbol{x}_i^T \boldsymbol{x}_j)^2$ for the problems respectively.

The training inputs $\boldsymbol{x}_i$ are contaminated by Gaussian noise with mean $[0, 0]$ and covariance matrix $\Sigma = \delta_i \mathbf{I}$ where δ_i is randomly chosen from $[0.1, 0.8]$. The matrix $\mathbf{I}$ denotes the 2×2 identity matrix. We randomly choose $0.1l$ from the first $0.2l$ inputs after inputs are ordered in an ascending order of their distances to the target boundary. For these $0.1l$ inputs, noise is generated by using a larger δ_i randomly drawn from $[0.5, 2]$.

In each problem, ten 50-input training data sets and ten 100-input training data sets are created along with five 10000-input test data sets which are generated in exactly the same manner as the training sets but without contamination. Five corrupted test data sets are then generated from the original test data sets under the same rule of creating training data sets. Models obtained by algorithms are evaluated over original test data sets and their contaminated counterparts respectively. The misclassification error rates averaged over five test data sets are collected in the test sessions. The rates collected from the test sets without contamination are called the test misclassification error (TME), which shows the difference between the obtained classifier and the true target function, and evaluates the abilities that different approaches retrieve the original target function from the corrupted inputs. The other misclassification error rate, named as the number of misclassified centres of uncertainties (NMCU), compares different algorithms by the number of the means ($\boldsymbol{x}_i$ in Definition 3.1) of uncertain inputs ($\boldsymbol{z}_i$ in Definition 3.1) being correctly discriminated in the contaminated test sets.

To select proper regularisation parameter C for each algorithm, stratified 10-fold cross-validation is introduced to evaluate NMCU in training data sets. The finally chosen C is selected from a geometric sequence $1, \sqrt{10}, 10, \ldots, 10^5$ with common ratio $\sqrt{10}$. To remove numerical difference of SOCP optimisers, SeDuMi (Sturm, 1999) is used to replace LOQO (Vanderbei, 2006) hereafter in solving SVC problem based on MATLAB SVM toolbox (Gunn, 1998). The results are shown in Table 4.1 and Table 4.2.

Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	12.91	1.24	11.76	0.66	16.19	3.59	14.04	1.05
SVC	12.00	1.07	11.96	0.95	16.42	3.21	14.19	1.04
TSVC	11.62	0.75	11.58	0.61	14.27	1.16	13.33	0.21

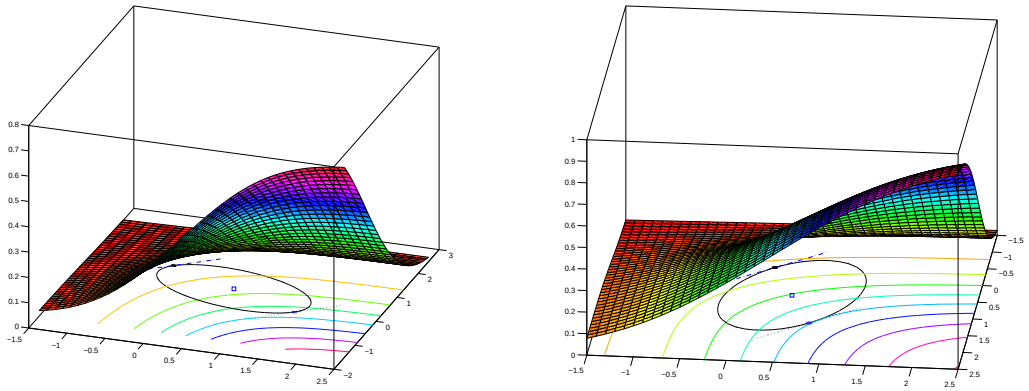
TABLE 4.1: Average test error percentages of NMCU of USVC, TSVC and standard SVC in reproducing Bi and Zhang's experiment, means and standard errors of NMCU are listed in the table.

TSVC performs overall better than USVC and SVC in both measures, and the classifiers obtained by USVC have close performance to those of SVC. TSVC geometrically chooses the farthest points in the uncertainties as reference to obtain the optimal solution. If we recall (3.4) and (3.7), we can find that TSVC's strategy consequently sets the probability confidence $r = -1$, which leads to lower predicted probabilities that the unknown original inputs $\boldsymbol{x}_{io}$ are going to be classified correctly when the farthest points

Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	7.94	2.18	4.67	2.30	9.21	5.03	5.92	2.55
SVC	5.79	2.60	4.73	2.57	9.22	4.66	5.42	2.38
TSVC	4.21	2.30	3.45	2.29	5.09	2.29	2.57	1.01

TABLE 4.2: Average test error percentages of TME of USVC, TSVC and standard SVC in reproducing Bi and Zhang's experiment, means and standard errors of TME are listed in the table.

of the covariance ellipsoids of distributions are designed to be correctly classified for their corrupted counterparts $\boldsymbol{z}_i$ in TSVC. On the contrary, USVC is a conservative approach, which wants to correctly classify uncertain inputs with as much as their distributions by choosing the nearest points in the uncertainties as reference to achieve its optimal solution. In USVC, the probability confidence $r = 1$, which ensure that if uncertain inputs $\boldsymbol{z}_i$ have been correctly classified, their unknown original counterparts $\boldsymbol{x}_{io}$ have higher probabilities to be classified as well. It is important for the discussion that will follow to notice that traditional probabilities differ from the predicted probabilities mentioned here, which predicts probabilities at some positions that unknown original inputs fall within particular subset of the range between these positions and the farthest ends of the distributions to the classifier. These probabilities are determined by both the distributions of uncertain inputs and the optimal classifier. For same uncertain inputs, different classifiers vary the predicted probabilities introduced in USVC and TSVC. A



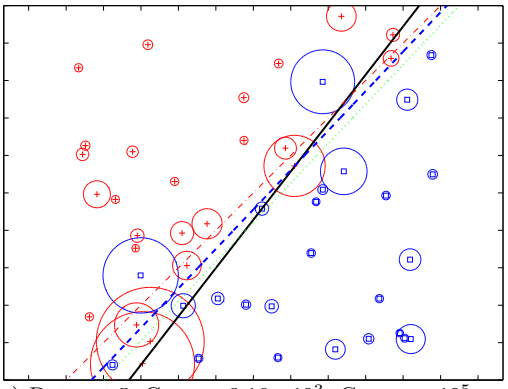
(a) $\boldsymbol{w} = [1; -1]$ (b) $\boldsymbol{w} = [1; 1]$

FIGURE 4.3: Probabilistic interpretation of TSVC and USVC. In the figure, the uncertain input $\boldsymbol{z}_i \sim \mathcal{N}([0.5; 0.5], [1.143, -0.286; -0.286, 0.571])$ remains unchanged, two linear classifiers are applied to this input with different weight vectors $\boldsymbol{w}$. 3D shaded surface and the contour beneath surface represent multivariate normal cumulative distribution function. Hollow point is the nearest point of ellipse to the optimal solution and the dotted line mark the locus of the margin of USVC solution. Solid point is the farthest point of ellipse and the locus of the margin of TSVC solution is marked by the dash-dot line.

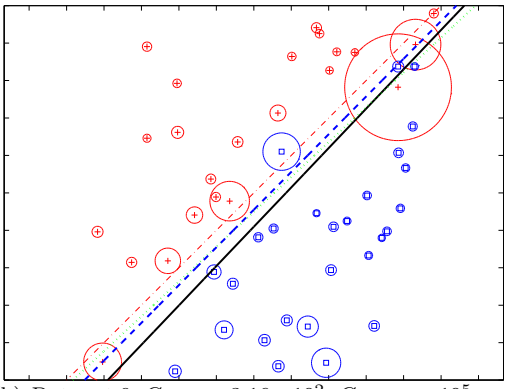
two-dimensional uncertain input is shown with its distribution and two linear classifiers

in Figure 4.3 to illustrate probabilistic difference between TSVC and USVC in classifying uncertain inputs which follow multivariate Gaussian distributions.

The results of linear classification of the 5th and 9th 50-input training data sets are shown in Figure 4.4 and the 1st and 8th data sets are shown for 100-input linear classification in Figure 4.5, in which all algorithms achieve improved performance with increasing training inputs.

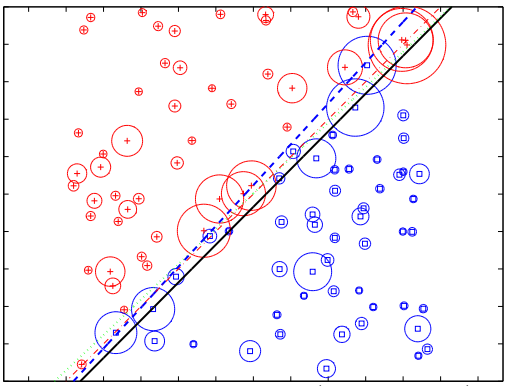


(a) Data set 5, $C_{\mathrm{SVC}} = 3.16 \times 10^3$, $C_{\mathrm{USVC}} = 10^5$ and $C_{\mathrm{TSVC}} = 31.62$

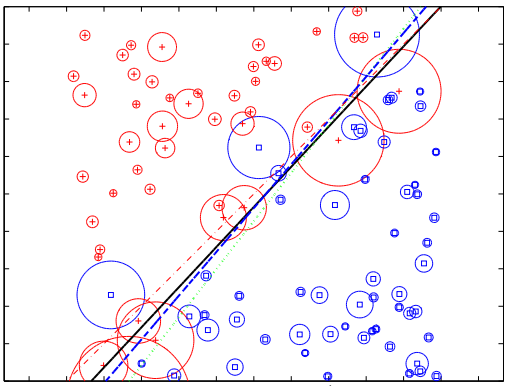


(b) Data set 9, $C_{\mathrm{SVC}} = 3.16 \times 10^2$, $C_{\mathrm{USVC}} = 10^5$ and $C_{\mathrm{TSVC}} = 10$

FIGURE 4.4: Selected results from the 5th and 9th 50-input training data sets for linear classification in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.



(a) Data set 1, $C_{\mathrm{SVC}} = 3.16 \times 10^4$, $C_{\mathrm{USVC}} = 10^4$ and $C_{\mathrm{TSVC}} = 3.16 \times 10^2$

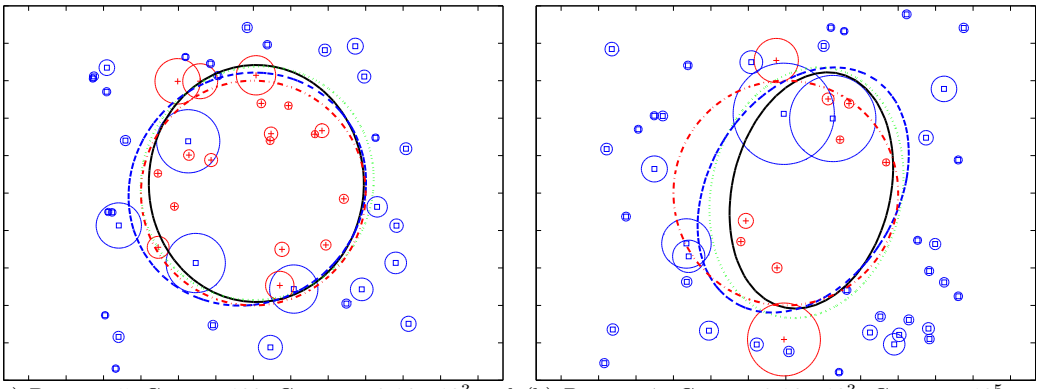


(b) Data set 8, $C_{\mathrm{SVC}} = 3.16 \times 10^4$, $C_{\mathrm{USVC}} = 10$ and $C_{\mathrm{TSVC}} = 10$

FIGURE 4.5: Selected results from the 1st and 8th 100-input training data sets for linear classification in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.

In Figure 4.4, the optimal solution of TSVC is the closest classifier of all algorithms to the target function. Relatively, deviation from the target function usually happens in the solution of USVC due to its characters and some specific training data sets. In the bottom left part of Figure 4.4(a) and the top right part of Figure 4.4(b), some large

uncertainties from one class tend to cross the original boundary, and these uncertainties, at the same time, dominate areas with low input density from the other class. Choosing the nearest points for USVC causes its optimal classifier to be stretched from the target function despite the fact that lower NMCU may be achieved in the training sessions. On the other hand, some sparsity in 50-input training data sets may cost the results of TME. For instance, in the bottom left part of Figure 4.4(b), that single uncertainty can well affect the results of TSVC and USVC. Because of its strategy, TSVC generally performs better than SVC and USVC in the experiments created by Bi and Zhang (2005). However, when the uncertainties near the original boundary tend to move towards the other class, and moreover, are spread evenly around the boundary, SVC and USVC can perform better than TSVC. For example, SVC performs better than USVC and TSVC in Figure 4.5(a), and USVC performs better than SVC and TSVC in Figure 4.5(b).

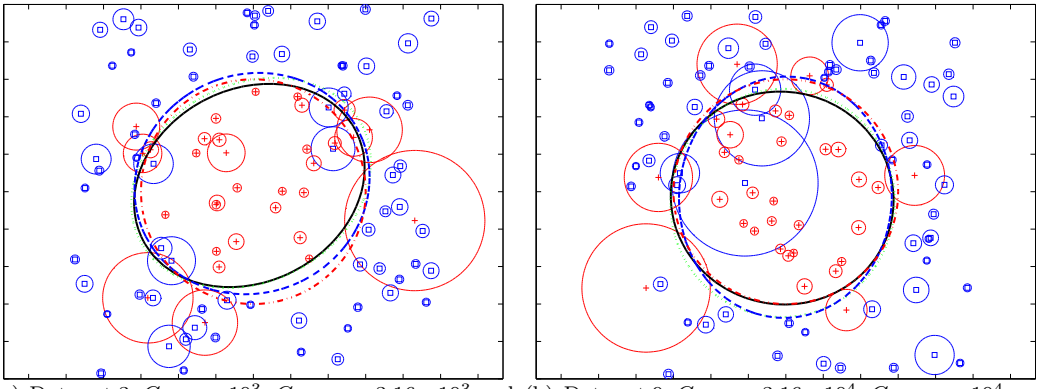


(a) Data set 5, $C_{\mathrm{SVC}} = 100$, $C_{\mathrm{USVC}} = 3.16 \times 10^3$ and $C_{\mathrm{TSVC}} = 3.16 \times 10^2$

(b) Data set 7, $C_{\mathrm{SVC}} = 3.16 \times 10^3$, $C_{\mathrm{USVC}} = 10^5$ and $C_{\mathrm{TSVC}} = 3.16 \times 10^2$

FIGURE 4.6: Selected results from the 5th and 7th 50-input training data sets for classification with quadratic kernel in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.

The results of classification with quadratic kernel are shown in Figure 4.6 and Figure 4.7. Figure 4.6(b) shows the case in which inputs are biased in distribution, this also happens in the top part of Figure 4.6(a). Consequently, there exists some sparsity of training inputs and uncertainties from one class dominate uncertainties from the other class in some areas around the original boundary. Though all algorithms are affected by this contamination, TSVC can better recover the target function with its closest optimal solution to the original boundary. The dominance from one class to the other class in some areas may also cause deviation in TSVC's solution. In the bottom left part of Figure 4.6(a), the optimal classifier is well stretched away from the target function by TSVC's farthest-point strategy. The different results coming out of different strategies implemented in TSVC and USVC are shown in Figure 4.7, especially in the top and bottom parts of Figure 4.7(a) and Figure 4.7(b).



(a) Data set 3, $C_{\text{SVC}} = 10^3$, $C_{\text{USVC}} = 3.16 \times 10^3$ and $C_{\text{TSVC}} = 100$

(b) Data set 9, $C_{\text{SVC}} = 3.16 \times 10^4$, $C_{\text{USVC}} = 10^4$ and $C_{\text{TSVC}} = 3.16 \times 10^4$

FIGURE 4.7: Selected results from the 3rd and 9th 100-input training data sets for classification with quadratic kernel in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.

However, if uncertain inputs are not generated from the rule provided by Bi and Zhang (2005), can TSVC still remain its advantages over other approaches? The training data sets are created by following the previous description except that original inputs from one class no longer move towards the other class under the contamination, which alternatively moves original inputs along traces in parallel with the target function. This contamination does not change the distribution of inputs drastically, but remains relatively high density of uncertainties near the boundary with few means of the distributions of corrupted inputs crossing the original boundary. To show the difference between varied contamination, the same origins with Gaussian noise that have previously been used to reproduce Bi and Zhang's experiment, are inherited to generate the training and the test data sets here. The means and standard errors of NMCU and TME over five test data sets and their corrupted counterparts are reported in Table 4.3 and Table 4.4.

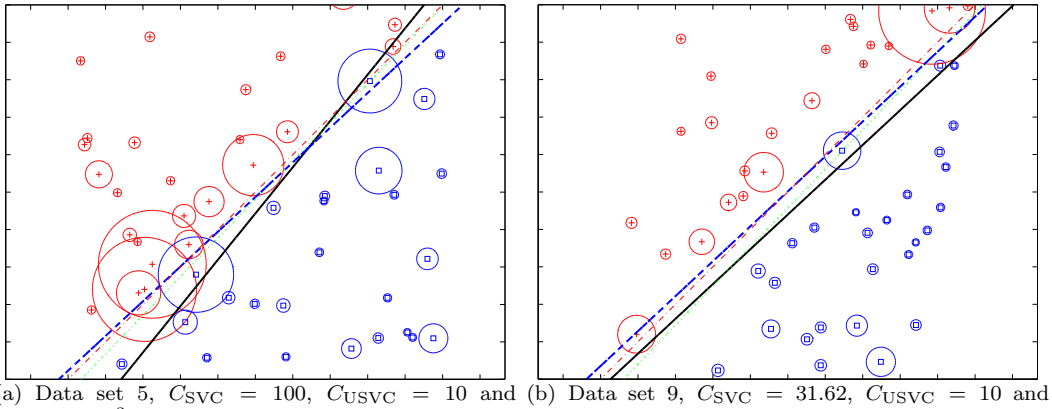
Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	5.70	2.69	3.82	2.18	6.68	1.87	4.58	1.89
SVC	2.88	2.10	1.52	1.18	4.51	1.33	3.07	1.07
TSVC	3.14	2.44	2.40	1.94	4.80	1.70	3.40	1.47

TABLE 4.3: Average test error percentages of NMCU of USVC, TSVC and standard SVC in recomposing Bi and Zhang's experiment, means and standard errors of NMCU are listed in the table.

It is known from the table that TSVC has the overall best performance of all algorithms. Figure 4.8 and Figure 4.9 show the linear case. Under this lighter contamination, all approaches have improved performance. Under its nearest-point strategy, the solution

Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	5.65	2.68	3.78	2.16	6.63	1.88	4.29	1.86
SVC	2.86	2.06	1.52	1.18	4.50	1.37	2.99	1.21
TSVC	3.13	2.44	2.36	1.88	4.85	1.74	3.44	1.70

TABLE 4.4: Average test error percentages of TME of USVC, TSVC and standard SVC in recomposing Bi and Zhang's experiment, means and standard errors of TME are listed in the table.



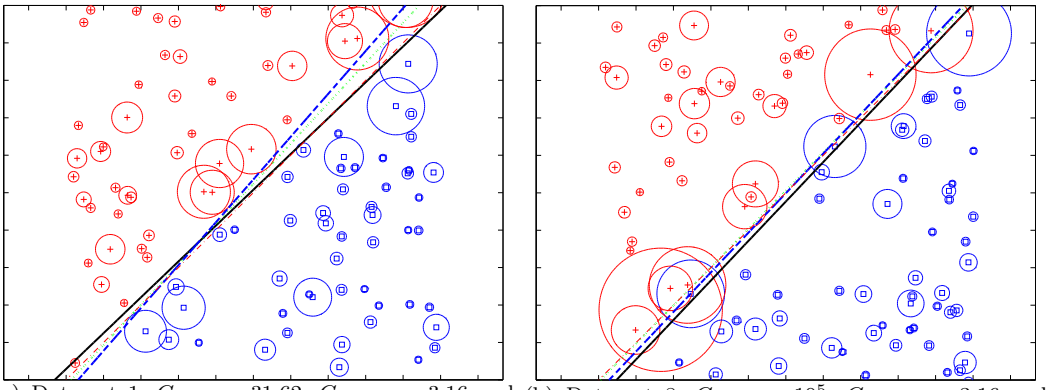
(a) Data set 5, $C_{\text{SVC}} = 100$, $C_{\text{USVC}} = 10$ and $C_{\text{TSVC}} = 10^3$

(b) Data set 9, $C_{\text{SVC}} = 31.62$, $C_{\text{USVC}} = 10$ and $C_{\text{TSVC}} = 31.62$

FIGURE 4.8: Selected results from the 5th and 9th 50-input training data sets for linear classification in the recomposition of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.

of USVC deviates from the target function in the top parts of Figure 4.8(a) and Figure 4.8(b), and the bottom part of Figure 4.8(a), where either some inputs spread in low density or some larger uncertainties from one class dominate other sparse smaller uncertainties from the other class. Meanwhile, the solution of TSVC deviates from the target function by following the opposite direction to the deviation of USVC. Because the corrupted inputs are evenly distributed around the original boundary and few of them cross the boundary after this contamination, choosing the means of inputs' distributions as reference gives SVC the best performance of all approaches. Sometimes, this contamination may help USVC to achieve better performance shown in the top parts of Figure 4.9(a) and Figure 4.9(b).

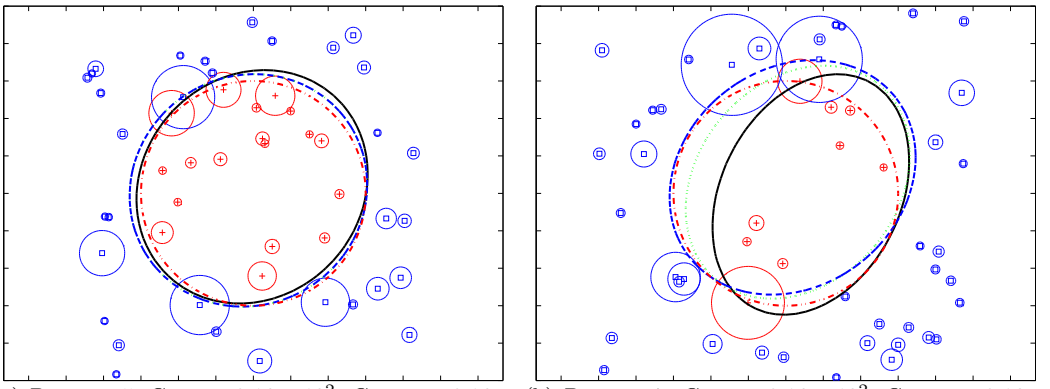
The results of TSVC, SVC and USVC with quadratic kernel functions over the training data sets are shown in Figure 4.10 and Figure 4.11.



(a) Data set 1, $C_{\text{SVC}} = 31.62$, $C_{\text{USVC}} = 3.16$ and $C_{\text{TSVC}} = 3.16 \times 10^4$

(b) Data set 8, $C_{\text{SVC}} = 10^5$, $C_{\text{USVC}} = 3.16$ and $C_{\text{TSVC}} = 10$

FIGURE 4.9: Selected results from the 1st and 8th 100-input training data sets for linear classification in the recomposition of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.



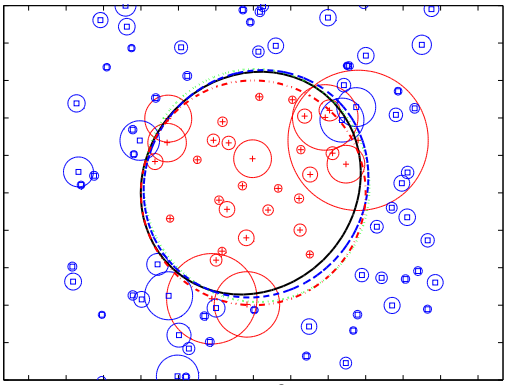
(a) Data set 5, $C_{\text{SVC}} = 3.16 \times 10^2$, $C_{\text{USVC}} = 3.16 \times 10^2$ and $C_{\text{TSVC}} = 3.16 \times 10^2$

(b) Data set 7, $C_{\text{SVC}} = 3.16 \times 10^2$, $C_{\text{USVC}} = 3.16 \times 10^2$ and $C_{\text{TSVC}} = 3.16 \times 10^2$

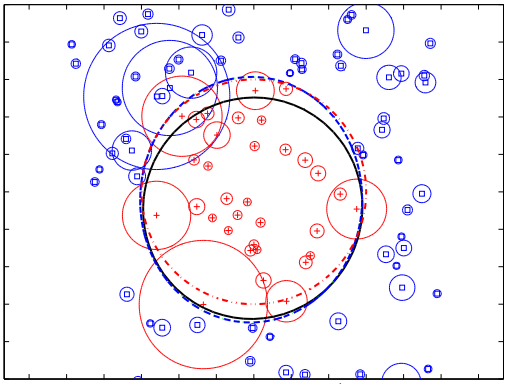
FIGURE 4.10: Selected results from the 5th and 7th 50-input training data sets for classification with quadratic kernel in the recomposition of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.

4.2 Second Order Cone Programming Formulation

Recently, another new classification approach subject to input uncertainty was proposed in Bhattacharyya (2004),Bhattacharyya et al. (2005) and Shivaswamy et al. (2006). This method is called second order cone programming formulation (SOCPF). For continuity and simplicity, Definition 3.1 is introduced to transform SOCPF's optimisation problem



(a) Data set 3, $C_{\text{SVC}} = 10^3$, $C_{\text{USVC}} = 31.62$ and $C_{\text{TSVC}} = 3.16 \times 10^2$



(b) Data set 9, $C_{\text{SVC}} = 3.16 \times 10^4$, $C_{\text{USVC}} = 31.62$ and $C_{\text{TSVC}} = 10^5$

FIGURE 4.11: Selected results from the 3rd and 9th 100-input training data sets for classification with quadratic kernel in the recomposition of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line and green dotted line represents SVC. The target function is illustrated by red dash-dot line.

to traditional formulation. Its linear optimisation formulation is as follows:

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & \sum_{i=1}^{l} \xi_i \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i + r_i\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|, \\ & \|\boldsymbol{w}\| \leq W, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l, \end{aligned} \tag{4.13}$$

where $r_i = \Phi^{-1}(\alpha)$ is the probability confidence defined in (3.4) and (3.7), W is a user-defined constant exploited as an upper bound of $\|\boldsymbol{w}\|$. The constraint $\|\boldsymbol{w}\| \leq W$ comes from an important equivalent formulation of Tikhonov regularisation (Golub et al., 1999), in which the traditional SVC

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & C\sum_{i=1}^{l} \xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i. \end{aligned} \tag{4.14}$$

can be transformed to

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & \sum_{i=1}^{l} \xi_i \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i, \\ & \|\boldsymbol{w}\| \leq W. \end{aligned} \tag{4.15}$$

It can be shown that if $W \leq \|\boldsymbol{w}^*\|$, where $\boldsymbol{w}^*$ is the solution to (4.14) with $\frac{1}{2}\|\boldsymbol{w}\|^2$ removed, then the solutions to (4.14) and to (4.15) are identical for an appropriately

chosen C. So when $r_i = 1$, problem (4.13) is similar to problem (3.10). To simplify the solutions to (4.13), we set $W = \sqrt{C}$ in (4.13). According to Definition 3.1, the non-linear version of the formulation in Bhattacharyya et al. (2005) and Shivaswamy et al. (2006) can be transformed to accommodate the kernel functions. The dual transformation, $\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i$ is introduced in (4.13) instead of $\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{l} (\mathbf{M}_i^{1/2})^T \boldsymbol{\beta}_i$ derived along with USVC. Thus, we have

$$\begin{aligned} \min_{\alpha_i, b, \xi_i} \quad & \sum_{i=1}^{l} \xi_i \\ s.t. \quad & y_i \left(\sum_{j=1}^{l} \alpha_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) + b \right) - 1 + \xi_i \geq r_i \left\| \sum_{j=1}^{l} \alpha_j y_j \mathbf{M}_i^{1/2} \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} \right\|, \\ & \left\| \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\| \leq W, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{4.16}$$

Based on different expressions of $\boldsymbol{w}$ used in optimisation problems, SOCPF considers only the means of the uncertain inputs but ignores the influence of the uncertainties to determine the weight vector $\boldsymbol{w}$ of the optimal hyperplane, while USVC considers both the means $\boldsymbol{x}_i$ and the sizes of the uncertainties $\mathbf{M}_i$ in the classification. In addition, the dual variables $\boldsymbol{\beta}_i$ ignored in SOCPF just like removing the first l second order cone constraints $\|\boldsymbol{\beta_i}\| \leq \alpha_i$ in dual problem (3.30). According to the characteristics of SOCP problem, the dual problem of SOCPF is not strictly feasible (Lobo et al., 1998). As the result, it is not guaranteed that there exist primal and dual feasible points that attain the equal optimal values (Nesterov and Nemirovskii, 1994). But since strictly feasible can be satisfied in the primal problem of SOCPF when a proper optimum is obtained to satisfy the constraints with strict inequality, the optimal value of the primal problem can be equal to that of the dual problem. On the other hand, if the optimal value of the primal problem does not satisfy strictly feasible, then the dual problem of SOCPF is weak duality, in which the optimal values from the primal and dual problems may be different. Whilst, the dual problem of USVC is weak duality only if neither of its primal and dual problems satisfy strictly feasible. Therefore, the dual problem of USVC is more likely to be strong duality that the primal and dual problems achieve the same optimum. Sometimes, USVC and SOCPF can perform at the same level no matter which dual transformation is exploited.

4.3 Minimax Probability Machine

The minimax probability machine (MPM) is a recent method introduced by Lanckriet et al. (2002a,b) and Huang et al. (2004). Unlike the generative approach, in which the

predictor makes distributional assumptions about the class-conditional densities and thereby estimates and controls the relevant probabilities, MPM attempts to control misclassification probabilities in a worst-case setting to choose a discriminative approach by minimising the probabilities that input data fall on the wrong side of the boundary. Before the introduction of MPM, some important definitions and theorems about the optimal bounds in probability are needed.

4.3.1 Optimal Bounds in Probability

Definition 4.6. (Bertsimas and Sethuraman, 2000) A sequence $\bar{\boldsymbol{\sigma}} : (\sigma_k)_{k_1+\ldots+k_n \leq k}$ is a feasible (n, k, Ω)-moment vector (or sequence), if there is a multivariate random variable $\boldsymbol{x} = (x_1, \ldots, x_n)$ with domain $\Omega \subseteq \mathbb{R}^n$, whose moments are given by $\bar{\boldsymbol{\sigma}}$, that is $\sigma_k = E[x_1^{k_1} \ldots x_n^{k_n}], \forall k_1 + \ldots + k_n \leq k$. We say that any such multivariate random variable $\boldsymbol{x}$ has a $\bar{\boldsymbol{\sigma}}$-feasible distribution and denote this as $\boldsymbol{x} \sim \bar{\boldsymbol{\sigma}}$.

Given a sequence $\bar{\boldsymbol{\sigma}}$ of up to kth order moments σ_k of a multivariate random variable $\boldsymbol{x}$ on $\Omega \subseteq \mathbb{R}^n$, our target is to find the "best possible" or "tight" upper bounds on $\Pr(\boldsymbol{x} \in S)$, for arbitrary events $S \subseteq \Omega$.

Definition 4.7. (Bertsimas and Sethuraman, 2000) We say that α is a tight upper bound on $\Pr(\boldsymbol{x} \in S)$, and we will denote it by $\sup_{\boldsymbol{x} \sim \bar{\boldsymbol{\sigma}}} \Pr(\boldsymbol{x} \in S)$ if:

(a) It is an upper bound, i.e., $\Pr(\boldsymbol{x} \in S) \leq \alpha$ for all random variables $\boldsymbol{x} \sim \bar{\boldsymbol{\sigma}}$;

(b) It cannot be improved, i.e., for any $\epsilon > 0$ there is a random variable $\boldsymbol{x}_\epsilon \sim \bar{\boldsymbol{\sigma}}$ for which $\Pr(\boldsymbol{x}_\epsilon \in S) > \alpha - \epsilon$.

The (n, k, Ω)-upper bound problem can be formulated as the following optimisation primal problem:

$$\begin{aligned} Z_P \;=\; \max \;& \int_S f(\boldsymbol{z}) d\boldsymbol{z} \\ s.t. \;& \int_\Omega z_1^{k_1} \ldots z_n^{k_n} f(\boldsymbol{z}) d\boldsymbol{z} = \sigma_k, \quad \forall k_1 + \ldots + k_n \leq k, \\ & f(\boldsymbol{z}) = f(z_1, \ldots, z_n) \geq \mathbf{0}, \quad \forall \boldsymbol{z} = (z_1, \ldots, z_n) \in \Omega. \end{aligned} \tag{4.17}$$

where $f(\boldsymbol{z})$ is a $\bar{\boldsymbol{\sigma}}$-feasible distribution. After introducing the dual variables u_k and constructing the dual polynomial $g(x_1, \ldots, x_n) = \sum_{k_1+\ldots+k_n \leq k} u_k x_1^{k_1} \ldots x_n^{k_n}$, we obtain the dual objective as follows:

$$\sum_k u_k \sigma_k = \sum_k u_k E[x_1^{k_1} \ldots x_n^{k_n}] = E[g(\boldsymbol{x})].$$

The dual problem of (4.17) can be written as:

$$\begin{aligned} Z_D \;=\; \min\;\; & E[g(\boldsymbol{x})] \\ s.t.\;\; & g(\boldsymbol{x})\;\; k\text{-degree, } n\text{-variate polynomial,} \\ & g(\boldsymbol{x}) \geq \chi_S(\boldsymbol{x}),\;\; \forall \boldsymbol{x} \in \Omega. \end{aligned} \tag{4.18}$$

where $\chi_S(\boldsymbol{x})$ is the indicator function of the set S:

$$\chi_S(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in S, \\ 0, & \text{otherwise.} \end{cases}$$

In general, the optimum may not be achievable. Whenever the primal optimum is achieved, we call the corresponding distribution an extremal distribution.

Theorem 4.8. (Weak Duality)(Bertsimas and Sethuraman, 2000) $Z_P \leq Z_D$.

Theorem 4.9. (Strong Duality and Complementary Slackness)(Bertsimas and Sethuraman, 2000) *If the moment vector $\bar{\boldsymbol{\sigma}}$ is an interior point of the set of feasible moment vectors, then the following results hold:*

(a) *Strong Duality: $Z_P = Z_D$.*

(b) *Complementary Slackness: If the dual is bounded, there exists a dual optimal solution $g_{opt}(.)$ and a discrete extremal distribution concentrated on points $\boldsymbol{x}$, where $g_{opt}(\boldsymbol{x}) = \chi_S(\boldsymbol{x})$, that achieves the bound.*

From Theorem 4.9, we know that if strong duality holds, we can obtain a tight bound on $\Pr(\boldsymbol{x} \in S)$ by optimising over the dual problem (4.18).

Now we consider the $(n, 2, \mathbb{R}^n)$-bound problem with the assumptions that the mean vector $\boldsymbol{M} = E[\boldsymbol{x}]$ and the covariance matrix $\Gamma = E[(\boldsymbol{x} - \boldsymbol{M})(\boldsymbol{x} - \boldsymbol{M})^T]$ are known. Given a set $S \subset \mathbb{R}^n$, the tight upper bound of the probability $\Pr(\boldsymbol{x} \in S)$ is derived by the following theorem.

Theorem 4.10. (Bertsimas and Sethuraman, 2000)

(a) *The tight $(n, 2, \mathbb{R}^n)$-upper bound for an arbitrary convex event S is given by:*

$$\sup_{\boldsymbol{x} \sim (\boldsymbol{m}, \Gamma)} \Pr(\boldsymbol{x} \in S) = \frac{1}{1 + d^2}, \tag{4.19}$$

where $d^2 = \inf_{\boldsymbol{x} \in S} (\boldsymbol{x} - \boldsymbol{M})^T \Gamma^{-1} (\boldsymbol{x} - \boldsymbol{M})$, is the squared distance from $\boldsymbol{M}$ to the set S, under the norm induced by the matrix Γ^{-1}

(b) *If $\boldsymbol{M} \notin S$ and if $d^2 = \inf_{\boldsymbol{x} \in S} (\boldsymbol{x} - \boldsymbol{M})^T \Gamma^{-1} (\boldsymbol{x} - \boldsymbol{M})$ is achievable, then there is an extremal distribution that exactly achieves the bound* (4.19)*; otherwise, if $\boldsymbol{M} \in S$*

or if d^2 is not achievable, then there is a sequence of $(\boldsymbol{M}, \Gamma)$-feasible distributions that asymptotically approach the bound (4.19).

Moreover, the result of the upper bound in (4.19) is stronger than the result in Chebyshev's inequality.

4.3.2 Linear Case

The following derivations mainly come from Lanckriet et al. (2002a,b). Let $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ denote random vectors in a binary classification problem, with mean vectors and covariance matrices given by $\boldsymbol{x}_+ \sim (\bar{\boldsymbol{x}}_+, \Sigma_{\boldsymbol{x}_+})$ and $\boldsymbol{x}_- \sim (\bar{\boldsymbol{x}}_-, \Sigma_{\boldsymbol{x}_-})$, and $\boldsymbol{x}_+, \bar{\boldsymbol{x}}_+, \boldsymbol{x}_-, \bar{\boldsymbol{x}}_- \in \mathbb{R}^n$ and $\Sigma_{\boldsymbol{x}_+}, \Sigma_{\boldsymbol{x}_-} \in \mathbb{R}^{n\times n}$. The hyperplane to be determined is $\boldsymbol{w}^T\boldsymbol{z} = b$ ($\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^n$ and $b \in \mathbb{R}$) which separates the two classes of data with maximal probability with respect to all distributions having these means and covariance matrices. The optimisation problem is

$$\begin{aligned} \max_{\alpha,\boldsymbol{w},b} \ \alpha \quad s.t. \quad & \inf_{\boldsymbol{x}_+\sim(\bar{\boldsymbol{x}}_+,\Sigma_{\boldsymbol{x}_+})} \Pr\{\boldsymbol{w}^T\boldsymbol{x}_+ \geq b\} \geq \alpha \\ & \inf_{\boldsymbol{x}_-\sim(\bar{\boldsymbol{x}}_-,\Sigma_{\boldsymbol{x}_-})} \Pr\{\boldsymbol{w}^T\boldsymbol{x}_- \leq b\} \geq \alpha, \end{aligned} \tag{4.20}$$

or equally, we have

$$\begin{aligned} \max_{\alpha,\boldsymbol{w},b} \ \alpha \quad s.t. \ 1-\alpha \geq & \sup_{\boldsymbol{x}_+\sim(\bar{\boldsymbol{x}}_+,\Sigma_{\boldsymbol{x}_+})} \Pr\{\boldsymbol{w}^T\boldsymbol{x}_+ \leq b\} \\ 1-\alpha \geq & \sup_{\boldsymbol{x}_-\sim(\bar{\boldsymbol{x}}_-,\Sigma_{\boldsymbol{x}_-})} \Pr\{\boldsymbol{w}^T\boldsymbol{x}_- \geq b\}, \end{aligned} \tag{4.21}$$

where α is the lower bound of the probability that the input data are classified correctly by the optimal hyperplane. Consider the second constraint in (4.21) and recall the result of (4.19), we have

$$d^2 = \inf_{\boldsymbol{w}^T\boldsymbol{x}_-\geq b} (\boldsymbol{x}_- - \bar{\boldsymbol{x}}_-)^T {\Sigma_{\boldsymbol{x}_-}}^{-1} (\boldsymbol{x}_- - \bar{\boldsymbol{x}}_-) = \frac{\max((b - \boldsymbol{w}^T\bar{\boldsymbol{x}}_-), 0)^2}{\boldsymbol{w}^T\Sigma_{\boldsymbol{x}_-}\boldsymbol{w}}. \tag{4.22}$$

Since $\boldsymbol{w}^T\bar{\boldsymbol{x}}_- \leq b$, we have $\max((b - \boldsymbol{w}^T\bar{\boldsymbol{x}}_-), 0) = b - \boldsymbol{w}^T\bar{\boldsymbol{x}}_-$ and this reduces to

$$b - \boldsymbol{w}^T\bar{\boldsymbol{x}}_- \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^T\Sigma_{\boldsymbol{x}_-}\boldsymbol{w}} \quad \text{where } \kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}. \tag{4.23}$$

Because $\kappa(\alpha)$ is a monotone increasing function of α, κ can be eliminated finally:

$$\min_{\boldsymbol{w}} \sqrt{\boldsymbol{w}^T\Sigma_{\boldsymbol{x}_+}\boldsymbol{w}} + \sqrt{\boldsymbol{w}^T\Sigma_{\boldsymbol{x}_-}\boldsymbol{w}} \quad s.t. \ \boldsymbol{w}(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) = 1, \tag{4.24}$$

or

$$\min_{\boldsymbol{w}} \|\Sigma_{\boldsymbol{x}_+}^{1/2}\boldsymbol{w}\| + \|\Sigma_{\boldsymbol{x}_-}^{1/2}\boldsymbol{w}\| \quad s.t. \ \boldsymbol{w}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-) = 1, \tag{4.25}$$

and

$$b_* = \boldsymbol{w}_*^T \bar{\boldsymbol{x}}_+ - \kappa(\alpha_*)\sqrt{\boldsymbol{w}_*^T \Sigma_{\boldsymbol{x}_+} \boldsymbol{w}_*} = \boldsymbol{w}_*^T \bar{\boldsymbol{x}}_- + \kappa(\alpha_*)\sqrt{\boldsymbol{w}_*^T \Sigma_{\boldsymbol{x}_-} \boldsymbol{w}_*}, \tag{4.26}$$

where $\boldsymbol{w}_*$ and α_* are the optimal values of $\boldsymbol{w}$ and α respectively. The optimisation primal problem (4.25) of the minimax probability machine is a SOCP problem. Like the derivations from (4.17) to (4.18), we introduce the dual vectors $\boldsymbol{u}$, $\boldsymbol{v}$ and the Lagrange multiplier λ. Therefore, problem (4.25) can be expressed as the following constrained problem:

$$\min_{\boldsymbol{w}} \max_{\lambda,\boldsymbol{u},\boldsymbol{v}} \ \boldsymbol{u}^T \Sigma_{\boldsymbol{x}_+}^{1/2} \boldsymbol{w} + \boldsymbol{v}^T \Sigma_{\boldsymbol{x}_-}^{1/2} \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-)) \qquad s.t. \ \|\boldsymbol{u}\| \leq 1, \|\boldsymbol{v}\| \leq 1.$$

and the dual polynomial is constructed as follows:

$$\begin{aligned} g(\boldsymbol{u}, \boldsymbol{v}, \lambda) &= \min_{\boldsymbol{w}} \boldsymbol{u}^T \Sigma_{\boldsymbol{x}_+}^{1/2} \boldsymbol{w} + \boldsymbol{v}^T \Sigma_{\boldsymbol{x}_-}^{1/2} \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-)) \\ &= \begin{cases} \lambda, & \text{if } \lambda \bar{\boldsymbol{x}}_+ - \Sigma_{\boldsymbol{x}_+}^{1/2} \boldsymbol{u} = \lambda \bar{\boldsymbol{x}}_- + \Sigma_{\boldsymbol{x}_-}^{1/2} \boldsymbol{v}, \\ -\infty, & \text{otherwise.} \end{cases} \end{aligned}$$

We obtain the dual problem as

$$\begin{aligned} \max_{\lambda,\boldsymbol{u},\boldsymbol{v}} \ \lambda \qquad s.t. \ \ & \bar{\boldsymbol{x}}_+ + \Sigma_{\boldsymbol{x}_+}^{1/2} \boldsymbol{u} = \bar{\boldsymbol{x}}_- + \Sigma_{\boldsymbol{x}_-}^{1/2} \boldsymbol{v}, \\ & \|\boldsymbol{u}\| \leq \frac{1}{\lambda}, \ \ \|\boldsymbol{v}\| \leq \frac{1}{\lambda}. \end{aligned} \tag{4.27}$$

Geometrically, the minimax approach uses $(\bar{\boldsymbol{x}}_+, \Sigma_{\boldsymbol{x}_+})$ and $(\bar{\boldsymbol{x}}_-, \Sigma_{\boldsymbol{x}_-})$ to construct two ellipsoids which represent the distribution of the input data of the two classes in the binary classification. When λ is small enough, the two ellipsoids intersect. (4.27) amounts to finding the largest λ for which these ellipsoids intersect. Whenever the optimal λ is achieved, the ellipsoids will be tangential to each other. The minimax optimal solution is then the common tangent to both optimal ellipsoids.

4.3.3 Non-Linear Case

Let $\phi(\boldsymbol{x}_+) \sim (\overline{\phi(\boldsymbol{x}_+)}, \Sigma_{\phi(\boldsymbol{x}_+)})$ and $\phi(\boldsymbol{x}_-) \sim (\overline{\phi(\boldsymbol{x}_-)}, \Sigma_{\phi(\boldsymbol{x}_-)})$ denote the data in the feature space, which are mapped by the mapping function $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$ from the input data $\{\boldsymbol{x}_{+i}\}_{i=1}^{N_{\boldsymbol{x}_+}}$ and $\{\boldsymbol{x}_{-i}\}_{i=1}^{N_{\boldsymbol{x}_-}}$ in the training classes. The decision hyperplane in the feature space is $\boldsymbol{w}^T \phi(\boldsymbol{z}) = b$ with $\boldsymbol{w}, \phi(\boldsymbol{z}) \in \mathbb{R}^m$ and $b \in \mathbb{R}$. The optimisation problem in non-linear case is

$$\begin{aligned} \min_{\boldsymbol{w}} \ & \sqrt{\boldsymbol{w}^T \Sigma_{\phi(\boldsymbol{x}_+)} \boldsymbol{w}} + \sqrt{\boldsymbol{w}^T \Sigma_{\phi(\boldsymbol{x}_-)} \boldsymbol{w}} \\ s.t. \ & \boldsymbol{w} \left(\overline{\phi(\boldsymbol{x}_+)} - \overline{\phi(\boldsymbol{x}_-)} \right) = 1. \end{aligned} \tag{4.28}$$

Since no uncertainties are introduced to the input data $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$, $\boldsymbol{w}$ can be assumed and written as

$$\boldsymbol{w} = \sum_{i=1}^{N_{\boldsymbol{x}_+}} \alpha_i \phi(\boldsymbol{x}_{+i}) + \sum_{i=1}^{N_{\boldsymbol{x}_-}} \beta_i \phi(\boldsymbol{x}_{-i}). \tag{4.29}$$

Covariance $\Sigma_{\phi(\boldsymbol{x}_+)}$ can be estimated by

$$\Sigma_{\phi(\boldsymbol{x}_+)} = \frac{1}{N_{\boldsymbol{x}_+}} \sum_{i=1}^{N_{\boldsymbol{x}_+}} \left(\phi(\boldsymbol{x}_{+i}) - \widehat{\phi(\boldsymbol{x}_+)}\right) \left(\phi(\boldsymbol{x}_{+i}) - \widehat{\phi(\boldsymbol{x}_+)}\right)^T,$$

where $\widehat{\phi(\boldsymbol{x}_+)} = \frac{1}{N_{\boldsymbol{x}_+}} \sum_{i=1}^{N_{\boldsymbol{x}_+}} \phi(\boldsymbol{x}_{+i})$. Introducing (4.29) and kernel function $K(\boldsymbol{z}_1, \boldsymbol{z}_2) = \phi(\boldsymbol{z}_1)^T \phi(\boldsymbol{z}_2)$ back into (4.28), we have

$$\begin{aligned} &\min_{\boldsymbol{\gamma}} \sqrt{\frac{1}{N_{\boldsymbol{x}_+}} \boldsymbol{\gamma}^T \widetilde{\mathbf{K}}_{\boldsymbol{x}_+}^T \widetilde{\mathbf{K}}_{\boldsymbol{x}_+} \boldsymbol{\gamma}} + \sqrt{\frac{1}{N_{\boldsymbol{x}_-}} \boldsymbol{\gamma}^T \widetilde{\mathbf{K}}_{\boldsymbol{x}_-}^T \widetilde{\mathbf{K}}_{\boldsymbol{x}_-} \boldsymbol{\gamma}} \\ &s.t.\ \boldsymbol{\gamma}^T (\widetilde{\mathbf{k}}_{\boldsymbol{x}_+} - \widetilde{\mathbf{k}}_{\boldsymbol{x}_-}) = 1, \end{aligned} \tag{4.30}$$

where

$$\begin{aligned} &\boldsymbol{\gamma} = [\alpha_1\, \alpha_2\, \cdots\, \alpha_{N_{\boldsymbol{x}_+}}\, \beta_1\, \beta_2\, \cdots\, \beta_{N_{\boldsymbol{x}_-}}]^T, \\ &\widetilde{\mathbf{k}}_{\boldsymbol{x}_+} \in \mathbb{R}^{N_{\boldsymbol{x}_+}+N_{\boldsymbol{x}_-}}, \quad [\widetilde{\mathbf{k}}_{\boldsymbol{x}_+}]_i = \frac{1}{N_{\boldsymbol{x}_+}} \sum_{j=1}^{N_{\boldsymbol{x}_+}} K(\boldsymbol{x}_{+j}, \boldsymbol{z}_i), \\ &\widetilde{\mathbf{k}}_{\boldsymbol{x}_-} \in \mathbb{R}^{N_{\boldsymbol{x}_+}+N_{\boldsymbol{x}_-}}, \quad [\widetilde{\mathbf{k}}_{\boldsymbol{x}_-}]_i = \frac{1}{N_{\boldsymbol{x}_-}} \sum_{j=1}^{N_{\boldsymbol{x}_-}} K(\boldsymbol{x}_{-j}, \boldsymbol{z}_i), \\ \text{and} \quad &\boldsymbol{z}_i = \boldsymbol{x}_{+i} \quad \text{for } i = 1, 2, \ldots, N_{\boldsymbol{x}_+}, \\ &\boldsymbol{z}_i = \boldsymbol{x}_{-(i-N_{\boldsymbol{x}_+})} \quad \text{for } i = N_{\boldsymbol{x}_+}+1, N_{\boldsymbol{x}_+}+2, \ldots, N_{\boldsymbol{x}_+}+N_{\boldsymbol{x}_-}, \end{aligned}$$

and $\widetilde{\mathbf{K}}$ in (4.30) is defined as:

$$\widetilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_{\boldsymbol{x}_+} - \mathbf{1}_{N_{\boldsymbol{x}_+}} \widetilde{\mathbf{k}}_{\boldsymbol{x}_+}^T \\ \mathbf{K}_{\boldsymbol{x}_-} - \mathbf{1}_{N_{\boldsymbol{x}_-}} \widetilde{\mathbf{k}}_{\boldsymbol{x}_-}^T \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{K}}_{\boldsymbol{x}_+} \\ \widetilde{\mathbf{K}}_{\boldsymbol{x}_-} \end{bmatrix},$$

where $\mathbf{1}_{N_{\boldsymbol{x}_+}}$ is a column vector with ones of dimension $N_{\boldsymbol{x}_+}$. $\mathbf{K}_{\boldsymbol{x}_+}$ and $\mathbf{K}_{\boldsymbol{x}_-}$ contain respectively the first $N_{\boldsymbol{x}_+}$ rows and the last $N_{\boldsymbol{x}_-}$ rows of the Gram matrix $\mathbf{K}$, which is composed by $\mathbf{K}_{ij} = \phi(\boldsymbol{z}_i)^T \phi(\boldsymbol{z}_j) = K(\boldsymbol{z}_i, \boldsymbol{z}_j)$. Once the optimal values γ_* is available,

we can attain the optimal values of κ and b respectively

$$\begin{aligned}\kappa_* &= \frac{1}{\sqrt{\frac{1}{N_{\boldsymbol{x}_+}}\boldsymbol{\gamma}_*^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_+}^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_+}\boldsymbol{\gamma}_*} + \sqrt{\frac{1}{N_{\boldsymbol{x}_-}}\boldsymbol{\gamma}_*^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_-}^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_-}\boldsymbol{\gamma}_*}}, \\ b_* &= \boldsymbol{\gamma}_*^T\widetilde{\mathbf{k}}_{\boldsymbol{x}_+} - \kappa_*\sqrt{\frac{1}{N_{\boldsymbol{x}_+}}\boldsymbol{\gamma}_*^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_+}^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_+}\boldsymbol{\gamma}_*} \\ &= \boldsymbol{\gamma}_*^T\widetilde{\mathbf{k}}_{\boldsymbol{x}_-} + \kappa_*\sqrt{\frac{1}{N_{\boldsymbol{x}_-}}\boldsymbol{\gamma}_*^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_-}^T\widetilde{\mathbf{K}}_{\boldsymbol{x}_-}\boldsymbol{\gamma}_*},\end{aligned} \tag{4.31}$$

where κ_* and b_* are the optimum of κ and b.

4.4 Summary

This chapter has surveyed three different approaches to classification using information about uncertain inputs. TSVC and SOCPF are derived from statistical methods, and their resulting optimisation problems can be regarded as extensions of traditional SVC. Besides, we kernelised TSVC and SOCPF by introducing new kernelisation method derived in USVC. Though SOCPF does not generate a strictly feasible dual problem since the dual variables related to uncertainties are ignored in dual transformation, it can sometimes perform at the same level as USVC by attaining the optimum with its strictly feasible primal problem. Generally, USVC is more likely to reach the optimum for both the primal and dual problem than SOCPF in accordance with their different dual optimisation problems.

Geometrically, with the probability confidence $r = 1$, USVC exploits the nearest points in the distributions (ellipsoids) of uncertain inputs as the reference to obtain the optimal hyperplane. On the contrary, the farthest points in the distributions (ellipsoids) are used to incorporate information of uncertainties into TSVC by equivalently setting the probability confidence $r = -1$. Statistically, for those uncertain inputs which follow multivariate Gaussian distributions, choosing the farthest points leads to lower probabilities that the unknown original inputs are going to be correctly classified by the optimal solution of TSVC. USVC is rather conservative by choosing the nearest points in ellipsoids to optimal classifier. Both USVC wants to guarantee higher probabilities that the unknown original inputs will be correctly classified in case their corrupted counterparts are set to be discriminated right. These probabilities relate to the distributions of uncertain inputs and the optimal classifier. Comparatively, SVC reaches a compromise between TSVC and USVC. Experimental results have illustrated that neither TSVC nor SVC has advantages over each other in all situations, and USVC generally performs worse than TSVC and SVC. Indeed, the performance of these approaches depends on their different strategies and the contamination.

Unlike USVC or TSVC, MPM considers a different case of the uncertain inputs, in which input data are not contaminated by noise individually but are considered together in each class without prior knowledge of distribution. MPM can efficiently extend its assumption from Gaussian distribution to other non-preferred prior distribution, which can be applied to generate different probability confidence r in (3.7).

Chapter 5

Iterative Constraints in Classification Subject to Input Uncertainty

In classification subject to input uncertainty, the traditional SVC has been extended to incorporate uncertain information. Based upon SVM, many approaches have been created to accommodate the uncertainties. Among these approaches, USVC has been developed with a new kernelisation formulation in Chapter 3. Some other recent SVM-based algorithms were also presented in Chapter 4, such as TSVC and SOCPF. However, these methods have limitations. USVC, along with SOCPF, is designed to evaluate the future contaminated data by its optimal solution obtained from corrupted training data sets. On the contrary, TSVC is designed to recover the true target function from the same corrupted training data sets. We have illustrated in Section 4.1.4 that under some circumstances, one representation is preferable to others and vice versa. Therefore, to achieve a better performance, an algorithm may be improved by allowing it greater control over the classification.

5.1 Adaptive Uncertainty Support Vector Classification

TSVC was introduced in the thesis not only because it is developed based on SVM, but also because TSVC has similar optimisation formulations as USVC. In fact, as previously shown in Figure 4.3, USVC chooses the nearest points to the optimal classifier in covariance ellipsoids of uncertain inputs by setting the probability confidence to +1, which leads to higher predicted probabilities that the unknown and uncontaminated original data will be correctly classified by USVC if their contaminated counterparts can be classified correctly. While through selecting the farthest points, the probability

confidence of TSVC is fixed to -1, which can only guarantee lower probabilities for the original data to be discriminated correctly.

Theoretically, USVC more suits discriminating contaminated inputs of different classes because its optimal solution is obtained by classifying contaminated inputs with their own distributions as much as possible, or alternatively through retaining high predicted probabilities for unknown original data as far as possible in train sessions. However, USVC is difficult to recover the true target function as the original inputs are unable to predict from the distributions of uncertain inputs. Under the assumption that noise tends to make inputs of different classes overlap, TSVC can better recover the target function. This result is illustrated in Section 4.1.4. Furthermore, setting high values to the probabilities that original inputs are set to be correctly classified can cause problems in USVC. Some contaminated areas of low input density of one class are dominated by some larger uncertainties from the other class. USVC inevitably deviates from the target function by these areas close to the original boundary. Since the larger uncertainties may appear in different areas even the contaminated training and test sets follow the same rule to generate, this deviation can seriously affect the performance of USVC in NMCU. This difference is also reflected in Section 4.1.4. So the configuration of the probability confidence needs to be reconsidered.

The similar structures make it possible to combine TSVC and USVC, finding proper probability confidence for USVC by recalling (3.4) and (3.7). In other words, different sizes and positions of uncertainties also affect the predicted probabilities that corresponding original inputs are set to be correctly classified. The classification subject to input uncertainty can benefit from lowering these probabilities and theoretically achieve a better performance than USVC. Recalling problem (4.5), the primal problem can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\xi_i \\ s.t. \quad & r_i\|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \leq y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i, \\ & \xi_i \geq 0 \qquad i = 1,\ldots,l, \end{aligned} \tag{5.1}$$

where $r_i \in [0,1] \quad i = 1,\ldots,l$ are the probability confidence for every single input. r_i are chosen individually for each uncertain input instead of selecting the probability confidence $r = 1$ for all uncertainties in USVC. To distinguish r_i from r, we name r_i the individual probability confidence. In fact, r_i perform as factors to relax the constraints separately without influencing other uncertainties. If considering the statistical interpretation of the probability confidence, we should choose $r_i \in [-1,1]$ based on TSVC and USVC. But when $r_i \in [-1,0)$, as we have proved before, (5.1) is no longer a convex optimisation. Though it can be transformed to a SOCP problem through Tikhonov regularisation (Bi and Zhang, 2005), problem (5.1) will become more complicated. More

importantly, as an extension of USVC, this new method want to attain improved performance in NMCU. If $r_i \in [-1, 0)$, the predicted probabilities of original data being correctly classified are too low to theoretically guarantee that the original data will be discriminated correctly.

Like USVC, the dual problem of (5.1) can also be extended to non-linear case through the kernelisation formulation introduced in Chapter 3 (see Appendix A for more details).

$$\begin{aligned}
\max_{\alpha_i, \boldsymbol{\beta}_i} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right. \\
& + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i y_i \left[\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_j} \right]^T (\mathbf{M}_j^{1/2})^T \boldsymbol{\beta_j} + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_j y_j \boldsymbol{\beta_i}^T \mathbf{M}_i^{1/2} \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} \\
& \left. + \sum_{i=1}^{l} \sum_{j=1}^{l} \boldsymbol{\beta_i}^T \mathbf{M}_i^{1/2} \frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} (\boldsymbol{M}_j^{1/2})^T \boldsymbol{\beta_j} \right) \\
s.t. \quad & \|\boldsymbol{\beta_i}\| \leq r_i \alpha_i \quad i = 1, \ldots, l, \\
& \sum_{i=1}^{l} \alpha_i y_i = 0, \\
& 0 \leq \alpha_i \leq C \quad i = 1, \ldots, l,
\end{aligned} \tag{5.2}$$

Comparing problems (3.30) and (5.2), we see that individual probability confidence r_i are introduced in the first l constraints of the dual problem. In (5.2), r_i actually reduces the effects of dual variables $\boldsymbol{\beta}_i$ upon computation of $\boldsymbol{w}$, which also leads to some decrease in the influence of the uncertainties $\mathbf{M}_i$ on determining the position and direction of $\boldsymbol{w}$. In general, individual probability confidence r_i diminish the effects of uncertainties on the optimal solution.

Since the exact value of r_i for each uncertain input is unknown, an iterative algorithm is proposed to vary individual probability confidence by following the rule that a small scalar will not be deducted from the individual probability confidence unless the mean of its corresponding uncertain input is misclassified by the obtained classifier. This new algorithm is termed adaptive uncertainty support vector classification (AUSVC), which is named after its iteratively varied individual probability confidence. Geometrically, AUSVC is an iterative method of searching for optimal solution from the nearest points to the central points in uncertainty ellipsoids.

In iterative algorithm, every change of individual probability confidence r_i solely depends on the classification result of its corresponding uncertain input in previous iteration. However, if some errors happen in one step, these errors can be amplified in the following steps. This leads to an overfitting problem of training inputs and makes generalisation more difficult to achieve, which is caused by an extra degree of freedom

introduced by tuning individual probability confidence r_i at each training point. Therefore, an extra rule is introduced to reduce the overfitting problem. Some inputs that are correctly classified for certain will never change their corresponding individual probability confidence in iterations and fix them to 1, which can avoid amplifying the errors generated by the overfitting in future iterations. The iterative algorithm of AUSVC is shown below,

Algorithm 3 AUSVC

1. Initialise $r_i = 1$, $i = 1, \ldots, l$, solve (5.2), which indeed is USVC optimisation, for the parameters α_j, $\boldsymbol{\beta}_j$, and b;
2. Substitute the obtained parameters α_j, $\boldsymbol{\beta}_j$, b and the training inputs $(\boldsymbol{z}_i, y_i)$ into

$$g(\boldsymbol{z}_i, y_i) = y_i \left(\sum_{j=1}^{l} \alpha_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) + \sum_{j=1}^{l} \boldsymbol{\beta}_j^T \mathbf{M}_j^{1/2} \frac{\partial K(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\partial \boldsymbol{x}_j} + b \right), \tag{5.3}$$

$$\text{where} \qquad \boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{x}_i, \mathbf{M}_i) \qquad \text{(from Definition 3.1)}$$

respectively to decide which inputs are correctly classified for certain by USVC, and split the training set $\mathcal{D}$ into two sets, set of certainly correctly classified inputs $\mathcal{D}_c = \{(\boldsymbol{z}_i, y_i) | \text{sgn}(g(\boldsymbol{z}_i, y_i) - 2) \geq 0, (\boldsymbol{z}_i, y_i) \in \mathcal{D}\}$ and set of not certainly correctly classified inputs $\mathcal{D}_u = \{(\boldsymbol{z}_i, y_i) | \text{sgn}(g(\boldsymbol{z}_i, y_i) - 2) < 0, (\boldsymbol{z}_i, y_i) \in \mathcal{D}\}$. The individual probability confidence r_i will never be modified again for inputs in $\mathcal{D}_c$, otherwise, a predefined positive scalar (normally 0.1) is deducted from r_i for inputs in $\mathcal{D}_u$. Repeat the following three steps (iteration 3 to 5) until $r_{inew} = r_i$, $i = 1, \ldots, l$;
3. Fix r_i, $i = 1, \ldots, l$ to the current value, solve (5.2) for the parameters α_j, $\boldsymbol{\beta}_j$, and b;
4. Substitute the obtained parameters α_j, $\boldsymbol{\beta}_j$, b and the training inputs $(\boldsymbol{z}_i, y_i)$ into (5.3) respectively to determine whether the inputs are misclassified, $\text{sgn}(g(\boldsymbol{z}_i, y_i)) < 0$ or correctly classified, $\text{sgn}(g(\boldsymbol{z}_i, y_i)) \geq 0$. If the inputs are misclassified and $(\boldsymbol{z}_i, y_i) \in \mathcal{D}_u$, the predefined positive scalar is deducted from its individual probability confidence r_i, otherwise, their r_i remain unchanged. All changed and unchanged individual probability confidence are saved in r_{inew};
5. If $r_{inew} = r_i$, the optimal results of α_i, $\boldsymbol{\beta}_i$, and b are achieved, otherwise, assign r_{inew} to r_i and return to step 3;

The uncertain inputs in $\mathcal{D}_c$ attain higher predicted probabilities of their original counterparts being correctly classified. These probabilities are $\Pr\left\{\boldsymbol{x} \leq \boldsymbol{x}_i + r_i \frac{\mathbf{M}_i \boldsymbol{w}}{\|\mathbf{M}_i^{1/2} \boldsymbol{w}\|}\right\}$, or alternatively denoted by $\Pr\left\{r_i \leq \frac{y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)}{\|\mathbf{M}_i^{1/2} \boldsymbol{w}\|}\right\}$ because the results are definite when $\boldsymbol{w}$ is determined.

The first two iterations in Algorithm 3 are preconditioning of its iterative last three steps, step 3 to step 5. In fact, we use 2 as a threshold for $g(\boldsymbol{z}_i, y_i)$ to discriminate uncertain inputs instead of selecting a proper p, such as $p = 0.99$, as a threshold for the probabilities of the corresponding original inputs being correctly classified. This is because some close-boundary uncertain inputs with small uncertainties have larger predicted probabilities than the preset p, keep those uncertain inputs in $\mathcal{D}_u$ can help SVM-based AUSVC decide support vectors for its optimal solution.

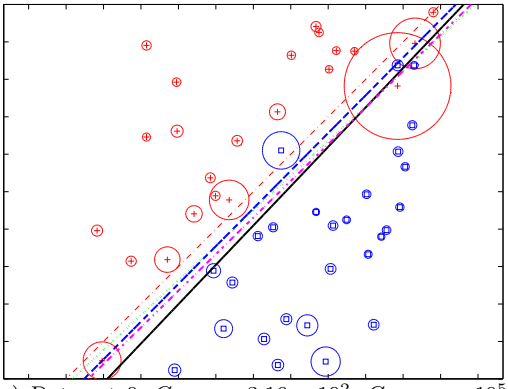
Since the objective function of (5.2) is the same as that of (3.30), the memory requirement of AUSVC is of order $O\left(l^2n^2\right)$. Like the iterative algorithm TSVC, the computational time of AUSVC is related to the number of iterations before the convergence of r_i, which depends on the training data set. Without loss of generality, L_a denotes the number of iterations of AUSVC. In every single iteration, the computational cost consists of the cost of constructing matrix $\mathbf{R}$ in objective function and the optimisation complexity of solving (5.2), plus the cost of changing data-driven individual probability confidence r_i by computing (5.3). In each iteration, the complexities of constructing $\mathbf{R}$ and optimisation in (5.2) are of the same order as those of USVC. Like the cost of constructing auxiliary parameters in TSVC, the cost of changing r_i is of order $O\left(l^2n^2\right)$. Therefore, the computational complexity of AUSVC makes up of the cost of constructing matrix $\mathbf{R}$ with order $O\left(L_al^2n^3\right)$, the optimisation complexity bounded by $O\left(L_al^{3/2}n^3\right)$ and the cost of constructing individual probability confidence r_i with order $O\left(L_al^2n^2\right)$.

To test the performance of AUSVC, the experiment reproduced from Bi and Zhang (2005) in Section 4.1.4 is used here with the same training and test sets created in 4.1.4. In the experiment, AUSVC is trained by stratified 10-fold cross-validation. For convenience, only the results of AUSVC are listed in Table 5.1. The linear case and the non-linear case with quadratic kernel are shown in Figure 5.1 and Figure 5.2. In

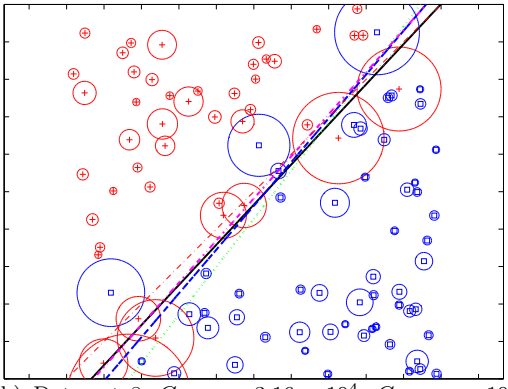
Measure / Measures	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
NMCU	12.32	1.27	11.81	0.84	15.67	3.10	13.80	0.82
TME	6.26	2.81	4.56	2.37	8.24	4.56	4.94	2.25

TABLE 5.1: Average test error percentages of NMCU and TME of AUSVC in reproducing Bi and Zhang's experiment, means and standard errors of NMCU and TME are listed in the table.

comparison of Table 4.1, 4.2 and Table 5.1, AUSVC generally performs better than USVC. Except for the 50-input linear classification, AUSVC also has a better performance than SVC. This is understandable as part of all individual probability confidence r_i vary from 1 to 0 in AUSVC according to the feedback of the classification situation of each uncertain input. The change of r_i can lower the predicted probabilities of original inputs being correctly classified for those uncertain inputs whose uncertainty ellipsoids may finally not be wholly correctly classified by the optimal solution. These adaptive r_i can give AUSVC advantages over USVC, which keeps higher probabilities of theoretically correctly classifying original inputs for their corresponding uncertain inputs by fixing all individual probability confidence equally to 1. Decreasing individual probability confidence r_i of those uncertain inputs which are not certainly correctly classified allows AUSVC to reduce the optimal solution's deviation from the target function that is caused by some sparsity of uncertain inputs close to the original boundary. This situation happens in the bottom left part of Figure 5.1(a) and the left part of Figure 5.2(a). Due to characters that only the means of uncertainties are introduced in



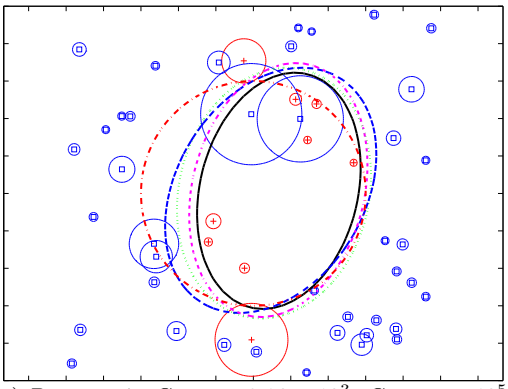
(a) Data set 9, $C_{\text{SVC}} = 3.16 \times 10^2$, $C_{\text{USVC}} = 10^5$, $C_{\text{TSVC}} = 10$ and $C_{\text{AUSVC}} = 10$



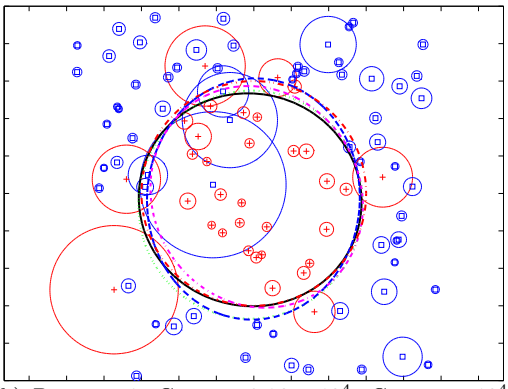
(b) Data set 8, $C_{\text{SVC}} = 3.16 \times 10^4$, $C_{\text{USVC}} = 10$, $C_{\text{TSVC}} = 10$ and $C_{\text{AUSVC}} = 10$

FIGURE 5.1: Selected results from the 9th 50-input and 8th 100-input training data sets for linear classification in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line, SVC is represented by green dotted line and magenta dash-dot line represents AUSVC. The target function is illustrated by the thin red dash-dot line.

classification, the obtained classifier of SVC sometimes deviates from the target function in some close-boundary areas where larger uncertainties from one class dominate smaller uncertainties from the other class, such as the bottom left part of Figure 5.1(b) and the bottom part of Figure 5.2(b). Meanwhile, this situation can also cause higher



(a) Data set 7, $C_{\text{SVC}} = 3.16 \times 10^3$, $C_{\text{USVC}} = 10^5$, $C_{\text{TSVC}} = 3.16 \times 10^2$ and $C_{\text{AUSVC}} = 10^3$



(b) Data set 9, $C_{\text{SVC}} = 3.16 \times 10^4$, $C_{\text{USVC}} = 10^4$, $C_{\text{TSVC}} = 3.16 \times 10^4$ and $C_{\text{AUSVC}} = 3.16 \times 10^4$

FIGURE 5.2: Selected results from the 7th 50-input and 9th 100-input training data sets for classification with quadratic kernel in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line, SVC is represented by green dotted line and magenta dash-dot line represents AUSVC. The target function is illustrated by the thin red dash-dot line.

NMCU in SVC than in other approaches shown in Table 4.1, 4.2 and Table 5.1, because the central points of ellipses can not entirely reflect the uncertain information, whereas which is fully incorporated in different ways in USVC, TSVC and AUSVC.

5.2 Minimax Probability Support Vector Classification

AUSVC has benefited from adaptive individual probability confidence r_i in constraints with a generally improved performance over USVC. However, the adaptive r_i also introduce additional degrees of freedom in optimisation, which may overfit training inputs by adjusting several predicted probabilities of original inputs being correctly classified in error even some individual probability confidence have been preconditioned in the iterative algorithm of AUSVC. Some mistakenly lowered r_i in previous iterations may lead to further errors in adjusting r_i afterwards, either mistakenly lowering r_i that should remain unchanged or mistakenly keeping r_i invariable that should be lowered. Therefore, some additional constraints are needed to deal with r_i.

5.2.1 Uncertain Inputs without Prior Distribution Information

In Section 4.3, MPM was introduced as a discriminative solution to control misclassification probabilities in a worst-case setting by minimising the probabilities that inputs fall on the wrong side of the boundary. Due to their similar statistical models, we then can extend the knowledge from MPM (Lanckriet et al., 2002a,b) to SVM-based classification methods subject to input uncertainty, USVC, TSVC and AUSVC, which all incorporate the probabilities that the original inputs are correctly classified with fixed or varied individual probability confidence r_i.

Recalling Definition 3.1 except that the distributions of uncertain inputs $\boldsymbol{z}_i$ are unknown, we can replace the hyperplane to be determined in MPM, $\boldsymbol{w}^T\boldsymbol{z} = b$ with the optimal solution used in SVM-based approaches, $y_i(\boldsymbol{w}^T\boldsymbol{z}_i + b) - 1 + \xi_i = 0$. Consequently, each uncertain input $\boldsymbol{z}_i$ is separated by maximising this correct classification probability α with respect to $\boldsymbol{z}_i \sim (\boldsymbol{x}_i, \mathbf{M}_i)$, where $\boldsymbol{x}_i$ and $\mathbf{M}_i$ are means and covariance matrices of $\boldsymbol{z}_i$ respectively. The optimisation problem can be expressed as:

$$\begin{aligned} \max_{\alpha,\boldsymbol{w},b} \quad & \alpha \\ s.t. \quad & \inf_{\boldsymbol{z}_i\sim(\boldsymbol{x}_i,\mathbf{M}_i)} \Pr\{y_i(\boldsymbol{w}^T\boldsymbol{z}_i + b) - 1 + \xi_i \geq 0\} \geq \alpha, \\ & i = 1, \ldots, l. \end{aligned} \tag{5.4}$$

From Theorem 4.10, which was proposed with convex optimisation techniques by Bertsimas and Sethuraman (2000), the probability of misclassified uncertain input is bounded by:

$$\begin{aligned} &\sup_{\boldsymbol{z}_i\sim(\boldsymbol{x}_i,\mathbf{M}_i)} \Pr\{y_i(\boldsymbol{w}^T\boldsymbol{z}_i + b) - 1 + \xi_i \leq 0\} = \frac{1}{1+d^2}, \\ &\text{with} \quad d^2 = \inf_{y_i(\boldsymbol{w}^T\boldsymbol{z}_i+b)-1+\xi_i\leq 0} (\boldsymbol{z}_i - \boldsymbol{x}_i)^T\mathbf{M}_i^{-1}(\boldsymbol{z}_i - \boldsymbol{x}_i), \end{aligned} \tag{5.5}$$

which shows that the maximal probability of the uncertain input being misclassified is determined by parameter d, which denotes the distance between the part of $\boldsymbol{z}_i$'s distribution that is misclassified by the soft margins of optimal classifier and $\boldsymbol{z}_i$'s mean $\boldsymbol{x}_i$. The minimal distance d admits a simple closed-form expression:

$$\begin{aligned} d^2 &= \inf_{y_i(\boldsymbol{w}^T\boldsymbol{z}_i+b)-1+\xi_i\leq 0} (\boldsymbol{z}_i-\boldsymbol{x}_i)^T\mathbf{M}_i^{-1}(\boldsymbol{z}_i-\boldsymbol{x}_i) \\ &= \frac{\max\left((y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1+\xi_i),0\right)^2}{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}, \end{aligned} \tag{5.6}$$

and the ith constraint of (5.4) can be transformed equivalently to

$$\sup_{\boldsymbol{z}_i\sim(\boldsymbol{x}_i,\mathbf{M}_i)} \Pr\{y_i(\boldsymbol{w}^T\boldsymbol{z}_i+b)-1+\xi_i\leq 0\} \leq 1-\alpha. \tag{5.7}$$

Combining (5.5) and (5.7), we have

$$1-\alpha \geq \frac{1}{1+d^2}, \qquad \text{or} \qquad d^2 \geq \frac{\alpha}{1-\alpha}.$$

If $y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1+\xi_i\geq 0$, we have

$$\max\left((y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1+\xi_i),0\right) = y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1+\xi_i,$$

in (5.6), which indeed reduces to

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1+\xi_i \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}, \qquad \text{where} \quad \kappa(\alpha)=\sqrt{\frac{\alpha}{1-\alpha}}. \tag{5.8}$$

Comparing (5.8) and the first l constraints in (5.1), we see that these inequalities are similar to each other except that the individual probability confidence $r_i\in[0,1]$ is replaced by the cumulative distribution function $\Phi^{-1}(\alpha)$ and $\kappa(\alpha)$ respectively, when the uncertain inputs follow the Gaussian distribution or no prior distribution information is available. Both $\Phi^{-1}(\alpha)$ and $\kappa(\alpha)$ are monotonically increasing, and the comparison of individual probability confidence between single-dimensional Gaussian distribution and no prior distribution is shown in Figure 5.3. The decreasing r_i can be explained as controlling probability α which holds the lower bound of probability of an uncertain input being correctly classified in theory. $r_i=1$ implies that $\alpha=\Phi(r_i)=\frac{\mathrm{erf}(\frac{1}{\sqrt{2}})+1}{2}=0.84$ in one-dimensional USVC, where $\mathrm{erf}=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ is error function. In one-dimensional SVC, $\alpha=0.5$ is derived by $r_i=0$. In Figure 5.3, we see that Gaussian distribution leads to a lower individual probability confidence r_i than no prior assumed distribution when probability α is set in advance. This comparison can also be generalised to multi-dimensional circumstances, in which $\alpha=\Phi_{\mathrm{mv}}(\boldsymbol{x}_i+r_i\frac{\mathbf{M}_i\boldsymbol{w}}{\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|})$ holds the lower bound of

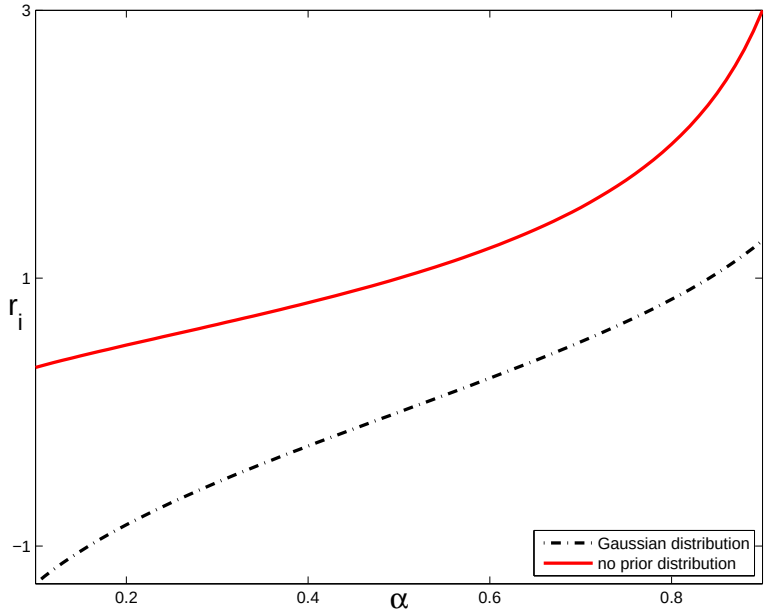


FIGURE 5.3: Comparison of individual probability confidence between Gaussian distribution and no prior assumption of distribution in one-dimensional input space.

correct classification probability of a multi-dimensional uncertain input following multivariate Gaussian distribution, where Φ_{mv} is a multivariate normal cumulative distribution function. Meanwhile, even the distributions that uncertain inputs follow are unknown, they still can be considered in ellipsoids centred around their means with shapes determined by their covariance matrices, we can use derivations from (3.2) to get the data at which the predicted probabilities need to compute. Therefore, the lower bound of a multi-dimensional uncertain input being correctly classified is $\alpha = \kappa_{\text{mv}}^{-1}(r_i \frac{\mathbf{M}_i \boldsymbol{w}}{\|\mathbf{M}_i^{1/2} \boldsymbol{w}\|})$ when the prior distribution of this input is not available, where κ_{mv} denotes a multivariate version of κ function. When the individual probability confidence r_i, the distributional information $\boldsymbol{x}_i$, $\mathbf{M}_i$ and the weight vector $\boldsymbol{w}$ are the same, the only difference lies in the value of the worst-case misclassification probability $1 - \alpha$ associated with r_i, α will be higher under Gaussian assumption, so the hyperplane will have higher predicted probabilities of classifying those unknown original data correctly. The comparison of correct classification probabilities associated with r_i between Gaussian assumption and no prior assumption is shown in Figure 5.4, which is the reproduction of Figure 4.3(a) with the case that the prior knowledge of distributions of those uncertain inputs is unknown. Please note that the multivariate function κ_{mv}^{-1} is not needed, thereby is not derived in this thesis.

In Figure 5.4, one-dimensional function κ^{-1} is used to simulate κ_{mv}^{-1} by following the direction determined by $\mathbf{M}_i$ and $\boldsymbol{w}$. Like the changes between multivariate and single variate Gaussian cumulative distribution function, the results obtained from κ^{-1} are also relaxed to fit κ_{mv}^{-1}. We see that the predicted probability under Gaussian assumption is larger than the predicted probability without prior knowledge, which is illustrated by the fact that the three-dimensional shaded surface of κ_{mv}^{-1} stays well beneath the surface

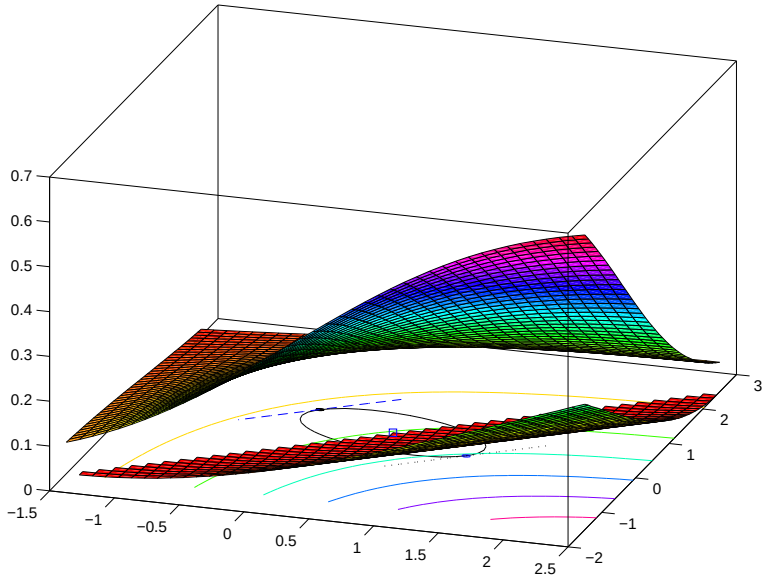


FIGURE 5.4: Comparison of predicted probability associated with individual probability confidence under Gaussian distribution and no prior assumption of distribution in two-dimensional input space. The weight vector of the optimal classifier is $\boldsymbol{w} = (1; -1)$. Under Gaussian assumption, $\boldsymbol{z}_i \sim \mathcal{N}\left((0.5; 0.5), (1.143, -0.286; -0.286, 0.571)\right)$ and $\boldsymbol{z}_i \sim \left((0.5; 0.5), (1.143, -0.286; -0.286, 0.571)\right)$ when the assumption of distribution is unavailable. The upper 3D shaded surface and the contour beneath surface represent multivariate normal cumulative distribution function Φ_{mv} and multivariate function κ_{mv}^{-1} is illustrated by the lower 3D shaded surface.

of Φ_{mv} in Figure 5.4. The extra knowledge resulting from the Gaussian assumption allows us to predict a higher probability of correct classification over unknown original data.

5.2.2 Minimax Probability Support Vector Classification

MPM provides a feasible way to extend the existing and developed SVM-based classification approaches subject to input uncertainty which are generated under the assumption that uncertain inputs follow Gaussian distributions to accommodate the information of uncertainties when no knowledge of their distributions is available. The drawback is that larger individual probability confidence r_i are required in the constraints of (5.1) when the distributions of uncertain inputs are unknown. While the same predicted probabilities of correctly classifying original data can be attained as those probabilities are applied by smaller r_i under Gaussian assumption. As what we have discussed before, large individual probability confidence can deteriorate the performance of classifiers. However, adaptively decreasing individual probability confidence in AUSVC also has overfitting problem that is resultantly brought by an extra degree of freedom introduced by the interpolation of individual probability confidence at each training point. Therefore, some additional constraints are needs to cope with these extra degrees of freedom.

MPM not only derives a discriminative method to classify inputs without prior knowledge of their distributions, but also can compare the performance of different algorithms with a new measure, which adds up the possible maximal misclassification probabilities of all uncertain inputs. Since this measure comes from MPM, it is named minimax probability error (MPE) after MPM. This error comes from the upper bound of misclassification probability (Bertsimas and Sethuraman, 2000):

$$\begin{aligned} &\sup_{\boldsymbol{z}_i \sim (\boldsymbol{x}_i, \mathbf{M}_i)} \Pr\{y_i(\boldsymbol{w}^T \boldsymbol{z}_i + b) \leq 0\} = \frac{1}{1+d^2}, \\ &\text{with} \quad d^2 = \inf_{y_i(\boldsymbol{w}^T \boldsymbol{z}_i + b) \leq 0} (\boldsymbol{z}_i - \boldsymbol{x}_i)^T \mathbf{M}_i^{-1} (\boldsymbol{z}_i - \boldsymbol{x}_i). \end{aligned} \tag{5.9}$$

Traditional penalties $\sum \xi_i$ which comes from margin errors rather than misclassification are incorporated in the optimisation problems of previous SVM-based classification approaches. Depending on the regularisation parameter C, penalties $\sum \xi_i$ may be made up of the margin errors from part of training inputs. While MPE comprises worst-case misclassification probabilities of all uncertain inputs in a training set no matter whether these inputs will be correctly classified or not by the optimal solution. MPM measures the performance of different algorithms by including the contribution from all uncertain inputs. Thus a new optimisation problem is formulated as below,

$$\begin{aligned} &\min_{\boldsymbol{w}, b} \sum_{i=1}^{l} \frac{1}{1+d_i^2} \\ &s.t.\ d_i^2 = \begin{cases} \frac{(\boldsymbol{w}^T \boldsymbol{x}_i + b)^2}{\boldsymbol{w}^T \mathbf{M}_i \boldsymbol{w}}, & y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) > 0, \\ 0, & y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \leq 0. \end{cases} \end{aligned} \tag{5.10}$$

From problem (5.10), we see that though MPE reflects the misclassification probabilities of uncertain inputs, it is indeed constructed by some geometric distances, which are related to the distributions of uncertain inputs and the optimal hyperplane. These distances are similar to the distance defined in traditional SVC except for the introduction of uncertain information and the removal of soft margins used. For this reason, we name this optimisation problem the minimax probability support vector classification (MPSVC).

The geometric interpretation is illustrated in Figure 5.5. In the figure, $d_i = \frac{p_i}{q_i}$, where $p_i = \|\boldsymbol{w}^T \boldsymbol{x}_i + b\|$ denotes the minimal distance from the mean of one uncertain input to the misclassified part of its own distribution. $q_i = \|\mathbf{M}_i^{1/2} \boldsymbol{w}\|$ is the distance between the mean of one uncertain input and the nearest edge of its distribution to the optimal solution. So d_i is proportional to both the uncertainty of the ith input and the optimal hyperplane. When the whole uncertainty ellipsoid is correctly classified by the optimal hyperplane, it is highly possible that $d_i \geq 1$ for this uncertain input, such as $\boldsymbol{z}_1$ and $\boldsymbol{z}_3$. When an uncertain input is partially correctly classified without its mean being misclassified, such as $\boldsymbol{z}_2$, $0 \leq d_2 < 1$. When an uncertainty is totally misclassified

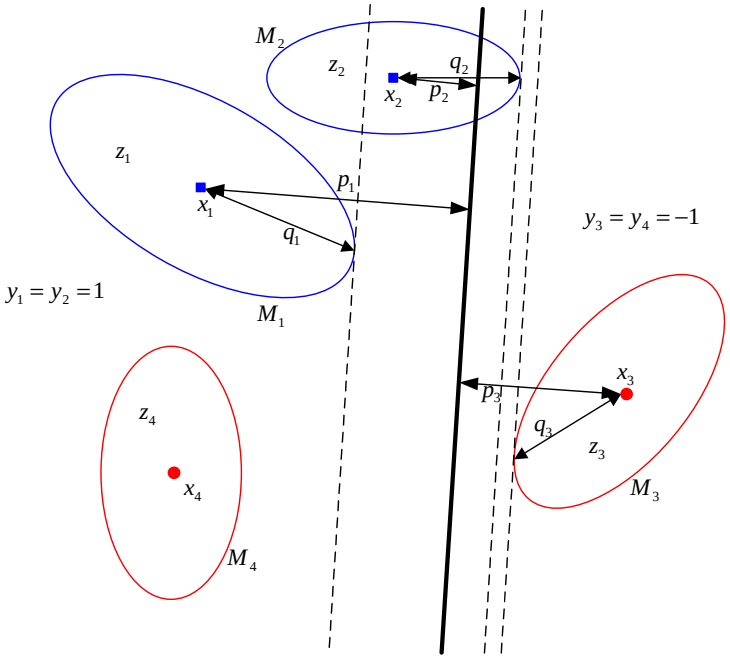


FIGURE 5.5: Geometric interpretation of the minimax probability error (MPE). In the figure, $d_i = \frac{p_i}{q_i}$, where $p_i = \|\boldsymbol{w}^T\boldsymbol{x}_i + b\|$ and $q_i = \|\mathbf{M}_i^{1/2}\boldsymbol{w}\|$. The solid line denotes the optimal classifier and the dashed lines represent the lines that are simultaneously in parallel with the optimal classifier and tangential to the nearest edges of uncertainties. For convenience, the distributions of uncertain inputs are set to Gaussian distribution here. In real classification, uncertain inputs can follow many other distributions.

or partially misclassified but with its mean misclassified, such as $\boldsymbol{z}_4$, $d_4 = 0$ and the resulting maximal misclassification probability is 1.

According to (5.10), $d_i^2 \in \left[0, \frac{(\boldsymbol{w}^T\boldsymbol{x}_i+b)^2}{\boldsymbol{w}^T\mathbf{M}_i\boldsymbol{w}}\right]$ can be confirmed, even if the ith uncertain input misclassified or not has yet been decided. The constraints of problem (5.10) can be transformed to

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq d_i\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|, \tag{5.11}$$

which are similar to the constraints of (5.1) with soft margins removed from AUSVC. Alternatively, we have $r_i \in \left[0, \frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1+\xi_i}{\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|}\right]$ from (5.1). Therefore, d_i can also be deemed as an individual probability confidence, which reflects the predicted probability of corresponding original input being classified correctly at each training point. And inequality (5.11) secures an upper bound for d_i to satisfy that the whole uncertainty ellipsoid is correctly classified at this training point. Without loss of generality, (5.10)

can be transformed as follows by replacing d_i with r_i.

$$\begin{aligned} \max_{\boldsymbol{w},b,r_i} \ & \sum_{i=1}^{l} r_i \\ s.t. \ & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq r_i\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|, \\ & \|\boldsymbol{w}\| \leq C, \\ & r_i \geq D_i \qquad i = 1, \ldots, l, \end{aligned} \tag{5.12}$$

where C is a constant, $r_i \in \mathbb{R}$ is the individual probability confidence of the ith uncertainty. The lower bound of r_i is held by $D_i \in \mathbb{R}$, which are set positive or negative in advance. For those misclassified by the future optimal hyperplane, D_i drive negative to decrease their individual probability confidence, otherwise; r_i will be bounded in a higher positive range driven by D_i. In general, D_i theoretically extends the statistical search area of the optimal solution from $r_i \geq 0$ in AUSVC to both $r_i \geq 0$ and $r_i < 0$ in problem (5.12) while this optimisation remains a convex and SOCP problem.

In fact, constraint $\|\boldsymbol{w}\| \leq C$ can be replaced by $\|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \leq L_i$, where L_i is a constant as well. We can transform (5.12) as shown below by using Tikhonov regularisation and appropriately chosen L_i.

$$\begin{aligned} \min_{\boldsymbol{w},b,r_i,q_i} \ & \frac{\|\boldsymbol{w}\|^2}{2} + \sum_{i=1}^{l_a} L_i q_i \\ s.t. \ & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq L_i r_i \quad \text{for all } 1 \leq i \leq l_a, \\ & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \leq L_i r_i \quad \text{for all } l_a + 1 \leq i \leq l, \\ & \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \leq L_i, \\ & r_i q_i \geq 1 \qquad\qquad \text{for all } 1 \leq i \leq l_a, \\ & 0 \leq r_i \leq r_{i\text{ausvc}} \qquad \text{for all } 1 \leq i \leq l_a, \\ & r_i \leq 0 \qquad\qquad \text{for all } l_a + 1 \leq i \leq l, \end{aligned} \tag{5.13}$$

where $r_{i\text{ausvc}}$ denote the final individual probability confidence obtained in the optimisation of AUSVC. We suppose that the individual probability confidence $r_{i\text{ausvc}}$ of first l_a inputs can eventually remain positive when AUSVC reaches its optimum, while other $r_{i\text{ausvc}} = 0$. From the definition of individual probability confidence, it is obviously that the predicted probabilities of correctly classifying original inputs are not limited to be cumulated in the area of their uncertainty ellipsoids. The individual probability confidence for those uncertain inputs whose uncertainty ellipsoids are wholly correctly classified should be greater than 1. Therefore, the upper bounds of these r_i are set to a large positive if $r_{i\text{ausvc}} = 1$ are finally achieved in AUSVC. Actually, MPSVC borrows $r_{i\text{ausvc}}$ from AUSVC to set additional constraints for extra degrees of freedom introduced by individual probability confidence.

Problem (5.13) is designed to maximise both the margin between different classes' uncertain inputs and their individual probability confidence which comprise MPE measure and are close related to the optimal classifier, uncertain inputs and the distance between them. L_i are introduced by Tikhonov regularisation and they can be deemed as regularisation parameters. Since the uncertain inputs close to the boundary make more contributions to MPE measure with their relatively lower individual probability confidence, we assign two values to L_i. For those close-boundary uncertain inputs, their L_i are set fixed while other L_i vary in the optimisation. And the exact values of the varied L_i can be chosen from a group of preset values through cross-validation method.

Introducing $\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{l} (\mathbf{M}_i^{1/2})^T \boldsymbol{\beta_i}$ that was derived in the dual problem of USVC and the developed kernelisation formulation back into (5.13), kernel functions can then be exploited to extend MPSVC to non-linear case. Problem (5.13) can be rewritten as:

$$
\begin{aligned}
\min_{\alpha_i, \boldsymbol{\beta}_i, b, r_i, q_i} \; & \sum_{i=1}^{l_a} L_i q_i + \frac{1}{2} \left(\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right. \\
& + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i y_i \left[\frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_j} \right]^T (\mathbf{M}_j^{1/2})^T \boldsymbol{\beta_j} \\
& + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_j y_j \boldsymbol{\beta_i}^T \mathbf{M}_i^{1/2} \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} \\
& \left. + \sum_{i=1}^{l} \sum_{j=1}^{l} \boldsymbol{\beta_i}^T \mathbf{M}_i^{1/2} \frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} (\boldsymbol{M}_j^{1/2})^T \boldsymbol{\beta_j} \right) \\
s.t. \; & y_i \left(\sum_{j=1}^{l} \alpha_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) + \sum_{j=1}^{l} \boldsymbol{\beta}_j^T \mathbf{M}_j^{1/2} \frac{\partial K(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\partial \boldsymbol{x}_j} + b \right) \geq L_i r_i \quad 1 \leq i \leq l_a, \\
& y_i \left(\sum_{j=1}^{l} \alpha_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) + \sum_{j=1}^{l} \boldsymbol{\beta}_j^T \mathbf{M}_j^{1/2} \frac{\partial K(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\partial \boldsymbol{x}_j} + b \right) \leq L_i r_i \quad l_a + 1 \leq i \leq l, \\
& \left\| \sum_{j=1}^{l} \alpha_j y_j \mathbf{M}_i^{1/2} \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i} + \sum_{j=1}^{l} \mathbf{M}_i^{1/2} \frac{\partial^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} (\mathbf{M}_j^{1/2})^T \boldsymbol{\beta}_j \right\| \leq L_i, \\
& r_i q_i \geq 1 \qquad \text{for all } 1 \leq i \leq l_a, \\
& 0 \leq r_i \leq r_{\text{iausvc}} \qquad \text{for all } 1 \leq i \leq l_a, \\
& r_i \leq 0 \qquad \text{for all } l_a + 1 \leq i \leq l, \\
& \|\boldsymbol{\beta_i}\| \leq \alpha_i \qquad \text{for all } 1 \leq i \leq l, \\
& 0 \leq \alpha_i \leq L_i \qquad \text{for all } 1 \leq i \leq l,
\end{aligned}
\tag{5.14}
$$

where $0 \leq \alpha_i \leq L_i$ is introduced according to the derivation of dual problems of SVM-based approaches, regularisation parameters L_i provide upper bounds for dual variables

α_i. Similarly, constraints $\|\boldsymbol{\beta_i}\| \leq \alpha_i$ that are generated in deriving dual problem of USVC and AUSVC, are also introduced. In fact, these two bunches of constraints are estimated by following some existing characters from SVM-based methods and SOCP problem, we do not really explore the dual problem of MPSVC. For this reason, we do not exploit $\|\boldsymbol{\beta_i}\| \leq r_i\alpha_i$ that has been used in AUSVC.

Algorithm 4 is the algorithm of MPSVC. Since MPSVC inherits the results of individual probability confidence from AUSVC, AUSVC should be included in the algorithm of MPSVC. Although the iterative algorithm of AUSVC has been illuminated in Section 5.1, for the reason of integrality, AUSVC is still shown as the first five steps in the algorithm of MPSVC without some redundant definitions, checking Algorithm 3 for more details.

Algorithm 4 MPSVC

1. Initialise $r_i = 1$, $i = 1, \ldots, l$, solve (5.2), which indeed is USVC optimisation, for the parameters α_j, $\boldsymbol{\beta}_j$, and b;
2. Substitute the obtained parameters α_j, $\boldsymbol{\beta}_j$, b and the training inputs $(\boldsymbol{z}_i, y_i)$ into (5.3) respectively to decide which inputs are correctly classified for certain by USVC, and split the training set $\mathcal{D}$ into two sets, set of certainly correctly classified inputs $\mathcal{D}_c$ and set of not certainly correctly classified inputs $\mathcal{D}_u$. The individual probability confidence r_i will never be modified again for inputs in $\mathcal{D}_c$, otherwise, a predefined positive scalar (normally 0.1) is deducted from r_i for inputs in $\mathcal{D}_u$. Repeat the following three steps (iteration 3 to 5) until $r_{inew} = r_i$, $i = 1, \ldots, l$;
3. Fix r_i, $i = 1, \ldots, l$ to the current value, solve (5.2) for the parameters α_j, $\boldsymbol{\beta}_j$, and b;
4. Substitute the obtained parameters α_j, $\boldsymbol{\beta}_j$, b and the training inputs $(\boldsymbol{z}_i, y_i)$ into (5.3) respectively to determine whether the inputs are misclassified, $\operatorname{sgn}(g(\boldsymbol{z}_i, y_i)) < 0$ or correctly classified, $\operatorname{sgn}(g(\boldsymbol{z}_i, y_i)) \geq 0$. If the inputs are misclassified and $(\boldsymbol{z}_i, y_i) \in \mathcal{D}_u$, the predefined positive scalar is deducted from its individual probability confidence r_i, otherwise, their r_i remain unchanged. All changed and unchanged individual probability confidence are saved in r_{inew};
5. If $r_{inew} = r_i$, the optimal results of α_i, $\boldsymbol{\beta}_i$, and b are achieved, otherwise, assign r_{inew} to r_i and return to step 3;
6. Preset the upper bounds of individual probability confidence r_i according to the final results of corresponding individual probability confidence $r_{i\text{ausvc}}$ obtained from AUSVC. If $r_{i\text{ausvc}} \neq 1$, then the upper bounds are set to $r_{i\text{ausvc}}$, otherwise, the upper bounds are set to a large positive (normally choosing infinity);
7. Substitute the parameters α_j, $\boldsymbol{\beta}_j$ and b obtained from optimised AUSVC to (5.3) to assign the preset two values L_{hi} and L_{lo} to regularisation parameters L_i. If $g(\boldsymbol{z}_i, y_i) - 1 \leq 0$ and $g(\boldsymbol{z}_i, y_i) + 1 \geq 0$, then corresponding $L_i = L_{\text{hi}}$, otherwise, $L_i = L_{\text{lo}}$;
8. Solve problem (5.14) for the parameters α_j, $\boldsymbol{\beta}_j$, b, r_i and q_i.

Normally, the preset value L_{hi} is fixed to a normal positive, while L_{lo} is chosen from a geometric sequence constructed between a small positive and a large positive around L_{hi} during the optimisation. After evaluating the selected measure, cross-validation method can choose proper L_{lo} for the optimisation of MPSVC.

Like AUSVC, the memory requirement of problem (5.14) is of order $O\left(l^2n^2\right)$. The cost of constructing optimisation problem, including the objective function and the constraints of (5.14), is of order $O\left(l^2n^3\right)$. The cost of computing $g(\boldsymbol{z}_i, y_i)$ for assigning L_i at step 7 is of order $O\left(l^2n^2\right)$. And the optimisation complexity is bounded with time complexity $O\left(l^{3/2}n^3\right)$ for SOCP problem solved by interior-point method. Although Algorithm 4 includes both optimisation problems of AUSVC and MPSVC, the contributions from (5.14) can be omitted from estimating the algorithmic complexity of the algorithm of MPSVC, because in most cases, such as non-separable cases, AUSVC may not be optimised in short iterations as a multi-iteration optimisation problem while problem (5.14) is a single-iteration optimisation problem. Consequently, the computational cost of AUSVC and problem (5.14) may not be commensurable. Therefore, the algorithmic complexity of MPSVC can be deemed as same as that of AUSVC.

The experiments previously exploited for different algorithms are rerun to test the performance of MPSVC. In the experiments, $L_{\text{hi}} = 1$, and the proper L_{lo} is chosen by stratified 10-fold cross-validation from a geometric sequence from 0.01 to 100 with common ratio $\sqrt{10}$. For the convenience of comparisons, the results of NMCU and TMER of MPSVC are shown in Table 5.2 and Table 5.3 along with the existing results of other algorithms from Table 4.1, Table 4.2 and Table 5.1.

Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	12.91	1.24	11.76	0.66	16.19	3.59	14.04	1.05
SVC	12.00	1.07	11.96	0.95	16.42	3.21	14.19	1.04
TSVC	11.62	0.75	11.58	0.61	14.27	1.16	13.33	0.21
AUSVC	12.32	1.27	11.81	0.84	15.67	3.10	13.80	0.82
MPSVC	12.24	1.14	11.59	0.72	16.01	3.25	13.73	0.80

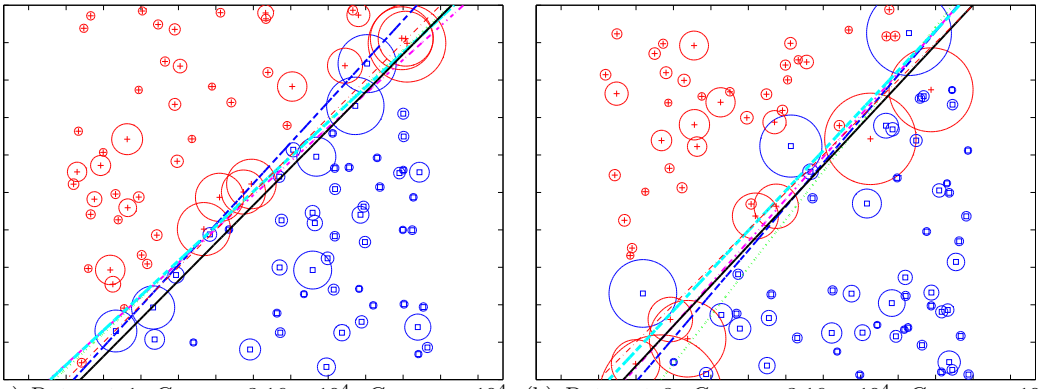
TABLE 5.2: Average test error percentages of NMCU of USVC, TSVC, SVC, AUSVC and MPSVC in reproducing Bi and Zhang's experiment, means and standard errors of NMCU are listed in the table.

Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	7.94	2.18	4.67	2.30	9.21	5.03	5.92	2.55
SVC	5.79	2.60	4.73	2.57	9.22	4.66	5.42	2.38
TSVC	4.21	2.30	3.45	2.29	5.09	2.29	2.57	1.01
AUSVC	6.26	2.81	4.56	2.37	8.24	4.56	4.94	2.25
MPSVC	6.26	2.81	3.61	2.33	8.10	4.54	4.55	1.83

TABLE 5.3: Average test error percentages of TME of USVC, TSVC, SVC, AUSVC and MPSVC in reproducing Bi and Zhang's experiment, means and standard errors of TME are listed in the table.

According to the tables, we see that MPSVC improves its performance of NMCU and TME over AUSVC, hence MPSVC surely also performs generally better than USVC and

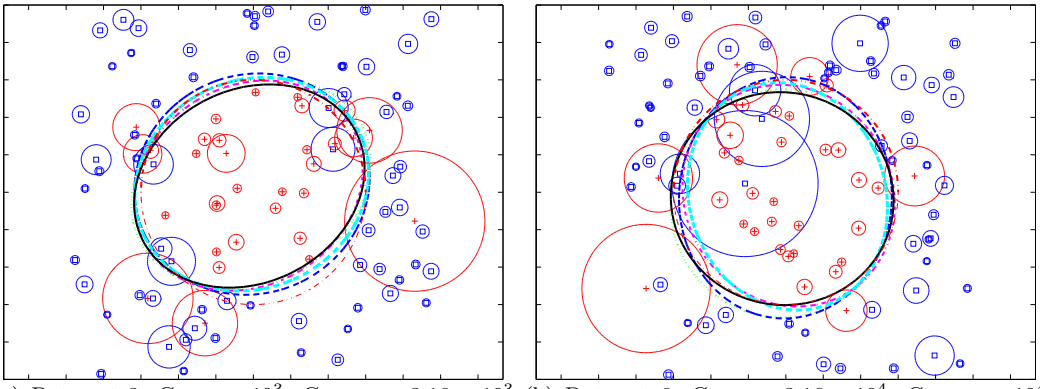
SVC except for linearly 50-input case. This reflects the advantages that MPSVC gains over other algorithms by maximising the predicted probability of corresponding original input being classified correctly at each training point. On the other hand, MPSVC still can not always perform as well as TSVC though some of its performance are close to the performance of TSVC. Some selected results of classification with linear and polynomial kernels are shown in Figure 5.6 and Figure 5.7.



(a) Data set 1, $C_{\mathrm{SVC}} = 3.16 \times 10^4$, $C_{\mathrm{USVC}} = 10^4$, $C_{\mathrm{TSVC}} = 3.16 \times 10^2$, $C_{\mathrm{AUSVC}} = 100$ and $L_{\mathrm{hi}} = 1$, $L_{\mathrm{lo}} = 1$

(b) Data set 8, $C_{\mathrm{SVC}} = 3.16 \times 10^4$, $C_{\mathrm{USVC}} = 10$, $C_{\mathrm{TSVC}} = 10$, $C_{\mathrm{AUSVC}} = 10$ and $L_{\mathrm{hi}} = 1$, $L_{\mathrm{lo}} = 0.01$

FIGURE 5.6: Selected results from the 1st and 8th 100-input training data sets for linear classification in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line, SVC is represented by green dotted line, AUSVC is represented by magenta dash-dot line and thick cyan dashed line represents MPSVC. The target function is illustrated by red dash-dot line.



(a) Data set 3, $C_{\mathrm{SVC}} = 10^3$, $C_{\mathrm{USVC}} = 3.16 \times 10^3$ and $C_{\mathrm{TSVC}} = 100$, $C_{\mathrm{AUSVC}} = 100$ and $L_{\mathrm{hi}} = 1$, $L_{\mathrm{lo}} = 0.32$

(b) Data set 9, $C_{\mathrm{SVC}} = 3.16 \times 10^4$, $C_{\mathrm{USVC}} = 10^4$ and $C_{\mathrm{TSVC}} = 3.16 \times 10^4$, $C_{\mathrm{AUSVC}} = 3.16 \times 10^4$ and $L_{\mathrm{hi}} = 1$, $L_{\mathrm{lo}} = 1$

FIGURE 5.7: Selected results from the 3rd and 9th 100-input training data sets for classification with quadratic kernel in the reproduction of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line, SVC is represented by green dotted line, AUSVC is represented by magenta dash-dot line and thick cyan dashed line represents MPSVC. The target function is illustrated by red dash-dot line.

MPSVC inherits the characters of lowering the predicted probabilities from AUSVC, which also benefits from these characters to achieve better performance than USVC and SVC. Besides, MPSVC can decrease some deviations from the target function in AUSVC by maximising the predicted probabilities introduced from MPM. For instance, MPSVC decreases AUSVC's deviations from the target function generated in the area where the uncertainties from one class dominate the uncertainties from the other class, which is shown in the top part of Figure 5.6(a) and the bottom part of Figure 5.6(b). And Figure 5.7(a) shows the case in which there exists some sparsity of training points from both classes around the original boundary. MPSVC can better recover the target function than USVC, SVC and AUSVC with its closer optimal solution to the original boundary.

Though MPSVC has taken advantages over USVC, SVC and AUSVC, it is still shadowed by TSVC which gives the best performance of all algorithms in the comparison over the data sets generated by following the rules in Bi and Zhang (2005). However, as what we have discussed before, it is arguable that TSVC will also outperform other algorithms in different settings that are used to create data sets. Furthermore, MPSVC is expected to provide generally better performance in different settings by minimising MPE for each data set. Therefore, the experiment used in Section 4.1.4 is rerun here with data sets comprising original inputs being contaminated to move along traces in parallel with the original boundary. The results of means and standard errors of NMCU and TME over five uncorrupted and corrupted test sets are reported in Table 5.4 and Table 5.5.

Measure / Algorithm	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	5.70	2.69	3.82	2.18	6.68	1.87	4.58	1.89
SVC	2.88	2.10	1.52	1.18	4.51	1.33	3.07	1.07
TSVC	3.14	2.44	2.40	1.94	4.80	1.70	3.40	1.47
AUSVC	3.05	1.77	2.40	2.11	5.10	2.05	4.02	1.30
MPSVC	3.01	1.85	1.86	1.22	4.33	1.23	3.09	1.22

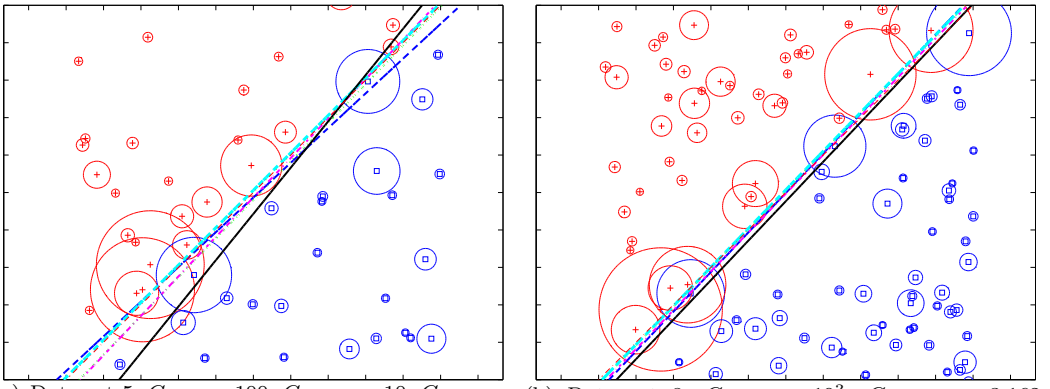
TABLE 5.4: Average test error percentages of NMCU of USVC, TSVC, SVC, AUSVC and MPSVC in recomposing Bi and Zhang's experiment, means and standard errors of NMCU are listed in the table.

Similar to the results listed in Table 4.3 and Table 4.4, SVC is still the best performer of all algorithms in this test. MPSVC, which benefits from its predecessor AUSVC, can achieve better performance over TSVC in comparison. Some selected classification results with linear and quadratic kernels are illustrated in Figure 5.8 and Figure 5.9.

In the figures, we see that the solution of MPSVC is close to that of TSVC when the optimal classifiers of TSVC and USVC are significantly different in the same training set. The reason is that individual probability confidence of some inputs can be driven to negative in MPSVC by composing the optimisation problem with MPE. And this is

Algorithm \ Measure	Linear Target Function				Quadratic Target Function			
	$l = 50$		$l = 100$		$l = 50$		$l = 100$	
	mean	std	mean	std	mean	std	mean	std
USVC	5.65	2.68	3.78	2.16	6.63	1.88	4.29	1.86
SVC	2.86	2.06	1.52	1.18	4.50	1.37	2.99	1.21
TSVC	3.13	2.44	2.36	1.88	4.85	1.74	3.44	1.70
AUSVC	3.02	1.74	2.36	2.11	4.98	2.00	3.72	1.37
MPSVC	3.01	2.38	1.84	1.21	4.30	1.26	3.40	0.85

TABLE 5.5: Average test error percentages of TME of USVC, TSVC, SVC, AUSVC and MPSVC in recomposing Bi and Zhang's experiment, means and standard errors of TME are listed in the table.

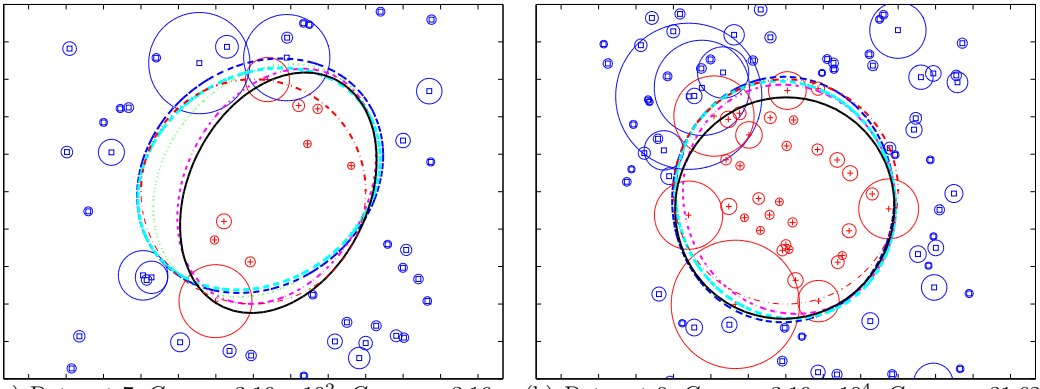


(a) Data set 5, $C_{\text{SVC}} = 100$, $C_{\text{USVC}} = 10$, $C_{\text{TSVC}} = 10^3$, $C_{\text{AUSVC}} = 100$ and $L_{\text{hi}} = 1$, $L_{\text{lo}} = 0.01$

(b) Data set 8, $C_{\text{SVC}} = 10^3$, $C_{\text{USVC}} = 3.162$, $C_{\text{TSVC}} = 10$, $C_{\text{AUSVC}} = 10$ and $L_{\text{hi}} = 1$, $L_{\text{lo}} = 1$

FIGURE 5.8: Selected results from the 5th 50-input and 8th 100-input training data sets for linear classification in the recomposition of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line, SVC is represented by green dotted line, AUSVC is represented by magenta dash-dot line and thick cyan dashed line represents MPSVC. The target function is illustrated by red dash-dot line.

similar to TSVC, in which individual probability confidence are fixed to -1 to provide lower predicted probabilities of original inputs being correctly classified. Selecting lower predicted probabilities can help algorithms better recover the target function when the contamination tends to move inputs across the boundary from one class to the other class. However, the current setting of creating data sets in this experiment does not make the means of the distributions of uncertain inputs cross the boundary, so that TSVC can not take advantages by using lower predicted probabilities, or equivalently choosing the farthest points in uncertainties. On the contrary, the predicted probabilities used in MPSVC are indeed set to achieve moderate values like those fixed to 0 in SVC, neither as aggressive as those in TSVC, nor as conservative as those in USVC.



(a) Data set 7, $C_{\mathrm{SVC}} = 3.16 \times 10^2$, $C_{\mathrm{USVC}} = 3.16 \times 10^2$, $C_{\mathrm{TSVC}} = 3.16 \times 10^2$, $C_{\mathrm{AUSVC}} = 100$ and $L_{\mathrm{hi}} = 1$, $L_{\mathrm{lo}} = 0.32$

(b) Data set 9, $C_{\mathrm{SVC}} = 3.16 \times 10^4$, $C_{\mathrm{USVC}} = 31.62$, $C_{\mathrm{TSVC}} = 10^5$, $C_{\mathrm{AUSVC}} = 10^3$ and $L_{\mathrm{hi}} = 1$, $L_{\mathrm{lo}} = 1$

FIGURE 5.9: Selected results from the 7th 50-input and 9th 100-input training data sets for classification with quadratic kernel in the recomposition of Bi and Zhang's experiment (Bi and Zhang, 2005). TSVC is represented by blue dashed line, USVC is represented by black solid line, SVC is represented by green dotted line, AUSVC is represented by magenta dash-dot line and thick cyan dashed line represents MPSVC. The target function is illustrated by red dash-dot line.

5.3 Algorithmic Complexity Analysis

Previously, the algorithmic complexities of newly developed approaches have been derived. However, model selection is required for choosing a proper regularisation parameter from a regularisation parameter repository whose size s is not large for selecting sole regularisation parameter in the experiments. The introduction of resampling method, such as k-fold cross-validation, also magnifies the cost of training. Without loss of generality, the computational cost, which comprise the cost of constructing arguments, the optimisation complexity and the cost of constructing auxiliary parameters, can be approximately magnified ks times. Besides, evaluating the measure for choosing proper regularisation parameters is required by the cross-validation, and the cost of computing the measure is introduced in the computational cost, and yet it is not commensurable to the cost of constructing auxiliary parameters. The results are summarised in Table 5.6 with SeDuMi exploited as the optimiser in the experiments.

	storage complexity	argument complexity	optimisation complexity	complexity of auxiliary parameters	complexity of computing measure
SVC	$O\left(l^2\right)$	$O\left(ksl^2n\right)$	$O\left(ksl^{3/2}\right)$	N/A	$O\left(sl^2n\right)$
TSVC	$O\left(l^2\right)$	$O\left(ksL_tl^2n\right)$	$O\left(ksL_tl^{3/2}\right)$	$O\left(ksL_tl^2n^2\right)$	$O\left(sL_tl^2n\right)$
USVC	$O\left(l^2n^2\right)$	$O\left(ksl^2n^3\right)$	$O\left(ksl^{3/2}n^3\right)$	N/A	$O\left(sl^2n^2\right)$
AUSVC	$O\left(l^2n^2\right)$	$O\left(ksL_al^2n^3\right)$	$O\left(ksL_al^{3/2}n^3\right)$	$O\left(ksL_al^2n^2\right)$	$O\left(sL_al^2n^2\right)$
MPSVC	$O\left(l^2n^2\right)$	$O\left(ksL_al^2n^3\right)$	$O\left(ksL_al^{3/2}n^3\right)$	$O\left(ksL_al^2n^2\right)$	$O\left(sL_al^2n^2\right)$

TABLE 5.6: Comparison of the algorithmic complexities of SVC, TSVC, USVC, AUSVC and MPSVC.

Here, l is the number of inputs, n is the number of features, the numbers of iterations of TSVC and AUSVC are denoted respectively by L_t and L_a, which solely depend on the precision setting of optimiser used. In the experiments, we select 10-fold cross-validation and small regularisation parameter repository. $k = 10$ and s are normally commensurable to the features of inputs, and can not be omitted when estimating the cost. Since single-iteration approaches do not involve any additional cost associated with auxiliary parameters, the complexity of auxiliary parameters is displayed as "N/A" for SVC and USVC in Table 5.6.

USVC, AUSVC and MPSVC require more memory than other approaches during optimisation because of the introduction of uncertainties $\mathbf{M}_i$ and dual variables $\boldsymbol{\beta}_i$. The larger n becomes, the more memory these methods need. As iterative algorithms, the computational cost of TSVC, AUSVC and MPSVC is firmly related to the number of iterations used in the optimisation. The larger L_t and L_a are, the more time-consuming these algorithms become. Benefiting from its simple structure and uncertainty-free optimisation problem, SVC needs the least time in constructing the optimisation problem and its optimisation complexity is of the least order of all. While MPSVC costs the most time. The argument construction complexity and the optimisation complexity have similar effect on the computational cost and are related by $\sqrt{l}$ in USVC, AUSVC and MPSVC. On the other hand, the argument construction complexity dominates the optimisation complexity by rate $\sqrt{l}n$ in SVC and TSVC. In fact, the cost of computing the measure for k-fold cross-validation can be omitted from estimating the computational cost. Practically, the computational complexities of SVC, USVC, AUSVC and MPSVC are mainly decided by argument complexity and optimisation complexity. On the contrary, the computational complexity of TSVC is mainly decided by auxiliary parameter complexity because uncertainties $\mathbf{M}_i$ are introduced to construct these auxiliary parameters. Moreover, the number of features n significantly affect the time consumed by SeDuMi. From the table, we see that the optimisation complexity of USVC may be n^3 times more than that of SVC.

5.4 Summary

In this chapter, two novel algorithm have been developed for classification subject to input uncertainty. Acting as a natural extension of USVC, AUSVC follows the same path of USVC to construct its optimisation formulation with the iterative constraints. The individual probability confidence of one uncertain inputs may vary in each step with respect to the classification of the same uncertain input from the last step. This variation directly leads to the changes of the predicted probabilities that the unknown original inputs are going to be correctly classified by the optimal solution. Kernelised dual problem of AUSVC can be obtained through the same way used in USVC.

In classification subject to input uncertainty, MPM can not only extend the SVM-based methods from the uncertain inputs' Gaussian distribution assumption to more general case, even the case that the prior knowledge of the distributions of uncertain inputs is not available, but also introduce a new measure, MPE, to evaluate the performance of different algorithms. Based on MPE, MPSVC is generated for classification subject to input uncertainty through a more direct way, maximising the predicted probability of corresponding original input being classified correctly at each training point. Some constraints of MPSVC are composed by borrowing existing results of AUSVC. And MPSVC can also be extended to non-linear case by using the kernelisation formulation. The iterative algorithm of MPSVC introduces both the strategies from MPSVC and AUSVC to allow the individual probability confidence of uncertain inputs to achieve values as high as possible. As the result, MPSVC incorporates the contributions from both uncertain inputs, being misclassified and being correctly classified, into the optimisation and achieves a generally better performance.

In the experiments, two kinds of settings are applied to create contaminated training and test sets, one setting follows Bi and Zhang (2005), the other introduces a lighter contamination in data sets. In general, TSVC and SVC perform the best of all algorithms respectively under two settings. AUSVC can reach improved performance than USVC. The overall performance of MPSVC ranks the second of all under both settings. On the other hand, MPSVC has the greatest algorithmic complexity of all algorithms. However, if we simply focus on selecting an algorithm that generally has good performance in both classifying the uncertain inputs and recovering the target function when how the data sets are contaminated is unknown, MPSVC may be the right choice. In the next chapter, these algorithms are going to be tested over more practical data sets.

Chapter 6

Data Analysis

In this chapter, the different algorithms proposed and developed in the thesis are compared with each other. Two aspects of the results are focused in the data analysis,

- **Classification**. The measures of the number of misclassified centres of the uncertainties (NMCU), the minimax probability error (MPE) and some other related measures can be used to evaluate the performance of algorithms in classifying the contaminated inputs. It compares the different methods simply through the number of the contaminated inputs being correctly discriminated in the test no matter how this obtained optimal classifier has been changed from the true target function;
- **Restoration**. One of the most important goals in this thesis is to recover the true target function when the training data sets are contaminated. The performance of the algorithms in restoration is determined by the measure of the test misclassification error (TME).

6.1 Statistical Comparison

In the last two chapters, several algorithms are compared by mainly focusing on their classification figures, besides, we also compute some average classification measures of the algorithms across the data sets. Although every measure of each classifier is commensurable to be averaged over all data sets, an overwhelming performance of the classifier on one data set can compensate this classifier for its overall bad performance, or a total failure on one data set may affect the fair results on others.

Statistical comparison is appropriate to measure differences between the classifiers from different aspects. In statistical comparison, the null hypothesis is going to be rejected when the decision p-value is less than a given statistical significance level α, which is

defined as probability of making a decision to reject the null hypothesis when the null hypothesis is actually true. This is also known as a Type I error. In this chapter, the null hypothesis is that all classifiers derived from different algorithms have no differences. Generally, p-value can be obtained by computing the results of classifiers. No matter what formula is used to get the p-value and what distribution the difference between classifiers follows, p-value can be considered to represent the difference between classifiers. For instance, we assume that the difference between classifiers follows a Gaussian distribution, and we want the null hypothesis can be rejected with a certain confidence $(1-\alpha)$ when this difference value exceed a certain value (one-sided risk, see Figure 6.1). Where q_α in Figure 6.1 is called the critical value. When the difference between classi-

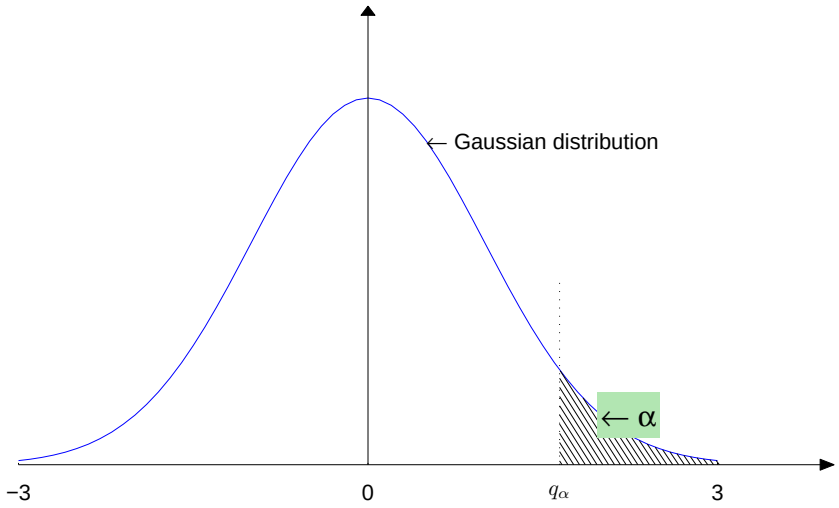


FIGURE 6.1: The difference between classifiers follows a Gaussian distribution.

fiers is large enough, the resulting p-value will be in the shade zone shown in Figure 6.1, thus the null hypothesis, that all classifiers derived from different algorithms have no differences, can be rejected with a certain confidence $(1-\alpha)$. Traditionally, the selected α is 0.05 or 0.1. If the performance of classifiers is similar, the p-value will be in the white zone shown in Figure 6.1, and the null hypothesis can not be rejected. In application, other distributions are widely introduced in analysing the performance of different algorithms, such as studentised range distribution in the Nemenyi test, the Bonferroni-Dunn test, the student's t-test and the Tukey test, the chi-square distribution and the F-distribution in the Friedman test, and the F-distribution in ANOVA. The distributions used in different tests diversify their critical values when the statistical significance level α remains unchanged. The lower q_α becomes, the more the differences can be detected. Thus the power of the statistical tests is much greater with smaller q_α.

The usual way of multiple hypothesis testing is to control the family-wise error, which is the probability of generating at least one Type I error in any of the comparisons between the algorithms. But the statistical significance of the differences between multiple

means can be directly analysed through specialised statistics procedures. Two well-known methods, analysis of variance (ANOVA) and its non-parametric counterpart, the Friedman test have been reviewed by Demšar (2006).

6.1.1 Analysis of Variance

Analysis of variance (ANOVA) proposed by Fisher (1970) is a common statistical method. ANOVA tests the differences between the performance of more than two classifiers derived from different algorithms over the same series of data sets.

The fundamental technique is a partitioning of the total variability into components related to the effects used in the model, including variability between the classifiers and the residual (error) variability. We have,

$$SS_t = SS_c + SS_e, \tag{6.1}$$

where SS is the abbreviation of sum of squares and represents the variance, SS_t is the total variance, SS_c is the variance between the classifiers, and SS_e represents the error variance. According to the definition of ANOVA, we have

$$\begin{aligned} F &= \frac{SS_c/df_c}{SS_e/df_e}, \\ \text{where} \quad df_t &= df_c + df_e, \end{aligned} \tag{6.2}$$

and df is the abbreviation of the number of degrees of freedom, which can be partitioned and denoted in a similar way. Following the definitions of chi-square distribution and F-distribution, we know that (6.2) is a F test statistic. If the p-value derived from this F test statistic and F-distribution is smaller than a given significance level α, the between-classifiers variability is significantly larger than the error variability. Therefore, the null hypothesis can be rejected with a certain confidence $(1 - \alpha)$, and we can conclude that there are some differences between the classifiers. If the differences exist, a post-hoc test, such as Tukey test (Tukey, 1949) and Dunnett test (Dunnett, 1955) can be proceeded to find out which classifiers actually differ.

However, ANOVA needs to fulfill some assumptions before it can be used to test the differences. The common assumption of ANOVA is that the errors are independently, identically and normally distributed. To satisfy this common assumption, the following assumptions are needed. First, ANOVA assumes that the distributions of every measure across the data sets are normal for each of the classifiers. Shapiro-Wilk test (Shapiro and Wilk, 1965) can be used to confirm normality. Second, ANOVA needs the equality of variances, called homoscedasticity, which means the variance of every measure should be the same for each of the classifiers. Levene's test (Levene, 1960), a statistic used to assess the equality of variance in different samples, is typically used to confirm the homogeneity

of variances. In the thesis, the data sets will be contaminated by different settings of noise, even the comparison is between classifiers across the data sets contaminated by the same setting of noise, there is no guarantee for the two assumptions. Especially, the homoscedasticity in data sets for each of the classifiers can not be taken for granted due to the different characters of the learning algorithms. ANOVA therefore is not suitable in this thesis for testing the differences between different algorithms.

6.1.2 Friedman Test

The Friedman test is a non-parametric statistical test developed by Friedman (1937). Similar to ANOVA, it is used to detect differences between classifiers by ranking all classifiers together in each data set and considering the values of ranks across the data sets.

We assume that there are totally N data sets and k algorithms in the comparison, and one of the measures is compared. The rank is recorded as a tableau $\{r_{ij}\}_{N\times k}$, if there are tied values, assign to each tied value the average of the ranks that would have been assigned without ties. r_{ij} denotes the rank for this measure of the jth of all k algorithms in the ith data set. The Friedman test statistic is given by

$$\chi_F^2 = \frac{SS_c}{SS_e}.$$

where

$$\bar{r}_{\cdot j} = \frac{1}{N}\sum_{i=1}^{N} r_{ij}, \qquad \bar{r} = \frac{1}{Nk}\sum_{i=1}^{N}\sum_{j=1}^{k} r_{ij},$$
$$SS_c = N\sum_{j=1}^{k}(\bar{r}_{\cdot j} - \bar{r})^2, \qquad SS_e = \frac{1}{N(k-1)}\sum_{i=1}^{N}\sum_{j=1}^{k}(r_{ij} - \bar{r})^2. \tag{6.3}$$

When N and k are large (i.e. $N > 15$ and $k > 4$), the Friedman test statistic can be approximated by the chi-square distribution with $k-1$ degrees of freedom. If N or k is small, the approximation to chi-square becomes poor and the χ_F^2 in (6.3) should be obtained from tables specially prepared for the Friedman test (Zar, 1998; Sheskin, 2000). Since $\sum_{j=1}^{k} r_{ij}^2$ can be approximated to $1^2 + 2^2 + \ldots + k^2$, we have

$$SS_e = \frac{1}{N(k-1)}\left(\sum_{i=1}^{N}\frac{k(k+1)(2k+1)}{6} - \frac{NK(k+1)^2}{4}\right) = \frac{k(k+1)}{12}.$$

Thus (6.3) can be simplified as follows,

$$\chi_F^2 = \frac{12N}{k(k+1)}\left(\sum_{j=1}^{k}\bar{r}_{\cdot j}^2 - \frac{k(k+1)^2}{4}\right). \tag{6.4}$$

And we want to know the minimal size N of data sets, that guarantees the average ranks of different classifiers across the data sets can be discriminated (or the null hypothesis can be rejected) with a certain confidence $1-\alpha$ obtained from χ_F^2. The second minimum of $\sum_{j=1}^{k} \bar{r}_{\cdot j}$ is required, because

$$\min_{\bar{r}_{\cdot j}=\frac{k+1}{2}} \sum_{j=1}^{k} \bar{r}_{\cdot j}^2 = \frac{k(k+1)^2}{4}.$$

Assume the only difference between all ranks is ϵ, the ranks can be represented by,

$$\frac{k+1-\epsilon}{2}, \overbrace{\frac{k+1}{2}, \ldots, \frac{k+1}{2}}^{k-2}, \frac{k+1+\epsilon}{2},$$

so the second minimum of $\sum_{j=1}^{k} \bar{r}_{\cdot j}$ is obtained and we have,

$$\sum_{j=1}^{k} \bar{r}_{\cdot j}^2 - \frac{k(k+1)^2}{4} = \frac{\epsilon^2}{2},$$

and

$$N = \frac{k(k+1)}{6\epsilon^2}\chi_F^2. \tag{6.5}$$

When ϵ becomes smaller or more classifiers need to be compared, more data sets are required in the experiment.

Based on the Friedman test statistic χ_F^2, a better statistic was proposed by Iman and J.M.Davenport (1980),

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{6.6}$$

which is distributed according to the F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

Since the advantage of no assumption of normality and homoscedasticity required in data sets, the Friedman test is chosen as the method in statistical comparison for multiple algorithms. If the null hypothesis is rejected, we can proceed with a post-hoc test.

6.1.3 Post-Hoc Analysis

Post-hoc analysis is used at the second stage of ANOVA or the Friedman test if the null hypothesis is rejected. The question of interest at this stage of the Friedman test is which classifiers significantly differ from others in respect to the ranks. Each time the ranks of a group of classifiers are considered, a statistical test is effectively performed. Four post-hoc tests are chosen in this thesis according to the reviews by Demšar (2006).

The Nemenyi test (Nemenyi, 1963) and the Bonferroni-Dunn test (Dunn, 1961) use same formula to compute their test statistics z.

$$\begin{aligned} z &= \|\bar{r}_{.i} - \bar{r}_{.j}\| \Big/ \sqrt{\frac{k(k+1)}{6N}}\ , \\ CD &= q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \end{aligned} \tag{6.7}$$

where CD is called the critical difference, and critical values q_α are based on the studentised range statistic divided by $\sqrt{2}$. The performance of the ith and jth algorithms is significantly different if the corresponding average ranks $\|\bar{r}_{.i} - \bar{r}_{.j}\| \geq CD$ (or test statistic $z \geq q_\alpha$). The difference between the Nemenyi test and the Bonferroni-Dunn test is that the power of the Bonferroni-Dunn test is greater than than the Nemanyi test by using smaller critical values derived from $\frac{\alpha}{k-1}$ when the significance level is set to α in both tests. Combining (6.5) and (6.7), we have

$$CD = \frac{q_\alpha}{\chi_F^2}\epsilon. \tag{6.8}$$

In the Nemenyi test and the Bonferroni-Dunn test, the critical difference is proportional to the possibly-detected minimal rank difference in the Friedman test. Generally, $q_\alpha < \chi_F^2$, so these two post-hoc tests can further find more difference between the performance of algorithms than the Friedman test.

Besides, the pairwise comparisons or the comparisons between all algorithms and a control algorithm are used as default ways for the Nemenyi test and the Bonferroni-Dunn test (Demšar, 2006). Indeed, although the algorithms with the best or worst performance sometimes can be used as the control algorithm, no algorithm is appointed the control algorithm in this thesis. Thus the pairwise comparisons are mainly considered here. To collect more information from these two tests in post-hoc analysis, some extra tricks are introduced in the comparisons. In the Nemenyi test, every two algorithms are compared, if $\bar{r}_{.i} - \bar{r}_{.j} \geq CD$, then the ith algorithm is voted as a bad-performance algorithm and the jth algorithm is voted as a good-performance algorithm, and if $\bar{r}_{.j} - \bar{r}_{.i} \geq CD$, we have the opposite voting result. After finishing all comparisons, one algorithm can generally be recognised as an algorithm with the bad or good performance if the least number of other algorithms that vote this algorithm as a bad-performance or good-performance algorithm is satisfied. The least number can be set to more than half of all other algorithms for example. In this thesis, this least number is set to $k - 2$.

In the Bonferroni-Dunn test, a definition of close-performance algorithms is introduced besides the definitions of good-performance and bad-performance algorithms. In fact, Close-performance algorithms are defined by being separated from those algorithms that

do not belong to good-performance algorithms in pairwise comparisons,

$$\begin{aligned} CD - \epsilon \leq \bar{r}_{.i} - \bar{r}_{.k} \leq CD, \\ \bar{r}_{.j} - \bar{r}_{.k} \leq CD - \epsilon, \end{aligned} \tag{6.9}$$

where ϵ is a small scalar. The ith algorithm has close performance to the algorithm that performs significantly better than the kth algorithm, the jth algorithm does not perform significantly better than the kth algorithm, and the ith algorithm is deemed as a close-performance algorithm and the jth algorithm is deemed as a bad-performance algorithm in the Bonferroni-Dunn test. The detailed procedure is shown as follows,

1. Finding the worst-performance algorithm with the largest average rank of all algorithms, all others are compared with this worst-performance algorithm to determine the good-performance, bad-performance and close-performance algorithms respectively;
2. Recomparing the good-performance algorithms with the algorithms whose performance are no better than or almost better than the worst performance to further discriminate the good-performance algorithms from bad-performance and close-performance algorithms in these existing good-performance algorithms;
3. Updating the good-performance, bad-performance and close-performance algorithms, return to step 2 until no more bad-performance and close-performance algorithm is left or in total only one algorithm is left;

In statistics, the Holm test (Holm, 1979) and the Hommel test (Hommel, 1988) perform more than one hypothesis test simultaneously. The Holm test uses step-down Bonferroni-Dunn procedures instead of the single-step procedure to sequentially test the hypotheses ordered by their significance. We assume the ordered p-values by p_1,p_2,...,p_{k-1}, and $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. The Holm test compare each p_i with $\frac{\alpha}{k-i}$, starting with the most significant p_1. If p_1 is lower than $\frac{\alpha}{k-1}$, the corresponding hypothesis is rejected and p_2 is allowed to compare with $\frac{\alpha}{k-1}$ in the test. If the second hypothesis is rejected, the test proceeds with the third hypothesis, and so on. As soon as a certain null hypothesis can not be rejected, all the remaining hypotheses are retained. In the Hommel test, the decisions for the individual hypotheses can be performed in the following way: find $j = \max\{i \in \{1,\ldots,n_h\} : p_{n_h-i+k} > k\alpha/i \quad \forall k = 1,\ldots,i\}$. If no such j exists, reject all hypotheses, otherwise reject all for which $p_i \leq \alpha/j$. Here, n_h is the number of the hypotheses in the Hommel test, if the classifier that has the best or the worst performance of all k classifiers is chosen as a control classifier and is compared with all other classifiers, $n_h = k - 1$. Some comparisons of post-hoc tests have been given by Demšar (2006). The multi-step tests are more powerful than the single-step the Nemenyi test and the Bonferroni-Dunn test. The Hommel test is slightly more powerful than the Holm test by gaining more complexity of its calculation.

6.2 Experimental Setup

In the experiment, we use Gunnar Rätsch's data sets (Rätsch, 2001b), which are generated based on the UCI, DELVE and STATLOG benchmark repositories, such as breast cancer, heart, titanic, etc. All problems have been configured by Rätsch (2001b) to adapt binary classification problems and 100 random realisations of training and test set are generated for each problem. The experimental data sets can be downloaded from Rätsch (2001a).

6.2.1 Contamination

To make the contamination totally random, we assume how the attributes of an original input $\boldsymbol{x}_{io}$ are affected by the contamination is unknown before a random scalar value ι is drawn from a uniform distribution on the unit interval. Meanwhile, a series $[0, \frac{1}{n+1}, \frac{2}{n+1}, \ldots, 1]$ is obtained and $bd_j = [\frac{j-1}{n+1}, \frac{j}{n+1}), j = 1, \ldots, n+1$ is the jth of all $n+1$ intervals. Here n denotes the dimensions of the input space. If $\iota \in bd_j$, $j-1$ out of n attributes of $\boldsymbol{x}_{io}$ are contaminated by Gaussian noise.

We adopt the rule of generating Gaussian noise from Chapter 4. These Gaussian noise have the same proportional sizes to that of the input space. In the thesis, the standard sizes of axes of ellipses are randomly chosen from $[0.01, 0.08]$ for small noise of two dimensions in the unit space, and the counterpart axes for large noise of two dimensions are chosen from $[0.05, 0.2]$. Then two factors related to the sizes of axes need to be considered for noise of higher dimensions in a non-unit space, the space factor $\boldsymbol{f}_s$ and the dimension factor f_d. The sizes of axes is affected by the space factor, which is computed through the sizes of the related dimensions of the input space divided by that of the unit space, and $\boldsymbol{f}_s = [f_{s1}, \ldots, f_{sk}, \ldots, f_{sn}]^T$, where f_{sk} represents the space factor for the kth dimension. To avoid overestimating the size of the input space from some unwanted interference, the maximal limit of each dimension is estimated by counting the mean of the top 1% data from the second maximum of this dimension. The minimal limit is the mean of the bottom 1% data from the second minimum of this dimension. The dimension factor f_d is determined by f_{2d} and f_{nd}, where f_{2d} are size factors that reflect the proportional sizes of Gaussian noise in two-dimensional unit space and f_{nd} are magnification factors that applying the same proportional sizes of two-dimensional Gaussian noise to a higher-dimensional input space. The computation of both f_{2d} and f_{nd} are based on the upper and lower limits of axes of ellipses respectively.

$$
\begin{aligned}
f_{2dsl} &= \sqrt[n]{1/(0.01 \times 2)^2}, & f_{ndsl} &= \sqrt[n]{1/(0.01 \times 2)^{n-2}},\\
f_{2dsu} &= \sqrt[n]{1/(0.08 \times 2)^2}, & f_{ndsu} &= \sqrt[n]{1/(0.08 \times 2)^{n-2}},\\
f_{2dll} &= \sqrt[n]{1/(0.05 \times 2)^2}, & f_{ndll} &= \sqrt[n]{1/(0.05 \times 2)^{n-2}},\\
f_{2dlu} &= \sqrt[n]{1/(0.2 \times 2)^2}, & f_{ndlu} &= \sqrt[n]{1/(0.2 \times 2)^{n-2}},
\end{aligned}
$$

where f_{2dsl}, f_{2dsu}, f_{2dll}, f_{2dlu} and f_{ndsl}, f_{ndsu}, f_{ndll}, f_{ndlu} represent the size factors and the magnification factors for lower and upper limits of axes of small and large noise respectively. Therefore, the size of kth axis of ellipsoids is randomly selected from $[0.01 f_{sk} f_{2dsl} f_{ndsl}, 0.08 f_{sk} f_{2dsu} f_{ndsu}]$ for small noise and $[0.05 f_{sk} f_{2dll} f_{ndll}, 0.2 f_{sk} f_{2dlu} f_{ndlu}]$ for large noise in higher-dimensional input space.

When generating ellipses or ellipsoids, the chosen axes for specified contaminated attributes are first randomly introduced to the diagonal elements of covariance matrix of Gaussian noise. If not all attributes of $\boldsymbol{x}_{io}$ are contaminated, other diagonal elements are assigned small amount of zero order regularisation $\epsilon = 10^{-12}$ to avoid singularity in matrix. Besides, several contaminated attributes may be covariant, so the rotation matrix is introduced to the obtained covariance matrix to make the experiments more comprehensive. In two-dimensional space, we have

$$\begin{aligned} \mathbf{M}_i &= \mathbf{R}\mathbf{S}_i\mathbf{R}^T \\ \mathbf{R} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} &\qquad \mathbf{S}_i = \begin{bmatrix} s_{11} & 0 \\ 0 & s_{22} \end{bmatrix}, \end{aligned} \tag{6.10}$$

where $\mathbf{R}$ is two-dimensional rotation matrix and $\mathbf{S}_i$ is diagonal matrix with assigned diagonal elements. We can generate three-dimensional rotation matrices based on the one of two dimensions, three-dimensional rotation matrices can be used to formulate four-dimensional rotation matrices, and so on. Using above method, we surely can get n-dimensional rotation matrices, but we prefer a rather simple way (see Algorithm 5) by the property, the covariance matrix $\mathbf{M}_i$ must be a positive semi-definite symmetric matrix. where r is a random number drawn from a uniform distribution on the unit

Algorithm 5 Generating Covariance Matrix

```
repeat
  for i = 1 to n do
    for j = i to n do
      if s_ii > ε and s_jj > ε then
        s_ij = s_ji = r min(s_ii, s_jj)
      end if
    end for
  end for
until Cholesky decomposition of the updated S_i is valid.
```

interval. Since the rotation matrix may differ the values of diagonal elements in $\mathbf{M}_i$ from those in $\mathbf{S}_i$. This updated $\mathbf{S}_i$ is not the same as $\mathbf{M}_i$ in (6.10), it is only an alternative for generating the covariance matrix. Contamination can be applied to the introduced data sets according to three different settings.

6.2.2 Bi and Zhang's Setting

The setting in the experiment of Bi and Zhang (2005) shows that the inputs being contaminated are set to move towards the other class by trying to cross the original boundary. Three experimental parameters, τ, ν and ζ, are introduced to strengthen the contamination procedure under Bi and Zhang's setting. We assume there are in total l inputs in each training data set (for instance, $l = 200$ for breast cancer problem). All the inputs are ordered in an ascending order of their distances to the target boundary. The first $(1 - \tau)l$ of all sorted inputs are selected to be contaminated by large Gaussian noise, while others are contaminated by small Gaussian noise.

Parameter ν illustrates part of all inputs, that are severely affected by large or small Gaussian noise. And the effects from severe or light contamination make the original input either move far from or come close to the mean of its contaminated counterpart, thus a factor named contamination factor f_c is introduced. $(1 - \nu)l$ of all inputs have relatively large f_c obtained from the inverse of cumulative distribution function at some probabilities. However, there is no unique inverse of cumulative distribution function for multivariate Gaussian distribution, so the f_c is not unique as well. But if the moving direction of this input is known, f_c can be retrieved by recursively invoking the multivariate cumulative distribution function. Without loss of difficulty in generating adversarial data sets, we prefer a more simple way to obtain f_c, which is derived from the inverse of standard Gaussian cumulative distribution function at the corresponding probability in a random number drawn from a uniform distribution on the unit interval. In this experiment, small f_c is obtained simply from half of a random number.

The previous discussion in Chapter 2 shows that the noise model dependent on the function and the distribution is probably the most adversarial model for algorithms. In classification problems, such contaminated data sets can be created by including both two kinds of contamination, which depend on the target function and the distribution from both of which inputs are originally generated in data sets. However, there is no information about the target function and the distribution along with data sets. So the information is required before it can be used to contaminate data sets.

When no prior knowledge of the distribution of inputs is available, it is possible to assume that the wanted distribution is composed of distinct subclasses or clusters. Each subclass is characterised by a set of parameters describing the mean and variation of the spectral components. Gaussian mixture model (GMM) is a statistically mature method for clustering and forming a probabilistic mixture model that is composed of a number of component clusters. Here, we introduce a MATLAB toolbox named Cluster (Bouman, 1997) for modelling Gaussian mixtures. Cluster programme is an unsupervised algorithm based on the expectation-maximisation (EM) algorithm and the minimum description length (MDL) order estimation criteria. This programme can also estimate the number of clusters directly from the inputs. Then the original input from one class is set to move

towards the mean of one of all clusters in the other class, the chosen cluster has the largest posterior probability for the original input, weighted by the cluster probability. Here, $\boldsymbol{x}_m$ denotes the mean of the selected cluster.

The contamination depending on the target function is such a noise that the contaminated original input is set to move towards a selected input from the other class, making the line between these two inputs geometrically vertical to the target hyperplane. When the target function is available, we can directly search the proper input from the other class. But the target function is not always known. In this experiment, we proceed a standard SVC over original inputs to approximate the target function before contamination. It is difficult to compute the moving direction straightway from the target function and its complicated expression $f(\alpha, b)$. Alternatively, a small multi-dimensional ball is created by fixing the original input as its centre. Searching possible solutions for the expression as follows,

$$\begin{aligned} &\max_{\boldsymbol{x}_s \in \text{surface of the ball}} \quad \|d_s - d_{io}\| \\ &d_{io} = y_{io}\left(\sum_{j=1}^{l} y_j \alpha_j K(\boldsymbol{x}_{io}, \boldsymbol{x}_j) + b\right), \\ &d_s = y_s\left(\sum_{j=1}^{l} y_j \alpha_j K(\boldsymbol{x}_s, \boldsymbol{x}_j) + b\right), \end{aligned} \tag{6.11}$$

where $\boldsymbol{x}_s$ are on the surface of the ball. We have the optimum $\boldsymbol{x}_{so}$. But the high dimensionality of the input space will lead to a large number of $\boldsymbol{x}_s$, make the contamination time-consuming. For instance, an n-dimensional original input will have k^n $\boldsymbol{x}_s$ when k observation points are set in each dimension of the space. Sometimes, we just simply use inputs from the chosen cluster of the other class to replace $\boldsymbol{x}_s$. In this case, (6.11) is changed to

$$\max_{\boldsymbol{x} \in \text{the other class}} \frac{\|d - d_{io}\|}{\|\boldsymbol{x} - \boldsymbol{x}_{io}\|}. \tag{6.12}$$

Both $\boldsymbol{x}_m$ and $\boldsymbol{x}_{so}$ contain the information acquired from the target function and the distribution. We can then compute the moving direction, and derive the new position of the corresponding original input being contaminated.

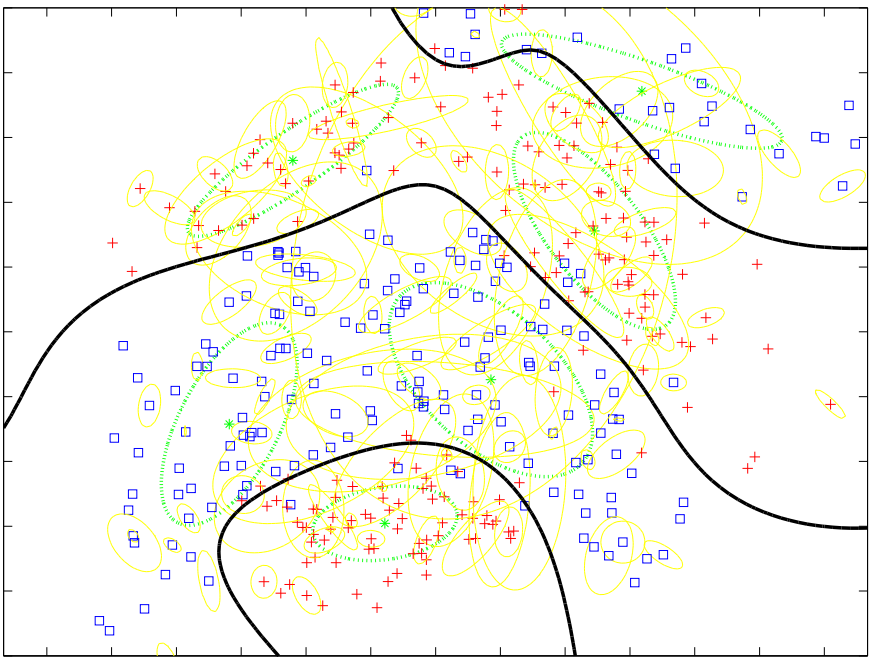
$$\boldsymbol{x}_i = \boldsymbol{x}_{io} \pm \frac{f_c \mathbf{M}_i(\boldsymbol{x}_{io} - \boldsymbol{x}_m)}{\sqrt{(\boldsymbol{x}_{io} - \boldsymbol{x}_m)^T f_c \mathbf{M}_i(\boldsymbol{x}_{io} - \boldsymbol{x}_m)}}, \tag{6.13}$$

and

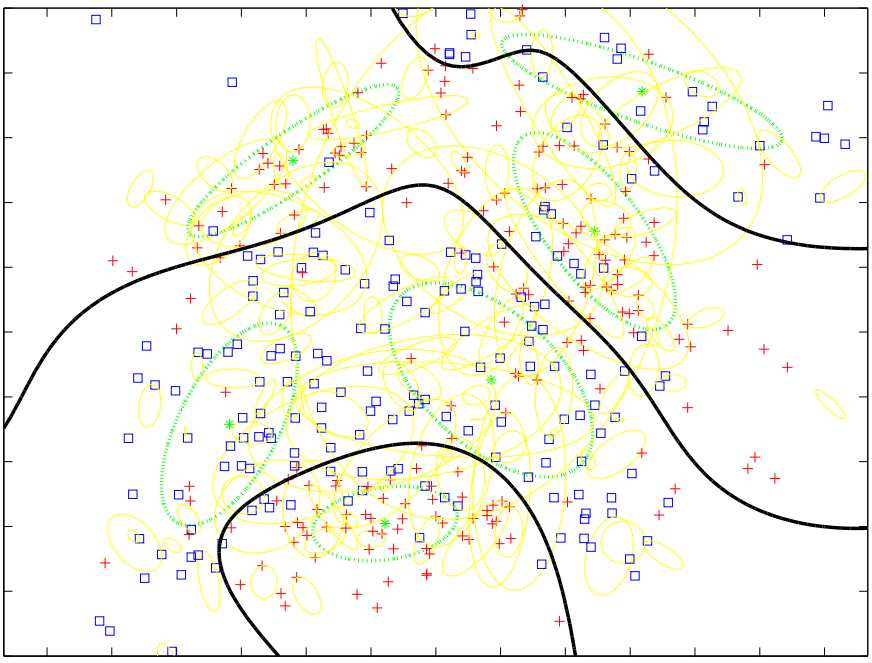
$$\boldsymbol{x}_i = \boldsymbol{x}_{io} \pm \frac{f_c \mathbf{M}_i(\boldsymbol{x}_{io} - \boldsymbol{x}_{so})}{\sqrt{(\boldsymbol{x}_{io} - \boldsymbol{x}_{so})^T f_c \mathbf{M}_i(\boldsymbol{x}_{io} - \boldsymbol{x}_{so})}}. \tag{6.14}$$

(6.13) and (6.14) are after the contamination depending on the distribution and the target function respectively. Parameter ζ is introduced to determine using either contamination or using both of them over data set. During the contamination procedure,

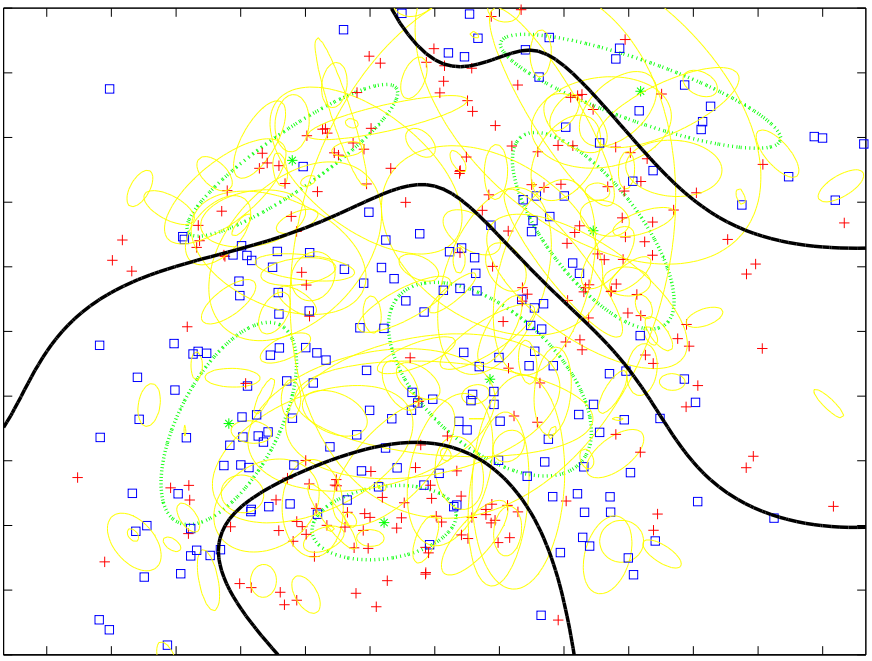
a generated random number greater than ζ leads to contaminating the corresponding original input with the information from target function, and vice versa. Each equation has two results, one is the closest point to the mean of selected cluster or to the target function, the other is the farthest point to them. The selection of two results depends on the settings of contamination. For instance, when the contamination depending on the target function is applied to the original input, its closest point is chosen by computing (6.11) under Bi and Zhang's setting. Some results of a banana data set from Gunnar Rätsch's repositories are shown as follows. To show the difference more



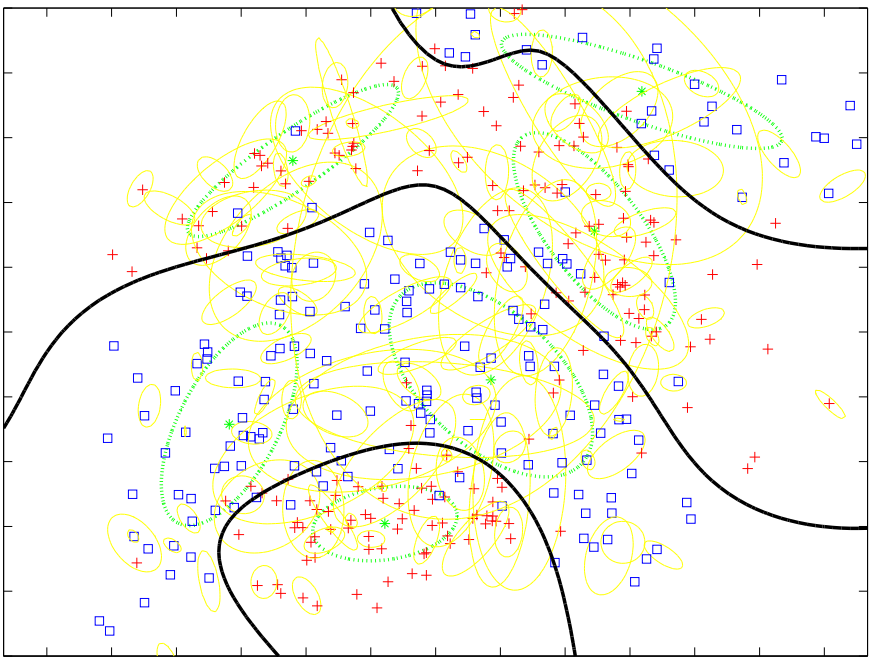
(a) Original Data Set



(b) Contaminated Data Set with $\tau = 0.8$, $\nu = 0.6$ and $\zeta = 1$



(c) Contaminated Data Set with $\tau = 0.8$, $\nu = 0.6$ and $\zeta = 0$



(d) Contaminated Data Set with $\tau = 0.8$, $\nu = 0.6$ and $\zeta = 0.5$

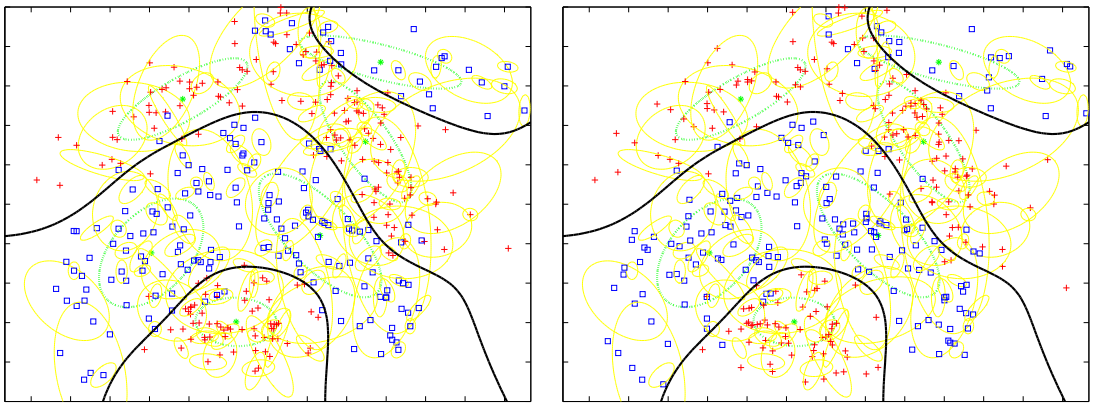
FIGURE 6.2: The contamination results of banana data set under Bi and Zhang's setting. Thick solid lines are the estimated true target function obtained from standard SVC, star and dotted lines represent the results of Gaussian mixtures.

distinctively between the contamination results, light colour is used to plot Gaussian noise in Figure 6.2. Data set is contaminated with $\tau = 0.8$ and $\nu = 0.6$, which means 20% of all inputs are contaminated by large Gaussian noise and 40% of all inputs are severely contaminated by their corresponding Gaussian noise. $\zeta = 1$ means that all the

contamination depend on the distribution. While, $\zeta = 0$ means that all the contamination depend on the target function. $\zeta = 0.5$ means that the contamination depending on the target function is chosen for almost half of all inputs, and the other half select the other kind of contamination. $\zeta = 1$ leads to a higher level of concentration of part of the contaminated inputs which have crossed the target function, because they all attempt to move close to the means of their corresponding limited number of clusters of the other class. $\zeta = 0.5$ can well disperse these inputs and make the formed data set more adversarial by introducing the contamination upon the knowledge from both the distribution and the target function. Without further notice, $\tau = 0.8$, $\nu = 0.6$ and $\zeta = 0.5$ are used as the default setup in contamination for all data sets thereafter.

6.2.3 The Reverse Setting

The reverse setting is named after its opposite design to Bi and Zhang's setting. Inputs being contaminated under the reverse setting are set to move far from the other class by trying to be away from the original boundary. Unlike Bi and Zhang's setting, the large and small Gaussian noise are assigned to inputs not according to their distances to the target boundary but based on random selection. Some results of banana data set under the reverse setting are shown as follows,



(a) Original Data Set

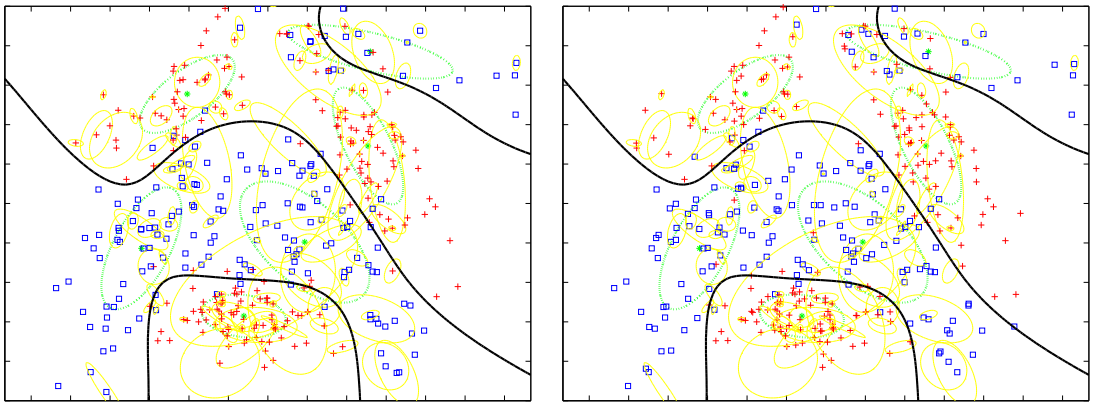
(b) Contaminated Data Set with $\tau = 0.8$, $\nu = 0.6$ and $\zeta = 0.5$

FIGURE 6.3: The contamination results of banana data set under the reverse setting. Thick solid lines are the estimated true target function obtained from standard SVC, star and dotted lines represent the results of Gaussian mixtures.

6.2.4 The General Setting

The general setting introduces a more normal kind of contamination, in which the moving directions of inputs being contaminated are not related to any factor. The moving directions are generated randomly. Consequently, these inputs can either move close to

or move far from the target function and clusters belonging to the other class. Like the reverse setting, the large Gaussian noise have no priority to be assigned to the inputs close to the target boundary. Due to its contamination procedure, the general setting has the most trivial changes of all three settings by comparing the original data set and the data set being contaminated. In general, the contamination under Bi and Zhang's



(a) Original Data Set

(b) Contaminated Data Set with $\tau = 0.8$, $\nu = 0.6$ and $\zeta = 0.5$

FIGURE 6.4: The contamination results of banana data set under the general setting. Thick solid lines are the estimated true target function obtained from standard SVC, star and dotted lines represent the results of Gaussian mixtures.

setting, the general setting and the reverse setting can be deemed as severe, moderate and light contamination respectively.

6.3 Experimental Platform

The experimental platform is setup on a INTEL core 2 quad Q6600 computer with 8GB ram memory. The operation system is windows xp 64-bit x86 professional version.

At this stage, SeDuMi is too time-consuming to handle experiments because of the large size and the high dimensionality of data sets. MOSEK is then introduced in the experiments since it is generally reckoned as the best of all optimisation softwares in solving SOCP problems (Mittelmann, 2003). More recently, MOSEK still overall outperforms other softwares including LOQO and SeDuMi in Mittelmann's latest SOCP benchmark (Mittelmann, 2008).

6.3.1 MOSEK

The MOSEK optimisation software is designed to solve large-scale and sparse mathematical optimisation problems. MOSEK has specialised solvers for linear problems, conic quadratic optimisation problems, mixed integer problems, etc. MOSEK provides

optimisation toolboxes for MATLAB in both 32-bit and 64-bit windows system, which combine the convenience of MATLAB with the speed of C code. The main computational engine in the MOSEK optimisation toolbox is a primal-dual type interior-point algorithm. The conic optimisation problem can accommodate a SOCP problem when its cone constraint specifies that the vector formed by a set of decision variables is constrained to lie within a second order cone. The following cones (MOSEK, 2008) in constraints of conic optimisation problems are used in the experiments,

$$\text{Qcone} : \left\{ \boldsymbol{x} \in \mathbb{R}^n : x_1 \geq \sqrt{\sum_{j=2}^{n} x_j^2} \right\},$$

$$\text{Rcone} : \left\{ \boldsymbol{x} \in \mathbb{R}^n : 2x_1 x_2 \geq \sum_{j=3}^{n} x_j^2 \right\},$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ and Qcone, Rcone denote the quadratic cones and the rotated quadratic cones respectively.

The only available optimiser for conic optimisation problems is an interior-point optimiser, which is an implementation of the self-dual and homogeneous algorithm. Three parameters control when conic interior-point optimiser terminates and the accuracy of the solution obtained by the interior-point optimiser, including primal feasibility tolerance for the primal solution, dual feasibility tolerance for the dual solution and relative primal-dual gap tolerance. The values of these parameters are changed from their default values 10^{-8} to 10^{-6} in experiments. Besides, the interior-point optimiser in MOSEK have been parallelised. We can take advantages of multiple CPUs because of the interior-point optimiser used for conic optimisation problems. The number of CPUs can be set to 2 for INTEL dual core CPU and 4 for INTEL quad core CPU. In a test run for INTEL quad core CPU, four-threaded performance is at least three times faster than single-threaded performance.

6.3.2 MATLAB External Interface

Besides the time spent within MOSEK solvers, still a lot of time is consumed in optimisation procedure, especially in the loops of forming large matrices for MOSEK solvers and iterations of computing multiplication of matrices and vectors for the measures wanted. To make the code more efficient, part of the MATLAB programs, including construction of large matrices and multiplication of matrices and vectors, are replaced by C subroutines called from the MATLAB command line as if they were built-in functions. These C programs are called binary MATLAB executable files (MEX-files), which are dynamically linked subroutines that the MATLAB interpreter loads and executes (MATLAB, 2007).

The compiler used on 32-bit Windows platform is the GNU compiler collection (gcc) from minimalistic GNU for Windows (MinGW) (MinGW, 2008), which provides a complete open source programming tool set. On 64-bit Windows platform, standard C99 compatible C compiler is used for building MEX-files from Pelles C (Orinius, 2008), which is a freely complete development kit for Windows and Pocket PC.

6.3.3 Parameters Setup

In previous chapters, the parameters of different algorithms are fixed during the classification, including the selected kernel and its parameters. These preset parameters can help us compare statistic and geometric difference explicitly between these chosen algorithms. In the experiments, we use the same kernel function as the one used by Rätsch (2001b), a Gaussian RBF kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$ which is generally a reasonable first choice because of its following characters. Firstly, the Gaussian RBF kernel can handle the case when the relation between class labels and attributes is non-linear since it non-linearly maps inputs into a higher dimensional space. Furthermore, the linear kernel is a special case of Gaussian RBF kernel as Keerthi and Lin (2003) shows that the linear kernel with a regularisation parameter has the same performance as the Gaussian RBF kernel with parameters C and σ. Secondly, the Gaussian RBF kernel has less hyperparameters that influence the complexity of model selection than other non-linear kernel functions. Finally, the Gaussian RBF kernel has less numerical difficulties. Its value is limited between 0 and 1 in contrast to polynomial kernel whose value may go to infinity or zero while the degree is large. The parameter σ of the Gaussian RBF kernel is inherited from the results shown by Rätsch (2001a).

However, different algorithms can hardly reach their best performance with the same preset regularisation parameters. Cross-validation can be used to find well-suit parameters for each algorithm. Cross-validation, sometimes called rotation estimation, is a technique for assessing how the results of a statistical analysis will generalise to an independent data set. With the advantages of mutually exclusive subsets and less computational cost, k-fold cross-validation, out of other validation methods such as repeated random sub-sampling validation and leave-one-out cross-validation, is selected to assess serial regularisation parameters in the experiments. The number of split subsets k determines the bias and the variance of an estimated parameter of an algorithm. The higher the number of folds k, the lower the bias. Whereas increasing k too much may increase the variance (Kohavi, 1995, 1996). Meanwhile, 10-fold cross-validation is commonly used. Therefore, $k = 10$ is selected in the experiments for the tradeoff between the bias and the variance. Stratification can slightly reduce the variance as well. So stratified 10-fold cross-validation is subsequently used in the experiments, this means that each fold contains roughly the same proportions of the two types of class labels.

6.4 Experimental Results

Besides NMCU and MPE, a new measure, the number of misclassified edges of the uncertainties (NMEU), is introduced to evaluate the performance of classifying the corrupted inputs for different algorithms. As its name suggests that NMEU calculates the number of input uncertainties whose nearest edges to the optimal hyperplane have been misclassified by the classifier obtained. NMEU can better reflect how many unknown original inputs are likely to be correctly classified by the optimal solution than NMCU. Like NMCU, NMEU is collected by evaluating the learner's optimal solution, which is obtained from a contaminated training set, in a test set corrupted by the same contamination used in that training set. NMCU, NMEU and MPE are also called the classification measures and TME is called the restoration measure.

Additionally, to show how the classification results can be affected by the contamination, another traditional SVC is introduced and directly trained with contaminated data sets and the optimal regularisation parameter given by Rätsch (2001a) instead of using cross-validation to search the regularisation parameter. For convenience, this approach is denoted by SVCRaetsch in the following comparisons. Indeed, SVCRaetsch can also be applied to the uncontaminated original data set to approximate the true target function. Moreover, a measure is defined here to illustrate the level of classification contamination for each training set, this measure is termed as the classification contamination level (CCL) which calculates the difference of the number of original inputs and corrupted inputs misclassified by the true target function respectively before and after contamination. For instance, if 6% original inputs are misclassified by the true target function, and after a contamination affecting this training set, 11% observed inputs are misclassified by this true target function, then CCL is 5% for this contaminated training set. Generally, the average classification contamination level (ACCL) is used instead of CCL to illustrate the level of classification contamination for each contamination setting by averaging CCL of all available contaminated training sets under this setting.

All the experimental data sets used in this chapter are introduced from Gunnar Rätsch's repositories (Rätsch, 2001a). There are in total 100 training sets and 100 test sets in each proposed example. To give consideration to both computational expense and the minimal request of the number of data sets for the Friedman test, Bi and Zhang's setting, the general setting and the reverse setting are applied to the first 40 data sets, the middle 30 data sets (from the 41st to the 70th data set) and the last 30 data sets respectively throughout the experiments. Besides, the significance for the Friedman test and especially for the post-hoc tests is chosen as 0.1 in all subsequent experiments since the critical values of significance 0.05 are too large to discriminate some slight difference between experimental approaches quite often. (Some examples of the outputs of statistical tests are shown in Appendix C.)

6.4.1 Banana Data Sets

Banana data sets are introduced as a two-dimensional example comprising toy data sets for classification subject to input uncertainty. Each training set contains 400 observed data $\{\boldsymbol{z}_i, y_i\}$, where $\boldsymbol{z}_i \in \mathbb{R}^2$ and $y \in \mathbb{R}$, and each test set has 4900 data in it. The training kernel is chosen as Gaussian RBF with $\sigma = 1$ and the optimal regularisation parameter is $C = 316.23$ according to Rätsch (2001a). Trained with uncertain inputs that are obtained by contamination under Bi and Zhang's setting, different algorithms can be evaluated by the average ranks of their different measures, which are shown is the table below. With in total six algorithms and 40 data sets, F_F is distributed according to

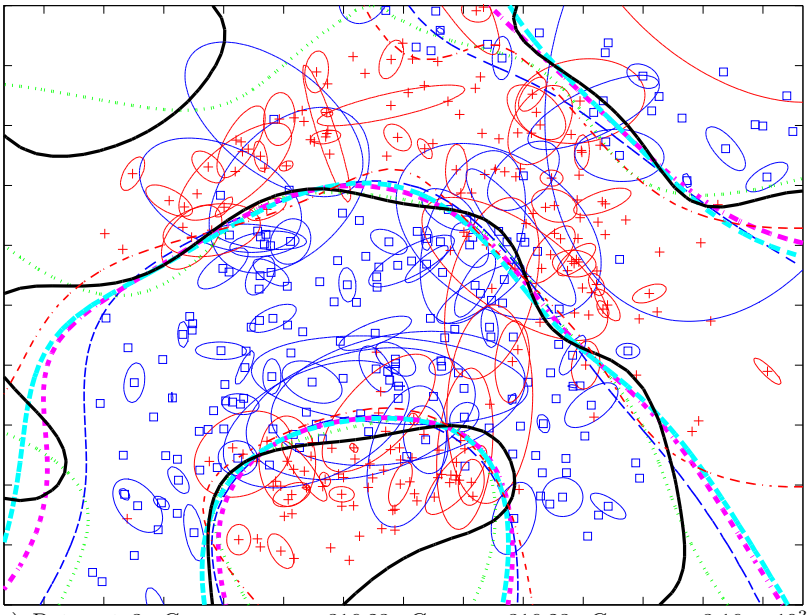
Algorithm \ Measure	NMCU	NMEU	MPE	TME
USVC	4.737	3.750	3.375	5.025
SVC	3.237	2.900	3.000	3.225
TSVC	2.950	2.663	2.750	1.925
AUSVC	3.750	3.275	2.825	4.125
MPSVC	3.400	3.487	3.875	3.188
SVCRaetsch	2.925	4.925	5.175	3.513

TABLE 6.1: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the first 40 banana data sets (from the 1st data set to the 40th data set) contaminated under Bi and Zhang's setting, whose ACCL is 6.03% (The average percentage of misclassified inputs in all inputs is 7.42% before contamination and 13.45% after contamination).

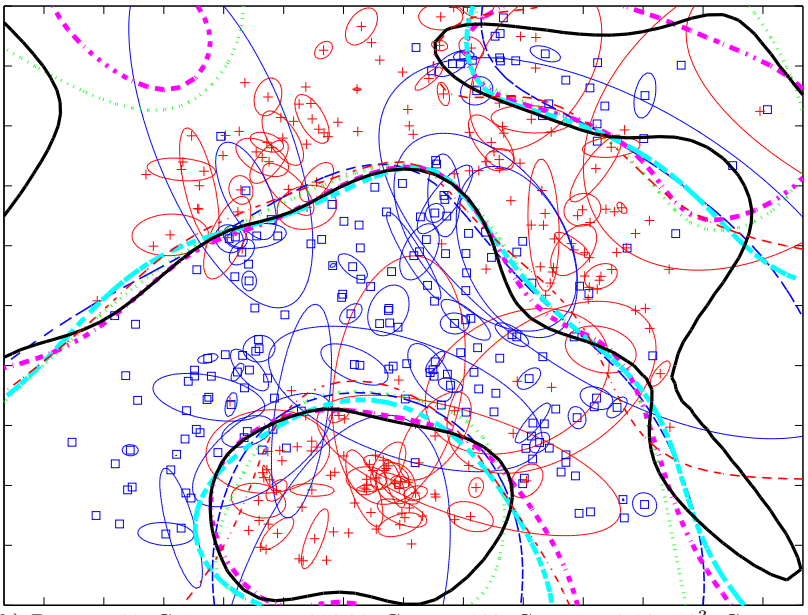
the F distribution with $6-1=5$ and $(6-1)\times(40-1)$ degrees of freedom. The critical value of $F(5, 195)$ for significance $\alpha = 0.1$ is 1.88, so the null hypothesis that these algorithms have similar performance is rejected for all measures according to equation (6.6) and the average ranks of these measures.

The post-hoc analysis is then proceeded and all four post-hoc tests are considered in coming to draw a conclusion. In the comparison of NMCU, USVC performs significantly worse than all other algorithms. While in the comparison of NMEU, SVCRaetsch performs significantly worse than all other algorithms and TSVC performs significantly better than USVC. In the comparison of MPE, SVCRaetsch performs significantly worse than all other algorithms, TSVC and AUSVC perform significantly better than MPSVC. In the comparison of TME, USVC performs significantly worse than TSVC, MPSVC, SVC and SVCRaetsch, where TSVC is significantly better than all other algorithms. Therefore, TSVC undoubtedly has the best performance of classification and restoration of all algorithms under Bi and Zhang's setting. USVC and SVCRaetsch are likely the worst-performance algorithms under this setting. Improved from USVC, AUSVC and MPSVC can achieve lower NMCU and NMEU than USVC, but the MPE of MPSVC is higher than those of USVC and AUSVC due to its optimisation problem which has smaller individual probability confidence than USVC and AUSVC at optimum. The optimisation problem of MPSVC simultaneously leads to generally lower TME than those

of all other algorithms except TSVC. This means MPSVC can better recover the true target function than many other approaches when the training and test banana sets are contaminated by noise under Bi and Zhang's setting. The results of the 3rd and 38th contaminated banana training sets are shown in Figure 6.5.



(a) Data set 3, $C_{\text{SVCRaetsch}} = 316.23$, $C_{\text{SVC}} = 316.23$, $C_{\text{USVC}} = 3.16 \times 10^3$, $C_{\text{TSVC}} = 31.62$, $C_{\text{AUSVC}} = 10$, $L_{\text{hi}} = 1$, $L_{\text{lo}} = 100$



(b) Data set 38, $C_{\text{SVCRaetsch}} = 316.23$, $C_{\text{SVC}} = 100$, $C_{\text{USVC}} = 3.16 \times 10^3$, $C_{\text{TSVC}} = 10$, $C_{\text{AUSVC}} = 316.23$, $L_{\text{hi}} = 1$, $L_{\text{lo}} = 31.62$

FIGURE 6.5: Selected results from the 3rd and 38th banana training sets contaminated by Bi and Zhang's setting. USVC is represented by black solid line, SVC is represented by green dotted line, TSVC is represented by blue dashed line, AUSVC is represented by thick magenta dash-dot line, MPSVC is depicted by thick cyan dashed line and the true target function is approximated by SVCRaetsch trained with the original noiseless data and illustrated as red dash-dot line.

From the point of view for recovering the true target function, it can be seen that USVC is severely overfitted in the figure. While SVC is also likely to be overfitted in some sparsity of training inputs, such as the top left part of Figure 6.5(a) and 6.5(b), though it has a generally better performance with banana sets under Bi and Zhang's setting. AUSVC can generally reduce the possibility of overfitting happening in USVC by using its adaptive constraints, but AUSVC can still have overfitting problems in some cases, such as Figure 6.5(b). MPSVC has a close performance to TSVC, and can better recover the true target function than USVC and AUSVC from the contaminated data by driving some individual probability confidence to negative through its problem optimised with MPE.

As what has been discussed before, it is arguable that TSVC will also outperform other algorithms in different settings of contamination. The following table compares the average ranks of different measures of these algorithms trained with uncertain inputs that are obtained from original inputs contaminated by the general setting. Calculating equation

Measure / Algorithm	NMCU	NMEU	MPE	TME
USVC	3.800	1.867	1.833	3.467
SVC	2.117	2.600	2.350	2.433
TSVC	4.167	3.633	3.500	4.100
AUSVC	3.417	2.950	3.067	3.833
MPSVC	3.067	4.317	4.800	2.817
SVCRaetsch	4.433	5.633	5.450	4.350

TABLE 6.2: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the middle 30 banana data sets (from the 41st data set to the 70th data set) contaminated under the general setting, whose ACCL is 2.22% (The average percentage of misclassified inputs in all inputs is 7.73% before contamination and 9.95% after contamination).

(6.6) for all four measures, we see that all F_F are larger than 1.89, which is the critical value of $F(5, 145)$ for significance $\alpha = 0.1$. Therefore, the null hypothesis is rejected. After considering all four post-hoc tests, the following conclusion can be drawn that SVC performs significantly better than all other algorithms, while MPSVC performs significantly better than SVCRaetach and TSVC in the comparison of NMCU. In the comparison of NMEU, SVCRaetsch performs significantly worse than all other algorithms, MPSVC performs significantly worse than USVC, SVC and AUSVC, TSVC performs significantly worse than USVC and SVC, USVC is significantly better than AUSVC. In the comparison of MPE, USVC, SVC, AUSVC and TSVC are significantly better than SVCRaetsch and MPSVC, USVC and SVC are significantly better than TSVC, USVC performs significantly better than AUSVC. In the comparison of TME, SVC performs significantly better than all other algorithms except MPSVC, while MPSVC is significantly better than SVCRaetsch and TSVC. Therefore, SVC is the one that has the best performance of classification of all algorithms under the general setting though two of all three classification measures of SVC rank lower than those of USVC, this is because

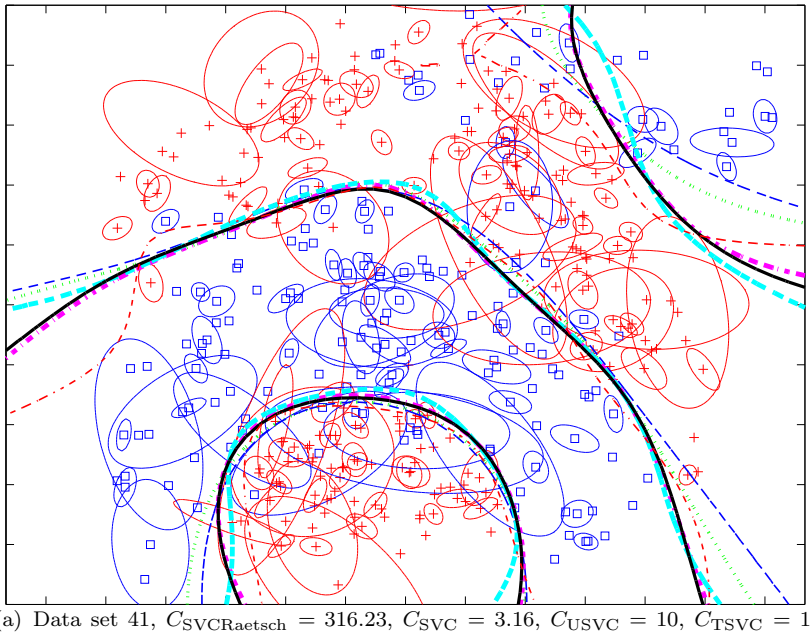
none of the classification measures of SVC are significantly worse than those of USVC, but NMCU of USVC is significantly worse than that of SVC. USVC can be deemed to have the second best performance of classification of all approaches. On the other hand, SVC has the best performance of restoration, MPSVC ranks the second and has close performance to SVC. The results of the 41st and 50th banana training sets contaminated by the general setting are shown in Figure 6.6. The reason that SVC can better recover the true target function is the contamination under the general setting, which, unlike Bi and Zhang's setting, does not force the original inputs move towards the other class. Therefore, the mean values of the observed uncertain inputs are unlikely to move across the original classifier, but generally stay close to their corresponding original inputs. SVC can then achieve the best performance of classification and restoration with its optimisation problem including only the mean values but no other distribution information of the uncertain inputs.

Another available setting of contamination is the reverse setting, under which the original inputs corrupted by noise move towards their own class in accordance with the distribution of original inputs and the true target function. These algorithms can be compared with each other by the average ranks of all four measures illustrated in the table listed below,

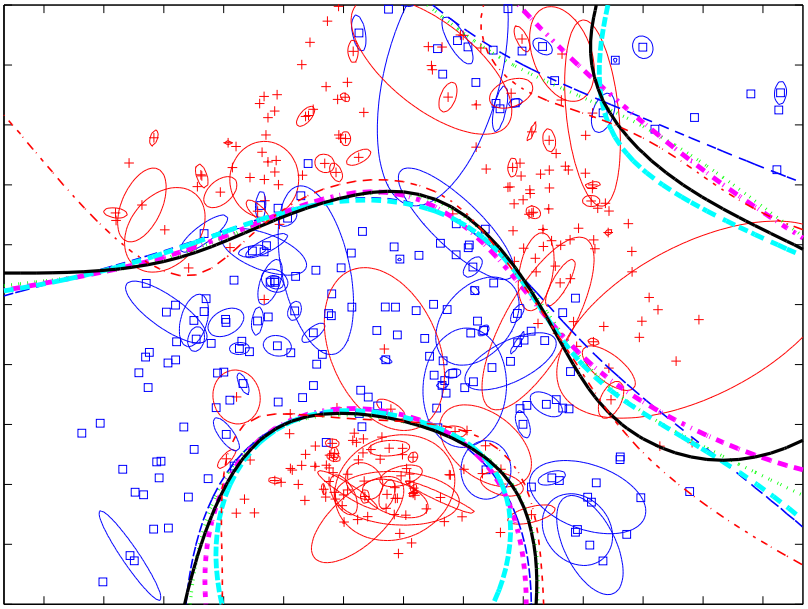
Algorithm \ Measure	NMCU	NMEU	MPE	TME
USVC	3.538	1.692	1.769	3.115
SVC	2.692	3.462	3.615	3.423
TSVC	2.577	3.423	3.423	3.423
AUSVC	3.308	3.000	2.538	3.192
MPSVC	3.846	3.692	3.923	2.769
SVCRaetsch	5.038	5.731	5.731	5.077

TABLE 6.3: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the last 30 banana data sets (from the 71st data set to the 100th data set) contaminated under the reverse setting, whose ACCL is -0.59% (The average percentage of misclassified inputs in all inputs is 7.73% before contamination and 7.14% after contamination).

In the comparison of NMCU, SVCRaetsch is significantly worse than all other approaches, TSVC and SVC are significantly better than MPSVC. In the comparison of NMEU, SVCRaetsch performs significantly worse than all other algorithms and USVC is significantly better than all other algorithms. In the comparison of MPE, SVCRaetsch is significantly worse than all other algorithms, USVC and AUSVC perform significantly better than all other approaches. In the comparison of TME, the only conclusion made is SVCRaetsch performs significantly worse than all other algorithms. According to the average ranks, TSVC has a close performance to SVC because the contaminated inputs are likely to be separable under the reverse setting, TSVC can reach its optimum at very first iterations and its optimal solution is then close to that of SVC. USVC has the best



(a) Data set 41, $C_{\text{SVCRaetsch}} = 316.23$, $C_{\text{SVC}} = 3.16$, $C_{\text{USVC}} = 10$, $C_{\text{TSVC}} = 1$, $C_{\text{AUSVC}} = 10$, $L_{\text{hi}} = 1$, $L_{\text{lo}} = 31.62$

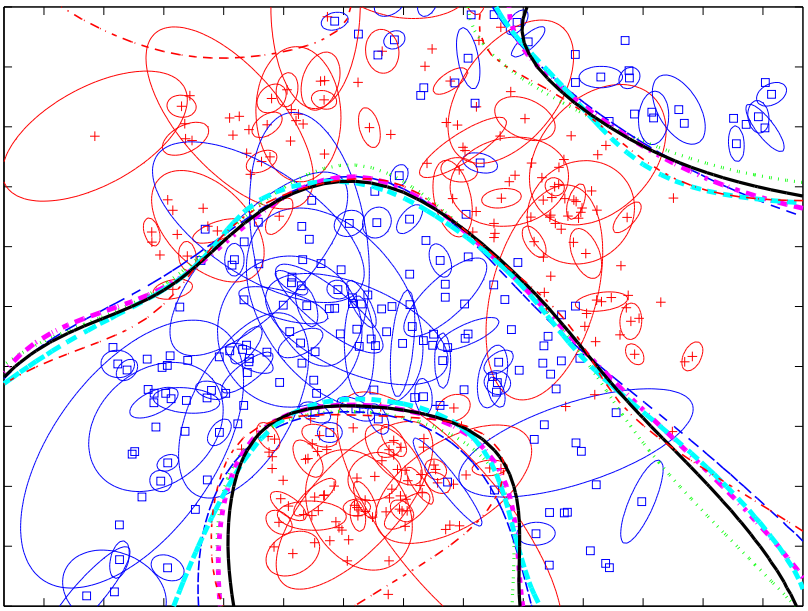


(b) Data set 50, $C_{\text{SVCRaetsch}} = 316.23$, $C_{\text{SVC}} = 3.16$, $C_{\text{USVC}} = 31.62$, $C_{\text{TSVC}} = 1$, $C_{\text{AUSVC}} = 1$, $L_{\text{hi}} = 1$, $L_{\text{lo}} = 0.32$

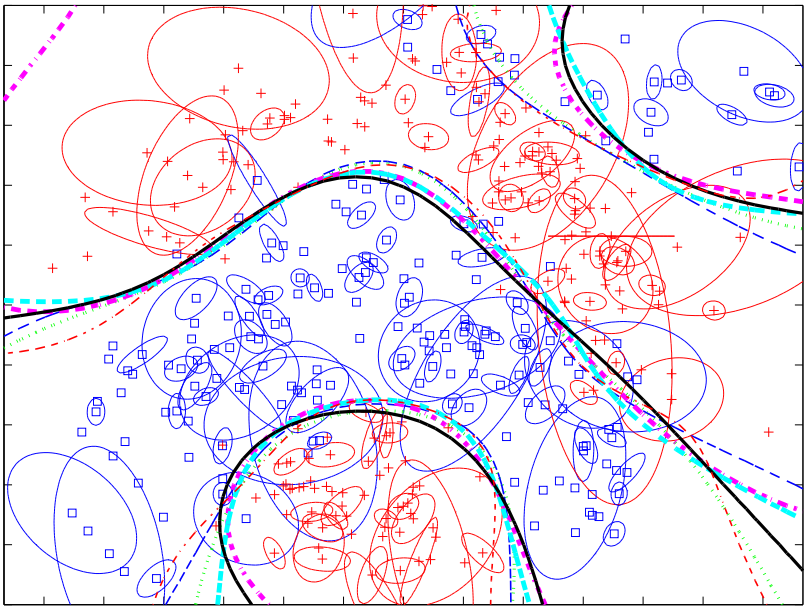
FIGURE 6.6: Selected results from the 41st and 50th banana training sets contaminated by the general setting. USVC is represented by black solid line, SVC is represented by green dotted line, TSVC is represented by blue dashed line, AUSVC is represented by thick magenta dash-dot line, MPSVC is depicted by thick cyan dashed line and the true target function is approximated by SVCRaetsch trained with the original noiseless data and illustrated as red dash-dot line.

performance of classification by significantly outperforming all other algorithms in the comparison of NMEU and MPE, AUSVC ranks the second in the classification with only the average rank of NMEU significantly lower than that of USVC. Though not a single algorithm performs significantly better than all other algorithms except SVCRaetsch in recovering the true target function, MPSVC, with USVC and AUSVC is generally better

than SVC and TSVC with higher average ranks of TME. The results of the 76th and 77th banana training sets contaminated by the reverse setting are shown in Figure 6.7.



(a) Data set 76, $C_{\text{SVCRaetsch}} = 316.23$, $C_{\text{SVC}} = 10$, $C_{\text{USVC}} = 31.62$, $C_{\text{TSVC}} = 1$, $C_{\text{AUSVC}} = 10$, $L_{\text{hi}} = 1$, $L_{\text{lo}} = 3.16$



(b) Data set 77, $C_{\text{SVCRaetsch}} = 316.23$, $C_{\text{SVC}} = 31.62$, $C_{\text{USVC}} = 10$, $C_{\text{TSVC}} = 31.62$, $C_{\text{AUSVC}} = 31.62$, $L_{\text{hi}} = 1$, $L_{\text{lo}} = 1$

FIGURE 6.7: Selected results from the 76th and 77th banana training sets contaminated by the reverse setting. USVC is represented by black solid line, SVC is represented by green dotted line, TSVC is represented by blue dashed line, AUSVC is represented by thick magenta dash-dot line, MPSVC is depicted by thick cyan dashed line and the true target function is approximated by SVCRaetsch trained with the original noiseless data and illustrated as red dash-dot line.

In General, these approaches perform at different levels over contaminated banana data sets obtained from different settings. USVC performs very well under the general and

reverse settings, but it is likely to be overfitted under Bi and Zhang's setting. Inheriting the same optimisation structure from USVC, AUSVC generally improves the performance of classifying the contaminated inputs under Bi and Zhang's setting because its adaptive constraint can effectively lower the probability of an unknown original input being correctly classified when the corresponding observed input is misclassified. However, the adaptive constraints can also negatively affect the classification performance of AUSVC, especially when the general and reverse settings are applied. Under the general and reverse settings, if the mean value of any uncertain input is erroneously discriminated as a misclassified input in one of AUSVC's iterations, then the corresponding individual probability confidence is likely to be reduced continuously afterwards since the inputs originally being misclassified are limited (This can be proved by the classification measures of USVC, which indeed is the very first iteration of AUSVC.) and resultantly not many other adaptive constraints corresponding to these inputs can be adjusted to stop reducing the wrong individual probability confidence. While due to the possibly large number of misclassified inputs under Bi and Zhang's setting, it is easy to learn the wrongly misclassified input in the most recent iteration as many other adaptive constraints related to these misclassified inputs may be adjusted to correctly classify this wrongly misclassified input correctly in the current iteration, stoping reducing its corresponding individual probability confidence. That is why AUSVC has a worse performance than USVC under the general and reverse settings. TSVC and SVC are the best solutions in both classifying the contaminated inputs and recovering the true target function under Bi and Zhang's setting and the general setting respectively. Considering all three settings, MPSVC is the best overall algorithm for recovering the true target function by combining the results obtained from AUSVC and the measure MPE. MPSVC tries to find a tradeoff between classification and restoration.

6.4.2 Titanic Data Sets

Titanic data sets from Gunnar Rätsch's benchmark repository are also contaminated by noise and introduced into classification subject to input uncertainty. This is a three-dimensional example, each training set contains 150 observed data and each test set has in total 2051 data in it. The training kernel is chosen as Gaussian RBF with $\sigma = 2$ and the optimal regularisation parameter is $C = 10^5$ according to Rätsch (2001a). With the uncertain inputs contaminated by Bi and Zhang's setting, the average ranks of different measures of these algorithms are shown in Table 6.4. From the table, we see that the null hypothesis of the Friedman test is rejected for all measures under Bi and Zhang's setting in accordance with equation (6.6) before more analysis comes from the post-hoc test. AUSVC, MPSVC, SVC and USVC perform significantly better than TSVC and SVCRaetsch in the comparison of NMCU. In the comparison of NMEU and MPE, SVCRaetsch is the one with the worst performance, while TSVC performs worse than all other algorithms except SVCRaetsch. USVC and AUSVC are significantly better than

Algorithm \ Measure	NMCU	NMEU	MPE	TME
USVC	3.125	1.488	1.300	3.563
SVC	3.125	3.638	3.800	3.737
TSVC	4.325	4.925	4.925	4.150
AUSVC	2.487	1.962	1.900	3.025
MPSVC	2.987	3.087	3.175	2.962
SVCRaetsch	4.950	5.900	5.900	3.563

TABLE 6.4: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the first 40 titanic data sets (from the 1st data set to the 40th data set) contaminated under Bi and Zhang's setting, whose ACCL is 6.05% (The average percentage of misclassified inputs in all inputs is 18.73% before contamination and 24.78% after contamination).

SVC and MPSVC. In the comparison of TME, the only conclusion can be drawn is that MPSVC and AUSVC are significantly better than TSVC. Therefore, AUSVC has the best overall performance in both classification and restoration, though it is outperformed by USVC and MPSVC in NMEU, MPE and TME respectively. This is because in these comparisons, USVC and AUSVC can be deemed as the best performers in classification, while MPSVC and AUSVC can be deemed as the best performers in restoration, but USVC and MPSVC have not gained absolute advantages over AUSVC. On the other hand, SVC and TSVC perform not as well as USVC, AUSVC and MPSVC in both classification and restoration.

When the general setting is applied to contaminate the original inputs, the average ranks of all four measures of these algorithms are obtained and illustrated in Table 6.5. In

Algorithm \ Measure	NMCU	NMEU	MPE	TME
USVC	2.400	1.917	1.500	3.550
SVC	2.867	3.367	3.333	3.400
TSVC	5.067	4.950	5.000	3.900
AUSVC	2.867	1.933	1.700	3.467
MPSVC	2.183	2.933	3.567	2.667
SVCRaetsch	5.617	5.900	5.900	4.017

TABLE 6.5: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the middle 30 titanic data sets (from the 41st data set to the 70th data set) contaminated under the general setting, whose ACCL is 1.00% (The average percentage of misclassified inputs in all inputs is 20.53% before contamination and 21.53% after contamination).

the comparison of NMCU, MPSVC, USVC, SVC and AUSVC are significantly better than SVCRaetsch and TSVC. In the comparison of NMEU and MPE, SVCRaetsch has the worst performance of all algorithms, TSVC performs significantly better than SVCRaetsch, but significantly worse than the rest approaches, among which USVC and AUSVC are significantly better than SVC and MPSVC. In the comparison of TME,

these algorithms perform at very close levels, but MPSVC still performs significantly better than TSVC and SVCRaetsch. Therefore, considering the times by which one algorithm outperforms another one, both USVC and AUSVC have the best performance of classification, while MPSVC has the best performance of restoration. USVC, with AUSVC and MPSVC, generally performs better than SVC and TSVC.

The reverse setting is also introduced to contaminate the original inputs and the results of the average ranks for all four measures are illustrated in Table 6.6. In the comparison

Measure / Algorithm	NMCU	NMEU	MPE	TME
USVC	3.325	1.575	1.250	3.550
SVC	3.250	3.475	3.725	3.400
TSVC	3.375	4.825	4.825	3.975
AUSVC	3.300	2.100	1.900	2.850
MPSVC	2.850	3.025	3.300	3.175
SVCRaetsch	4.900	6.000	6.000	4.050

TABLE 6.6: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the last 30 titanic data sets (from the 71st data set to the 100th data set) contaminated under the general setting, whose ACCL is -1.03% (The average percentage of misclassified inputs in all inputs is 22.33% before contamination and 21.30% after contamination).

of NMCU, SVCRaetsch is significantly the worst algorithm of all. In the comparison of NMEU, TSVC performs significantly better than SVCRaetsch, but significantly worse than the rest algorithms. USVC and AUSVC are significantly better than SVC, USVC is significantly better than MPSVC. In the comparison of MPE, the results are the same to that of the comparison of NMEU except that MPSVC is also significantly worse than AUSVC. In the comparison of TME, the null hypothesis of the Friedman test is accepted, which means that these algorithms perform similarly in the comparison of TME. Indeed, USVC is voted as the best algorithm by the Nemenyi test in the comparison of NMEU and no other algorithms are voted as the best algorithms in the comparison of other classification measures, thus USVC can be deemed as the one that has best performance of classification. And AUSVC can be deemed as the second best performer in classification with considering the times by which it outperforms other algorithms. Though no algorithms are significantly better than each other in the comparison of TME, AUSVC and MPSVC have generally better performance of restoration.

6.4.3 Thyroid Data Sets

Thyroid data sets are five-dimensional examples that can be contaminated to affect the performance of classification and restoration of these experimental algorithms. Every training and test sets contain 140 and 75 observed data. The training kernel is chosen as Gaussian RBF with $\sigma = 3$ and the optimal regularisation parameter is $C = 10$ according

to Rätsch (2001a). After trained with observed inputs being contaminated by noise under Bi and Zhang's setting, the average ranks of these approaches for all four measures are shown in Table 6.7. From the table, we see that the null hypothesis of the Friedman

Measure / Algorithm	NMCU	NMEU	MPE	TME
USVC	4.412	1.738	1.663	5.650
SVC	3.587	4.112	4.213	2.825
TSVC	3.487	5.487	5.350	2.112
AUSVC	3.575	1.875	1.938	4.550
MPSVC	3.100	3.800	3.888	3.112
SVCRaetsch	2.837	3.987	3.950	2.750

TABLE 6.7: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the first 40 thyroid data sets (from the 1st data set to the 40th data set) contaminated under Bi and Zhang's setting, whose ACCL is 24.57% (The average percentage of misclassified inputs in all inputs is 3.43% before contamination and 28.00% after contamination).

test is rejected for all four measures under Bi and Zhang's setting. In the comparison of NMCU, SVCRaetsch and MPSVC are significantly better than USVC. In the comparison of NMEU and MPE, TSVC performs worse than all other algorithms, USVC and AUSVC are significantly better than SVC, SVCRaetsch and MPSVC. Indeed, USVC and AUSVC have been voted as good-performance algorithms by Nemenyi test in the comparison of NMEU and MPE. In the comparison of TME, the performance of USVC and AUSVC is exact the opposite of that in the comparison of NMEU and MPE, USVC is the worst approach in this comparison and AUSVC is significantly better than USVC but significantly worse than rest algorithms, TSVC is significantly better than MPSVC. It shows that USVC and AUSVC can continuously gain advantages in classifying the contaminated inputs, especially, AUSVC can be deemed as the best performer in classification because it has not been outperformed by other algorithms. SVC and TSVC have advantages in recovering the true target function, whilst TSVC can be deemed as the best performer in restoration since it has outperformed other algorithms the most times. MPSVC attempts to achieve a tradeoff between classification and restoration with its resultantly moderate performance in both comparisons under Bi and Zhang's setting.

Under the general setting, the average ranks of these algorithms for all four measures are illustrated in Table 6.8. In the comparison of NMCU, USVC is the algorithm with the worst performance, SVCRaetsch, SVC, TSVC and MPSVC are significantly better than USVC, SVCRaetsch and SVC are significantly better than AUSVC. In the comparison of NMEU, AUSVC, USVC and SVCRaetsch are significantly better than TSVC, while AUSVC and USVC perform significantly better than MPSVC and SVC. In the comparison of MPE, USVC, AUSVC and SVCRaetsch are significantly better than TSVC and MPSVC, USVC and AUSVC are significantly better than SVC. In the comparison of TME, SVCRaetsch, SVC, MPSVC, TSVC are significantly better than USVC and

Algorithm \ Measure	NMCU	NMEU	MPE	TME
USVC	4.841	2.750	2.227	5.250
SVC	2.705	4.068	4.091	2.773
TSVC	3.182	4.659	4.841	2.886
AUSVC	4.295	2.409	2.227	4.500
MPSVC	3.477	4.136	4.455	2.841
SVCRaetsch	2.500	2.977	3.159	2.750

TABLE 6.8: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the middle 30 thyroid data sets (from the 41st data set to the 70th data set) contaminated under the general setting, whose ACCL is 9.97% (The average percentage of misclassified inputs in all inputs is 3.25% before contamination and 13.21% after contamination).

AUSVC. Therefore, slightly better than USVC, AUSVC is the algorithm with the second best performance of classification besides SVCRaetsch. On the other hand, MPSVC can achieve an improved performance of restoration based on AUSVC with affordable loss in classification by inheriting the results from AUSVC.

Table 6.9 lists the average ranks for all four measures which are obtained from test sessions of these algorithms on noiseless test sets and corrupted test sets contaminated by noise under the reverse setting. In the comparison of NMCU, SVCRaetsch,

Algorithm \ Measure	NMCU	NMEU	MPE	TME
USVC	4.591	2.614	2.273	5.136
SVC	3.136	3.977	4.091	2.977
TSVC	2.955	4.432	4.591	2.932
AUSVC	3.932	3.182	2.545	4.250
MPSVC	3.523	4.227	4.545	2.455
SVCRaetsch	2.864	2.568	2.955	3.250

TABLE 6.9: Average ranks of NMCU, NMCU, MPE and TME of different algorithms over the last 30 thyroid data sets (from the 71st data set to the 100th data set) contaminated under the general setting, whose ACCL is 2.92% (The average percentage of misclassified inputs in all inputs is 3.93% before contamination and 6.85% after contamination).

SVC and TSVC perform significantly better than USVC. In the comparison of NMEU, SVCRaetsch and USVC are significantly better than TSVC, MPSVC and SVC. In the comparison of MPE, USVC, AUSVC and SVCRaetsch are significantly better than TSVC and MPSVC, while SVC performs significantly worse than USVC and AUSVC. In the comparison of TME, USVC has the worst performance, MPSVC, TSVC, SVC and SVCRaetsch are significantly better than USVC, while MPSVC, TSVC and SVC perform significantly better than AUSVC. Generally, besides SVCRaetsch, AUSVC has the second best overall performance of classification of all algorithms since it has not been significantly outperformed by any other algorithm. Whilst, USVC is outperformed once,

TSVC, MPSVC and SVC are outperformed twice respectively by other algorithms in the comparison of classification. MPSVC is likely the best performer in the comparison of restoration with slightly better performance than TSVC and SVC.

6.5 Summary

A number of real data sets have been contaminated and applied to statistically compare the performance of classification and restoration of several approaches for the classification subject to input uncertainty. These approaches are all developed based on SVM and basically can be divided into two classes, TSVC and USVC. TSVC implements the farthest points on the uncertainties as the reference in the classification, which provides uncertain inputs with lower probabilities that their corresponding unknown original inputs are going to be correctly classified by the optimal solution of TSVC. USVC exploits the nearest points on the uncertainties as the reference, which conservatively secures higher probabilities of the unknown original inputs being correctly classified. The other approaches introduced in this experiment are iterative algorithms, AUSVC and MPSVC, which are derived from USVC, TSVC and MPM.

In this chapter, A non-parametric statistical comparison, the Friedman test, is first introduced since the experimental environment set in this experiment can not guarantee two assumptions which the parametric statistical test, ANOVA, needs to satisfy. Then four post-hoc tests are used to analyse the result of the Friedman test more specifically. In the experiment, the training and test data sets are contaminated by noise under three settings, Bi and Zhang's setting, the general setting and the reverse setting. Under Bi and Zhang's setting, the inputs being contaminated move towards the other class, this is exactly the same as the preassumption of TSVC, which can effectively let TSVC handle the classification with relatively lower probabilities of the original inputs being correctly classified. The general and reverse settings make the inputs being contaminated move around their original positions without appointed directions and move towards their own class according to the distribution of inputs and the true target function respectively. Some parameters are introduced to control the contamination specifically under different settings.

In the experiment, banana, titanic and thyroid data sets are introduced from Gunnar Rätsch's repositories (Rätsch, 2001a). The contaminated training sets and the noiseless and contaminated test sets are used to evaluate the performance of different algorithms in classifying uncertain inputs and recovering the true target function. After considering the Friedman test and all four post-hoc tests, the average ranks of all four measures, NMCU, NMEU, MPE and TME, for different algorithms are compared with each other statistically. Generally, TSVC has a better performance of restoration under Bi and Zhang's setting and SVC has a better performance of restoration under the general

setting because their optimisation structures can give TSVC and SVC advantages over other algorithms for recovering the true target function under these two settings. USVC and AUSVC generally perform significantly better than all other algorithms in the comparison of NMEU and MPE under all settings, though they have poor performance in NMCU. This is because both USVC and AUSVC conservatively consider not only the mean values of the uncertain inputs but the whole uncertain inputs in the classification. Since the adaptive constraints of AUSVC can not correct the wrongly misclassified inputs with limited misclassified inputs, AUSVC performs even worse than USVC in NMEU and MPE under the general and reverse settings. On the contrary, AUSVC performs better than USVC in NMEU and MPE under Bi and Zhang's setting. With its adaptive constraints, AUSVC also has a better performance in NMCU than USVC by effectively lowering the probabilities of the original inputs being correctly classified. Developed from AUSVC and MPM, MPSVC can significantly improve the performance in NMCU and TME with its optimisation over the probability confidence, which at the same time makes NMEU and MPE worse than those of USVC and AUSVC. In general, MPSVC tries to reach a kind of balance between classification and restoration, though it does not have the best overall performance of classification. Normally, the type of contamination is unknown in classification problems. Therefore, AUSVC is recommended for the purpose of classifying the observed uncertain inputs. For the purpose of recovering the true target from contaminated inputs, MPSVC is recommended. However, if both the performance of classification and the performance of restoration are required for the classification subject to input uncertainty, we will consider MPSVC as the solver.

Chapter 7

Conclusions and Future Work

In machine learning research, incomplete or incorrect information may exist in any aspect of data pre-processing and collection, obscuring the original inputs. As a result, processing the incompletion or handling either incorrect or corrupted observed inputs can introduce uncertainties, which contain the estimated amounts or information by which the observed or calculated inputs may differ from the original inputs. The problems related to these uncertain inputs are increasingly attracting the attention of many machine learning researchers in recent years. Support vector machines (SVMs) as one of the kernel-based maximum margin methods, maximising the margin between two class to hold an upper bound for the generalisation error, have played a key role in the problems arising in data classification and mining. However, as one of the classic applications of SVM, support vector classification (SVC) can only accommodate isotropic uncertain information. This thesis has developed a series of kernel-based algorithms on the base of SVM capable of incorporating the uncertain information with different kernel functions. This concluding chapter summarises the work presented in this thesis and suggests some future directions.

7.1 Summary of Work

The aim of this thesis has been to explore the construction of the classification subject to input uncertainty with data-driven iterative constraints which are statistically and mathematically well-founded and yet have the flexibility to model complex uncertainties. An important component of this has been to provide a kernelisation formulation allowing the impact from the uncertainties to be extended from lower-dimensional input space to higher-dimensional feature space when the classification needs to be extended to non-linear case. This work has been concerned with Gaussian distribution which either can be directly obtained as the prior knowledge of an uncertain input or can be approximated from estimating the missing attributes of an input using the assumption

that the attributes of this inputs follow a joint Gaussian distribution. The distribution of the uncertain inputs can be further set to other prior beliefs in advance to accommodate different assumptions. This thesis compares several applicable SVM-based methods with the developed approaches over the classification subject to input uncertainty, and benchmarks them by several real data sets.

In Chapter 2, in total four noise models that are related to classification problems were introduced besides the traditional additive noise. These noise models can contaminate original inputs and generate corrupted inputs in accordance with the distribution of original inputs and the true target function in classification. Processing the corrupted inputs generated or estimating the missing attributes of original inputs can introduce input uncertainty. Considering the Gaussian process capable of approximating the missing label of an input, the target (output) uncertainty estimation was extended to the input uncertainty estimation for estimating the missing attributes of an input under the assumption that all attributes of this input follow a joint Gaussian distribution. From a statistical model introduced from Bi and Zhang (2005), the relationship between the original inputs and their corrupted counterparts was illustrated and can be used to derive the model of input uncertainty. An input uncertainty model is finally developed, which can be applied to other kinds of contamination to accommodate more complicated noise.

Based on the input uncertainty model obtained, Chapter 3 gave the definition of uncertain inputs and further enabled these uncertain inputs to be incorporated into traditional SVC through geometric interpretation and statistical approach. The resulting optimisation problem is termed as the uncertainty support vector classification (USVC) and involves a second order cone program (SOCP) with a unique solution, from which the dual problem of USVC was derived. USVC was extended to non-linear case through a novel kernelisation formulation, the resulting dual problem can conveniently accommodate different kernel functions. The experiment later in this chapter illustrates the similarity and difference between USVC and SVC under the control of different parameters.

In Chapter 4, several existing approaches were explored in classification subject to input uncertainty. Bi and Zhang (2005) proposed the total support vector classification (TSVC), a SVM-based iterative method, which is indeed an alternative algorithm of the original method directly derived from the statistical models on input uncertainty. TSVC was also extended to accommodate anisotropic uncertainties and non-linear classification by introducing the dual transformation and the new kernelisation formulation. Though TSVC is similar to USVC mathematically and geometrically, both of them have limitations to deal with different contamination situation because they expect different probabilities of the unknown original inputs being correctly classified by the optimal solution. Second order cone programming formulation (SOCPF) proposed by Bhattacharyya et al. (2005) has an equivalent optimisation expression to USVC, though SOCPF is more likely

to be weak duality than USVC because of different dual transformation used, and can perform the same as USVC with appropriately chosen parameters. Minimax probability machine (MPM) was introduced by Lanckriet et al. (2002b) to solve the classification without prior knowledge of the distribution of the inputs. The derivation of MPM also gives an important corollary of the theorem from Bertsimas and Sethuraman (2000), which can transform a probability inequality to a general inequality.

In Chapter 5, motivated by USVC and TSVC, a new iterative algorithm, the adaptive uncertainty support vector classification (AUSVC) was developed by statistically combining TSVC and USVC to adaptively achieve lower probabilities of the original inputs being correctly classified for those uncertain inputs which have been misclassified in the latest iteration. Based on AUSVC, another new method, the minimax probability support vector classification (MPSVC) was developed incorporating MPM to minimise a new measure, the minimax probability error (MPE). MPSVC can amend the possible biased optimal hyperplane of AUSVC caused by the imbalanced distribution of the inputs and can achieve a better performance in recovering the true target function. Both AUSVC and MPSVC incorporate the uncertain information into the optimisation, their resulting algorithms are iterative data-driven optimisation problems with adaptive constraints. Like USVC, AUSVC and MPSVC are able to be extended to non-linear classification by introducing the dual transformation and the kernelisation formulation. Finally, the algorithmic complexities of different algorithms were compared with each other. With the introduction of input uncertainties, the optimisation complexity of the algorithms directly incorporating input uncertainty may be up to the number of attributes cubed times more than that of SVC.

In Chapter 6, a non-parametric statistical test, the Friedman test and in total four post-hoc tests were introduced to evaluate the performance of classification and restoration for different algorithms. A number of real data sets were introduced from Gunnar Rätsch's repositories (Rätsch, 2001a) and contaminated by noise under three different settings: Bi and Zhang's setting, the general setting and the reverse setting. These algorithms were trained and tested with the contaminated training sets and the noiseless and contaminated test sets, then the obtained average ranks of all four measures, NMCU, NMEU, MPE and TME were compared with each other statistically in the Friedman test and post-hoc tests. Generally, TSVC and SVC perform better than other algorithms in recovering the true target function under Bi and Zhang's setting and the general setting respectively. While MPSVC performs better than other algorithms in recovering the true target function under the reverse setting and it has the best overall performance of restoration under all three settings. In the comparison of NMEU and MPE, TSVC and SVC can not perform as well as in the comparison of NMCU because TSVC and SVC choose the farthest points and the central points respectively in the uncertainties as reference to classify the inputs, leading relatively low probabilities of the original inputs being correctly classified for TSVC and SVC. AUSVC generally can improve the

performance of classification than USVC with its adaptive constraints. Combining the advantages from AUSVC in classification and MPM in restoration makes MPSVC reach a balance between classifying the contaminated inputs and recovering the true target function. But all in all, the performance of classification in MPSVC is not comparable to that of AUSVC. Therefore, AUSVC is considered when classifying the observed uncertain inputs is required, MPSVC is considered for the purpose of recovering the true target function from contaminated inputs.

7.2 Future Work

The work of this thesis was originally inspired by the idea of discriminating the training data being contaminated by noise, and then was extended to the restoration of the true target function from the corrupted training data. Meanwhile, some progress has been made in describing the theoretical and practical aspects of the SVM-based classification subject to input uncertainty. In this section some aspects of the current work which could form the basis for further investigation are discussed.

7.2.1 Other Prior Assumptions

This thesis mainly focused on the Gaussian distribution which is defined as the prior knowledge of the uncertain inputs and can be obtained from estimating the missing attributes under the joint-Gaussian assumption. However, the introduction of MPM has provided an idea to extend the prior assumption of the uncertain inputs. Equation (3.5) can be used to incorporate other prior distribution assumptions. Specifically, formula (5.8) can be introduced to handle the uncertain inputs without knowing their individual distributions. Generally, this can lead to further applications in the future. For example, the uncertain inputs $\boldsymbol{z}_i$ and their distributions are unknown, but its samples $\boldsymbol{z}_i = \{\boldsymbol{z}_{it} : t \in T\}$ in time series are available instead. Then these time series samples can be used to compute the mean and the covariance of $\boldsymbol{z}_i$, which can be incorporated into the classification subject to input uncertainty.

7.2.2 Large Scale Implementations

In this thesis, all the approaches have used the standard QP and SOCP optimiser to estimate complexities of these algorithms. The results show that the methods incorporating the uncertainties are expensive in memory requirement and computational cost. Nevertheless, some methods have been proposed to exploit sparsity and structure in solving the SVM problems. Joachims (1998) presented an algorithm based on a decomposition strategy and effectively selecting the variables. Platt (1998) not only introduced decomposition strategy in its developed algorithm, sequential minimal optimisation (SMO),

but also solved the small QP problems, obtained from the decomposition, analytically to avoid a time-consuming inner loop. The memory requirement of SMO is linear $O(l)$ and the computation cost is somewhere between linear and quadratic time. And an improved SMO method was proposed in Keerthi et al. (1999). In order to make the techniques developed here more widely applicable, further work is warranted to find more efficient algorithms exploiting the structure of the input uncertainty problem.

Appendix A

General Derivations from Primal Problems to Dual Problems

The following derivations are used extensively in USVC and AUSVC to obtain the dual problems from the original primal problems. In the derivations, we initially exploit a general primal problem, which can be transformed to the specific primal problems of USVC and AUSVC by setting parameter r. The general primal problem is shown as follows:

$$\begin{aligned} \min\ & t + C\sum_{i=1}^{l}\xi_i \\ s.t.\ & r\sigma_i \le y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \\ & \|\mathbf{M}_i^{1/2}\boldsymbol{w}\| \le \sigma_i \\ & \left\| \begin{bmatrix} \boldsymbol{w} \\ \frac{t-1}{\sqrt{2}} \end{bmatrix} \right\| \le \frac{t+1}{\sqrt{2}} \\ & \xi_i \ge 0 \qquad i = 1, \ldots, l. \end{aligned} \tag{A.1}$$

Letting $\boldsymbol{w}_1 = \left[t \,\vdots\, \boldsymbol{w}^T \,\vdots\, b \,\vdots\, \sigma_1 \ldots \sigma_l \,\vdots\, \xi_1 \ldots \xi_l \right]^T \in \mathbb{R}^{n+2l+2}$, we obtain

$$\begin{aligned} \min\ & \boldsymbol{f}^T\boldsymbol{w}_1 \\ s.t.\ & \|\mathbf{A}_i\boldsymbol{w}_1 + \boldsymbol{b}_i\| \le \boldsymbol{c}_i^T\boldsymbol{w}_1 + d_i, \qquad i = 1, \ldots, 3l+1, \end{aligned} \tag{A.2}$$

and the parameters in (A.2) are listed as follows:

$$\boldsymbol{f} = \left[\,1\,\middle|\,\mathbf{0}^T\,\middle|\,0\,\middle|\,\mathbf{0}^T\,\middle|\,\boldsymbol{C}^T\,\right]^T \qquad i = 1,\ldots,l$$

$$\mathbf{A}_i = \left[\begin{array}{c|c|c|c|c} 0 & & 0 & & \\ \vdots & \mathbf{0} & \vdots & \mathbf{0} & \mathbf{0} \\ 0 & & 0 & & \end{array}\right] \qquad \boldsymbol{b}_i = \mathbf{0}$$

$$\boldsymbol{c}_i = \left[\,0\,\middle|\,y_i\boldsymbol{x}_i^T\,\middle|\,y_i\,\middle|\,0\ldots-r\ldots0\,\middle|\,0\ldots1\ldots0\,\right]^T \qquad d_i = -1$$

$$\mathbf{A}_{l+i} = \left[\begin{array}{c|c|c|c|c} 0 & & 0 & & \\ \vdots & \mathbf{M}_i^{1/2} & \vdots & \mathbf{0} & \mathbf{0} \\ 0 & & 0 & & \end{array}\right] \qquad \boldsymbol{b}_{l+i} = \mathbf{0}$$

$$\boldsymbol{c}_{l+i} = \left[\,0\,\middle|\,\mathbf{0}^T\,\middle|\,0\,\middle|\,0\ldots1\ldots0\,\middle|\,\mathbf{0}^T\,\right]^T \qquad d_{l+i} = 0$$

$$\mathbf{A}_{2l+1} = \left[\begin{array}{c|c|c|c|c} \frac{1}{\sqrt{2}} & \mathbf{0}^T & 0 & 0 & 0 \\ \hline 0 & & 0 & 0 & 0 \\ \vdots & \mathbf{I} & \vdots & \vdots & \vdots \\ 0 & & 0 & 0 & 0 \end{array}\right] \qquad \boldsymbol{b}_{2l+1} = \left[\begin{array}{c} -\frac{1}{\sqrt{2}} \\ \hline \mathbf{0} \end{array}\right]$$

$$\boldsymbol{c}_{2l+1} = \left[\,\tfrac{1}{\sqrt{2}}\,\middle|\,\mathbf{0}^T\,\middle|\,0\,\middle|\,\mathbf{0}^T\,\middle|\,\mathbf{0}^T\,\right] \qquad d_{2l+1} = \frac{1}{\sqrt{2}}$$

$$\mathbf{A}_{2l+1+i} = \left[\begin{array}{c|c|c|c|c} 0 & & 0 & & \\ \vdots & \mathbf{0} & \vdots & \mathbf{0} & \mathbf{0} \\ 0 & & 0 & & \end{array}\right] \qquad \boldsymbol{b}_{2l+1+i} = \mathbf{0}$$

$$\boldsymbol{c}_{2l+1+i} = \left[\,0\,\middle|\,\mathbf{0}^T\,\middle|\,0\,\middle|\,\mathbf{0}^T\,\middle|\,0\ldots1\ldots0\,\right]^T \qquad d_{2l+1+i} = 0.$$

Reintroducing these parameters into (3.12) and setting $\boldsymbol{\beta}_{2l+1} = \left[\,\beta'\,\middle|\,\boldsymbol{\beta}'_{2l+1}\,\right]^T$, the (3.12) can be rewritten as:

$$\max\ \frac{1}{\sqrt{2}}\beta' + \sum_{1=1}^{l}\alpha_i - \frac{1}{\sqrt{2}}\alpha_{2l+1}$$

$$s.t.\quad \left[\begin{array}{c} 0 \\ \hline \vdots \\ \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta}_{l+i} \\ \vdots \\ \hline 0 \\ \hline \vdots \\ 0 \\ \vdots \\ \hline \vdots \\ 0 \\ \vdots \end{array}\right] + \left[\begin{array}{c} 0 \\ \hline \vdots \\ \sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i \\ \vdots \\ \hline \sum_{i=1}^{l}\alpha_i y_i \\ \hline \vdots \\ -r\alpha_i \\ \vdots \\ \hline \vdots \\ \alpha_i \\ \vdots \end{array}\right] + \left[\begin{array}{c} \frac{1}{\sqrt{2}}\beta' \\ \hline \vdots \\ \boldsymbol{\beta}'_{2l+1} \\ \vdots \\ \hline 0 \\ \hline \vdots \\ 0 \\ \vdots \\ \hline \vdots \\ 0 \\ \vdots \end{array}\right] + \left[\begin{array}{c} \frac{1}{\sqrt{2}}\alpha_{2l+1} \\ \hline \vdots \\ 0 \\ \vdots \\ \hline 0 \\ \hline \vdots \\ \alpha_{l+i} \\ \vdots \\ \hline \vdots \\ \alpha_{2l+1+i} \\ \vdots \end{array}\right] = \left[\begin{array}{c} 1 \\ \hline \vdots \\ 0 \\ \vdots \\ \hline 0 \\ \hline \vdots \\ 0 \\ \vdots \\ \hline \vdots \\ C \\ \vdots \end{array}\right]. \tag{A.3}$$

So we have

$$\begin{aligned}
\max \quad & \frac{1}{\sqrt{2}}\beta' + \sum_{1=1}^{l}\alpha_i - \frac{1}{\sqrt{2}}\alpha_{2l+1} \\
s.t. \quad & \frac{1}{\sqrt{2}}\beta^{'} + \frac{1}{\sqrt{2}}\alpha_{2l+1} = 1. && \text{(A.4)} \\
& \sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta}_{l+i} + \boldsymbol{\beta}'_{2l+1} = \mathbf{0}. && \text{(A.5)} \\
& \sum_{i=1}^{l}\alpha_i y_i = 0. && \text{(A.6)} \\
& \alpha_{l+i} - r\alpha_i = 0 \qquad i = 1,\ldots,l. && \text{(A.7)} \\
& \alpha_i + \alpha_{2l+1+i} = C \qquad i = 1,\ldots,l. && \text{(A.8)} \\
& \|\boldsymbol{\beta}_{l+i}\| \le \alpha_{l+i} \qquad i = 1,\ldots,l. && \text{(A.9)}
\end{aligned}$$

From the standard dual problem of SOCP, we have

$$\boldsymbol{\beta}_{l+i} = -\alpha_{l+i}\frac{\mathbf{M}_i^{1/2}\boldsymbol{w}}{\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|}. \tag{A.10}$$

(A.10) is introduced back into (A.5) to reformulate the constraint (A.5) as:

$$\sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i - \sum_{i=1}^{l}\alpha_{l+i}\frac{\mathbf{M}_i^{1/2}\boldsymbol{w}}{\|\mathbf{M}_i^{1/2}\boldsymbol{w}\|} + \boldsymbol{\beta}'_{2l+1} = \mathbf{0}. \tag{A.11}$$

$\boldsymbol{w}^T \times$ (A.11) on both sides, we have

$$\sum_{i=1}^{l}\alpha_i y_i \boldsymbol{w}^T\boldsymbol{x}_i - \sum_{i=1}^{l}\alpha_{l+i}\|\mathbf{M}_i^{1/2}\boldsymbol{w}\| + \boldsymbol{w}^T\boldsymbol{\beta}'_{2l+1} = 0. \tag{A.12}$$

Introducing the constraints of (A.1) and (A.7) into (A.12), we have

$$\sum_{i=1}^{l}\alpha_i \le -\boldsymbol{w}^T\boldsymbol{\beta}'_{2l+1}. \tag{A.13}$$

Like (A.10), $\boldsymbol{\beta}_{2l+1}$ can be transformed as:

$$\boldsymbol{\beta}_{2l+1} = -\alpha_{2l+1}\frac{\begin{pmatrix} \boldsymbol{w} \\ \frac{t-1}{\sqrt{2}} \end{pmatrix}}{\left\|\begin{pmatrix} \boldsymbol{w} \\ \frac{t-1}{\sqrt{2}} \end{pmatrix}\right\|}. \tag{A.14}$$

According to the definition of $\boldsymbol{\beta}_{2l+1}$, we have

$$\sqrt{\|\boldsymbol{w}\|^2 + \frac{(t-1)^2}{2}}\beta' = -\alpha_{2l+1}\frac{t-1}{\sqrt{2}}. \tag{A.15}$$

$$\sqrt{\|\boldsymbol{w}\|^2 + \frac{(t-1)^2}{2}}\boldsymbol{\beta}'_{2l+1} = -\alpha_{2l+1}\boldsymbol{w}. \tag{A.16}$$

$$\frac{(t-1)^2}{2}(\alpha_{2l+1}^2 - \beta'^2) = \beta'^2\|\boldsymbol{w}\|^2. \tag{A.17}$$

(A.15)/(A.16), we have $\frac{\sqrt{2}}{t-1}\beta'\|\boldsymbol{w}\| = \|\boldsymbol{\beta}'_{2l+1}\|$, thus,

$$\alpha_{2l+1}^2 = \beta'^2 + \|\boldsymbol{\beta}'_{2l+1}\|^2 = \frac{2\beta'^2\|\boldsymbol{w}\|^2 + (t-1)^2\beta'^2}{(t-1)^2} \leq \frac{(t+1)^2}{(t-1)^2}\beta'^2. \tag{A.18}$$

Introducing (A.4) into (A.18), we have

$$\alpha_{2l+1}^2 \leq \frac{(t+1)^2}{(t-1)^2}(\sqrt{2} - \alpha_{2l+1})^2. \tag{A.19}$$

Solving (A.19), we obtain

$$\alpha_{2l+1} \geq \frac{t+1}{\sqrt{2}}. \tag{A.20}$$

Introducing (A.4) into the objective function, we have $\max \sum_{i=1}^{l}\alpha_i + 1 - \sqrt{2}\alpha_{2l+1}$. According to the characteristics of lagrangian dual problem, we have

$$\alpha_i + 1 - \sqrt{2}\alpha_{2l+1} \leq t. \tag{A.21}$$

Combining (A.20) and (A.21), we have $\sum_i^l \alpha_i \leq 2t$ and this inequality turns to an equality when the optimal solution is achieved. Recalling (A.13), we have

$$\boldsymbol{\beta}'_{2l+1} = -\boldsymbol{w}. \tag{A.22}$$

Reintroducing (A.22) into (A.5), we have

$$\boldsymbol{w} = \sum_{i=1}^{l}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{l}(\mathbf{M}_i^{1/2})^T\boldsymbol{\beta}_{l+i}. \tag{A.23}$$

Introducing (A.20) back into the objective function, we obtain the dual problem as follows:

$$\begin{aligned} \max \ & \sum_{i=1}^{l} \alpha_i - \frac{\|\boldsymbol{w}\|^2}{2} \\ s.t. \ & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & \|\boldsymbol{\beta}_{l+i}\| \leq \alpha_{l+i} = r\alpha_i \\ & 0 \leq \alpha_i \leq C. \end{aligned} \tag{A.24}$$

Appendix B

Derivations of Kernelising TSVC in the General Case

The following derivations are implemented to obtain $\triangle \boldsymbol{x}_i$ and ξ_i of the iterative algorithm of TSVC based on Definition 3.1. According to (4.2), (4.4) and the previous result of (3.2), $\triangle \boldsymbol{x}_i$ in linear case can be derived by following Lemma 4.1,

$$\triangle \boldsymbol{x}_i = y_i (\mathbf{M}_i^{1/2})^T \frac{\mathbf{M}_i^{1/2} \boldsymbol{w}}{\|\mathbf{M}_i^{1/2} \boldsymbol{w}\|}. \tag{B.1}$$

Therefore, we have

$$y_i(\boldsymbol{w}^T(\boldsymbol{x}_i + \triangle \boldsymbol{x}_i) + b) = y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) + y_i \triangle \boldsymbol{x}_i^T \boldsymbol{w} = y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) + \|\mathbf{M}_i^{1/2} \boldsymbol{w}\|. \tag{B.2}$$

Thus,

$$\begin{aligned} \xi_i &= \max\left\{0, 1 - \left[y_i(\boldsymbol{w}^T(\boldsymbol{x}_i + \triangle \boldsymbol{x}_i) + b)\right]\right\} \\ &= \max\left\{0, 1 - \left[y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) + \|\mathbf{M}_i^{1/2} \boldsymbol{w}\|\right]\right\}. \end{aligned} \tag{B.3}$$

In the non-linear case, $\phi(\mathbf{M}_i^{1/2}) = \mathbf{M}_i^{1/2} \mathbf{J}^T$ and $\boldsymbol{w} = \sum_j \alpha_j y_j \phi(\boldsymbol{x}_j)$ are introduced to kernelise TSVC, where ϕ is the function mapping the data of input space to feature space and $\mathbf{J}$ is Jacobian matrix. Following the derivations of TSVC in the linear case, we have

$$\triangle \boldsymbol{x}_i = y_i (\mathbf{M}_i^{1/2})^T \frac{\phi(\mathbf{M}_i^{1/2}) \boldsymbol{w}}{\|\phi(\mathbf{M}_i^{1/2}) \boldsymbol{w}\|} = y_i (\mathbf{M}_i^{1/2})^T \frac{\mathbf{M}_i^{1/2} \sum_j \alpha_j y_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i}}{\|\mathbf{M}_i^{1/2} \sum_j \alpha_j y_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle \boldsymbol{x}_j)}{\partial \boldsymbol{x}_i}\|}, \tag{B.4}$$

where $\mathbf{J}^T\boldsymbol{w} = \sum_j \alpha_j y_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j)}{\partial \boldsymbol{x}_i}$ and $K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j + \triangle\boldsymbol{x}_j)$. Implementing Taylor expansion, we have

$$\begin{aligned}
& y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i + \triangle\boldsymbol{x}_i) + b) \\
&= y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) + y_i\left[\frac{\partial\phi(\boldsymbol{x}_i)}{\partial\boldsymbol{x}_i}\triangle\boldsymbol{x}_i\right]^T\boldsymbol{w} + O\left(\frac{1}{2}\left[\frac{\partial^2\phi(\boldsymbol{x}_i)}{\partial\boldsymbol{x}_i^2}\triangle\boldsymbol{x}_i\right]^T\boldsymbol{w} + \cdots\right) \\
&\simeq y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) + y_i\triangle\boldsymbol{x}_i^T\left[\frac{\partial\phi(\boldsymbol{x}_i)}{\partial\boldsymbol{x}_i}\right]^T\boldsymbol{w} \\
&= y_i\left(\sum_j \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j) + b\right) + y_i\left(y_i\frac{\left[\phi(\mathbf{M}_i^{1/2})\boldsymbol{w}\right]^T}{\|\phi(\mathbf{M}_i^{1/2})\boldsymbol{w}\|}\mathbf{M}_i^{1/2}\right)\mathbf{J}^T\boldsymbol{w} \\
&= y_i\left(\sum_j \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j) + b\right) + \left\|\mathbf{M}_i^{1/2}\sum_j \alpha_j y_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j)}{\partial\boldsymbol{x}_i}\right\|.
\end{aligned} \tag{B.5}$$

So

$$\begin{aligned}
\xi_i = \max\Bigg\{0, 1 - \Bigg[& y_i\left(\sum_j \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j) + b\right) \\
& + \left\|\mathbf{M}_i^{1/2}\sum_j \alpha_j y_j \frac{\partial K(\boldsymbol{x}_i, \boldsymbol{x}_j + \triangle\boldsymbol{x}_j)}{\partial\boldsymbol{x}_i}\right\|\Bigg]\Bigg\}.
\end{aligned} \tag{B.6}$$

Appendix C

Demonstration of the Friedman Test and Post-Hoc Tests

To show how statistical analysis works, some examples are demonstrated to show the analysis outputs from the Friedman test and all four post-hoc tests. The following example is the statistical comparison of the average ranks of NMEU of different algorithms on titanic data sets contaminated by noise under the general setting. This matlab function name is "FriedmantestRatschData", in which "titanic data sets" and "the general setting" are the first two parameters, the third and fourth parameters are τ and ν, which are set as 0.8 and 0.6 respectively. The fifth and sixth parameters are the serial numbers of the first and the last of a group of data sets used. The last parameter is the significance equal to 0.1 used in the Friedman test. While both 0.05 and 0.1 are used as the significance for post-hoc tests.

```
>> FriedmantestRatschData('titanic','general',0.8,0.6,41,70,6,0.1)
The kernel is RBF.
Sigma = 2.
********************************************************************************
ANALYSING MISCLASSIFIED EDGE OBSERVED DATA NUMBER PERCENTAGE IN TEST.
********************************************************************************
SVCRatsch = 5.900
SVC = 3.367
USVC = 1.917
TSVC = 4.950
AUSVC = 1.933
MPSVC = 2.933
********************************************************************************
The Friedman test begins:
Reject the null hypothesis under Chi-square distribution, these algorithms are different
Reject the null hypothesis under F distribution, these algorithms are different
********************************************************************************
Post-hoc Nemenyi test begins:
Nemenyi test at 0.05: CD = 1.377
Nemenyi test at 0.10: CD = 1.250
The difference between:
```

```
SVCRatsch&SVC = 2.533
SVCRatsch&USVC = 3.983
SVCRatsch&TSVC = 0.950
SVCRatsch&AUSVC = 3.967
SVCRatsch&MPSVC = 2.967
SVC&USVC = 1.450
SVC&TSVC = 1.583
SVC&AUSVC = 1.433
SVC&MPSVC = 0.433
USVC&TSVC = 3.033
USVC&AUSVC = 0.017
USVC&MPSVC = 1.017
TSVC&AUSVC = 3.017
TSVC&MPSVC = 2.017
AUSVC&MPSVC = 1.000
The difference between the best and the worst performing algorithm is already larger than CD
in the Nemenyi test at 0.05
SVCRatsch,TSVC, is significantly worse than the other algorithms in the Nemenyi test at 0.05
Not enough algorithms vote for the best algorithm in the Nemenyi test at 0.05
The difference between the best and the worst performing algorithm is already larger than CD
in the Nemenyi test at 0.10
SVCRatsch,TSVC, is significantly worse than the other algorithms in the Nemenyi test at 0.10
Not enough algorithms vote for the best algorithm in the Nemenyi test at 0.10
Post-hoc Nemenyi test ended.
**************************************************************************
**************************************************************************
Post-hoc Bonferroni-Dunn test begins:
Bonferroni-Dunn test at 0.05: CD = 1.244
Bonferroni-Dunn test at 0.10: CD = 1.124
The difference between:
SVCRatsch&SVC = 2.533
SVCRatsch&USVC = 3.983
SVCRatsch&TSVC = 0.950
SVCRatsch&AUSVC = 3.967
SVCRatsch&MPSVC = 2.967
SVCRatsch has the worst performance of all algorithms
**************************
More details of the Bonferroni-Dunn test at 0.05
SVC,USVC,AUSVC,MPSVC, performs significantly better than SVCRatsch in the Bonferroni-Dunn
test at 0.05
TSVC, does not perform significantly better than SVCRatsch in the Bonferroni-Dunn test at 0.05
The further test is applied to TSVC,  in the Bonferroni-Dunn test at 0.05
TSVC&SVC = 1.583
TSVC&USVC = 3.033
TSVC&AUSVC = 3.017
TSVC&MPSVC = 2.017
SVC,USVC,AUSVC,MPSVC, performs significantly better than TSVC in the Bonferroni-Dunn test at 0.05
**************************
More details of the Bonferroni-Dunn test at 0.10
SVC,USVC,AUSVC,MPSVC, performs significantly better than SVCRatsch in the Bonferroni-Dunn
test at 0.10
TSVC, does not perform significantly better than SVCRatsch in the Bonferroni-Dunn test at 0.10
The further test is applied to TSVC,  in the Bonferroni-Dunn test at 0.10
TSVC&SVC = 1.583
TSVC&USVC = 3.033
TSVC&AUSVC = 3.017
TSVC&MPSVC = 2.017
SVC,USVC,AUSVC,MPSVC, performs significantly better than TSVC in the Bonferroni-Dunn test at 0.10
Post-hoc Bonferroni-Dunn test ended.
```

```
**************************************************************************
**************************************************************************
Post-hoc Holm test begins:
More details of the Holm test at 0.05
Compare the other algorithms with SVCRatsch in Holm test at 0.05
Algorithm=USVC   i=1   p=0.000  threshold=0.010
Algorithm=AUSVC   i=2   p=0.000  threshold=0.013
Algorithm=MPSVC   i=3   p=0.000  threshold=0.017
Algorithm=SVC   i=4   p=0.000  threshold=0.025
Algorithm=TSVC   i=5   p=0.049  threshold=0.050
USVC,AUSVC,MPSVC,SVC,TSVC, are significantly better than SVCRatsch in Holm test at 0.05
Compare the other algorithms with TSVC in Holm test at 0.05
Algorithm=USVC   i=1   p=0.000  threshold=0.013
Algorithm=AUSVC   i=2   p=0.000  threshold=0.017
Algorithm=MPSVC   i=3   p=0.000  threshold=0.025
Algorithm=SVC   i=4   p=0.001  threshold=0.050
USVC,AUSVC,MPSVC,SVC, are significantly better than TSVC in Holm test at 0.05
Compare the other algorithms with SVC in Holm test at 0.05
Algorithm=USVC   i=1   p=0.003  threshold=0.017
Algorithm=AUSVC   i=2   p=0.003  threshold=0.025
Algorithm=MPSVC   i=3   p=0.370  threshold=0.050
USVC,AUSVC, are significantly better than SVC in Holm test at 0.05
Compare the other algorithms with MPSVC in Holm test at 0.05
Algorithm=USVC   i=1   p=0.035  threshold=0.025
Algorithm=AUSVC   i=2   p=0.038  threshold=0.050
AUSVC, are significantly better than MPSVC in Holm test at 0.05
Compare the other algorithms with AUSVC in Holm test at 0.05
Algorithm=USVC   i=1   p=0.972  threshold=0.050
**************************
More details of the Holm test at 0.10
Compare the other algorithms with SVCRatsch in Holm test at 0.10
Algorithm=USVC   i=1   p=0.000  threshold=0.020
Algorithm=AUSVC   i=2   p=0.000  threshold=0.025
Algorithm=MPSVC   i=3   p=0.000  threshold=0.033
Algorithm=SVC   i=4   p=0.000  threshold=0.050
Algorithm=TSVC   i=5   p=0.049  threshold=0.100
USVC,AUSVC,MPSVC,SVC,TSVC, are significantly better than SVCRatsch in Holm test at 0.10
Compare the other algorithms with TSVC in Holm test at 0.10
Algorithm=USVC   i=1   p=0.000  threshold=0.025
Algorithm=AUSVC   i=2   p=0.000  threshold=0.033
Algorithm=MPSVC   i=3   p=0.000  threshold=0.050
Algorithm=SVC   i=4   p=0.001  threshold=0.100
USVC,AUSVC,MPSVC,SVC, are significantly better than TSVC in Holm test at 0.10
Compare the other algorithms with SVC in Holm test at 0.10
Algorithm=USVC   i=1   p=0.003  threshold=0.033
Algorithm=AUSVC   i=2   p=0.003  threshold=0.050
Algorithm=MPSVC   i=3   p=0.370  threshold=0.100
USVC,AUSVC, are significantly better than SVC in Holm test at 0.10
Compare the other algorithms with MPSVC in Holm test at 0.10
Algorithm=USVC   i=1   p=0.035  threshold=0.050
Algorithm=AUSVC   i=2   p=0.038  threshold=0.100
USVC,AUSVC, are significantly better than MPSVC in Holm test at 0.10
Compare the other algorithms with AUSVC in Holm test at 0.10
Algorithm=USVC   i=1   p=0.972  threshold=0.100
Post-hoc Holm test ended.
**************************************************************************
**************************************************************************
Post-hoc Hommel test begins:
More details of the Hommel test at 0.05
```

```
Compare the other algorithms with SVCRatsch in Hommel test at 0.05
Algorithm=USVC   i=1   p=0.000  threshold=0.010  tighterthreshold=0.010
Algorithm=AUSVC   i=2   p=0.000  threshold=0.020  tighterthreshold=0.010
Algorithm=MPSVC   i=3   p=0.000  threshold=0.030  tighterthreshold=0.010
Algorithm=SVC   i=4   p=0.000  threshold=0.040  tighterthreshold=0.010
Algorithm=TSVC   i=5   p=0.049  threshold=0.050  tighterthreshold=0.010
USVC,AUSVC,MPSVC,SVC are significantly better than SVCRatsch in Hommel test at 0.05
Compare the other algorithms with TSVC in Hommel test at 0.05
Algorithm=USVC   i=1   p=0.000  threshold=0.013  tighterthreshold=0.013
Algorithm=AUSVC   i=2   p=0.000  threshold=0.025  tighterthreshold=0.013
Algorithm=MPSVC   i=3   p=0.000  threshold=0.038  tighterthreshold=0.013
Algorithm=SVC   i=4   p=0.001  threshold=0.050  tighterthreshold=0.013
USVC,AUSVC,MPSVC,SVC are significantly better than TSVC in Hommel test at 0.05
Compare the other algorithms with SVC in Hommel test at 0.05
Algorithm=USVC   i=1   p=0.003  threshold=0.017  tighterthreshold=0.017
Algorithm=AUSVC   i=2   p=0.003  threshold=0.033  tighterthreshold=0.017
Algorithm=MPSVC   i=3   p=0.370  threshold=0.050  tighterthreshold=0.017
MPSVC are not significantly better than SVC in Hommel test at 0.05
USVC,AUSVC are significantly better than SVC in Hommel test at 0.05
Compare the other algorithms with MPSVC in Hommel test at 0.05
Algorithm=USVC   i=1   p=0.035  threshold=0.025  tighterthreshold=0.025
USVC,AUSVC are not significantly better than MPSVC in Hommel test at 0.05
Compare the other algorithms with AUSVC in Hommel test at 0.05
Algorithm=USVC   i=1   p=0.972  threshold=0.050  tighterthreshold=0.050
USVC are not significantly better than AUSVC in Hommel test at 0.05
**************************
More details of the Hommel test at 0.10
Compare the other algorithms with SVCRatsch in Hommel test at 0.10
Algorithm=USVC   i=1   p=0.000  threshold=0.020  tighterthreshold=0.020
Algorithm=AUSVC   i=2   p=0.000  threshold=0.040  tighterthreshold=0.020
Algorithm=MPSVC   i=3   p=0.000  threshold=0.060  tighterthreshold=0.020
Algorithm=SVC   i=4   p=0.000  threshold=0.080  tighterthreshold=0.020
Algorithm=TSVC   i=5   p=0.049  threshold=0.100  tighterthreshold=0.020
USVC,AUSVC,MPSVC,SVC are significantly better than SVCRatsch in Hommel test at 0.10
Compare the other algorithms with TSVC in Hommel test at 0.10
Algorithm=USVC   i=1   p=0.000  threshold=0.025  tighterthreshold=0.025
Algorithm=AUSVC   i=2   p=0.000  threshold=0.050  tighterthreshold=0.025
Algorithm=MPSVC   i=3   p=0.000  threshold=0.075  tighterthreshold=0.025
Algorithm=SVC   i=4   p=0.001  threshold=0.100  tighterthreshold=0.025
USVC,AUSVC,MPSVC,SVC are significantly better than TSVC in Hommel test at 0.10
Compare the other algorithms with SVC in Hommel test at 0.10
Algorithm=USVC   i=1   p=0.003  threshold=0.033  tighterthreshold=0.033
Algorithm=AUSVC   i=2   p=0.003  threshold=0.067  tighterthreshold=0.033
Algorithm=MPSVC   i=3   p=0.370  threshold=0.100  tighterthreshold=0.033
MPSVC are not significantly better than SVC in Hommel test at 0.10
USVC,AUSVC are significantly better than SVC in Hommel test at 0.10
Compare the other algorithms with MPSVC in Hommel test at 0.10
Algorithm=USVC   i=1   p=0.035  threshold=0.050  tighterthreshold=0.050
Algorithm=AUSVC   i=2   p=0.038  threshold=0.100  tighterthreshold=0.050
USVC,AUSVC are significantly better than MPSVC in Hommel test at 0.10
Compare the other algorithms with AUSVC in Hommel test at 0.10
Algorithm=USVC   i=1   p=0.972  threshold=0.100  tighterthreshold=0.100
USVC are not significantly better than AUSVC in Hommel test at 0.10
Post-hoc Hommel test ended.
**************************************************************************
```

Another example is the statistical comparison of the average ranks of TME of different algorithms on titanic data sets contaminated by noise under the reverse setting, in

which the hypothesis is accepted so that these algorithms have similar performance of restoration.

```
>> FriedmantestRatschData('titanic','reverse',0.8,0.6,71,100,4,0.1)
The kernel is RBF.
Sigma = 2.
********************************************************************************
ANALYSING GENERALISATION ERROR.
********************************************************************************
SVCRatsch = 4.050
SVC = 3.400
USVC = 3.550
TSVC = 3.975
AUSVC = 2.850
MPSVC = 3.175
********************************************************************************
The Friedman test begins:
Accept the null hypothesis under Chi-square distribution, these algorithms are similar
Accept the null hypothesis under F distribution, these algorithms are similar
```

Bibliography

F. Aires, C. Prigent, and W.B. Rossow. Neural network uncertainty assessment using bayesian statistics with application to remote sensing: 2. output errors. *Journal of Geographical Research*, 109(D10304), 2004.

A. Aizermann, E.M. Braverman, and L.I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Cntrol*, 25:821–837, 1964.

D.N. Anderson. A multivariate linnik distribution. *Statistics & Probability Letters*, 14 (4):333–336, 1992.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

D. Bertsimas and J. Sethuraman. Moment problems and semidefinite optimization. In *Handbook of Semidefinite Optimization*, pages 469–509. Kluwer Academic Publishers, 2000.

C. Bhattacharyya. Second order cone programming formulations for feature selection. *Journal of Machine Learning Research*, 5:1417–1433, 2004.

C. Bhattacharyya, K.S. Pannagadatta, and A.J. Smola. A second order cone programming formulation for classifying missing data. In *Advances in Neural Information Processing Systems*, volume 17, pages 153–160. MIT Press, 2005.

J. Bi and V. Vapnik. Learning with rigorous support vector machines. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop (COLT/Kernel 2003)*, volume 2777 of *Lecture Notes in Computer Science*, pages 231–242, Washington DC, USA, August 2003. Springer.

J. Bi and T. Zhang. Support vector classification with input data uncertainty. In *Advances in Neural Information Processing Systems*, volume 17, pages 161–168. Cambridge, MA: MIT Press, 2005.

V. Blanz, B. Schölkopf, H. Bülthoff, C.J.C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3d models. In

Proceedings of International Conference on Artificial Neural Networks, volume 1112 of *Lecture Notes in Computer Science*, pages 251–256, Berlin, 1996. Springer.

B.E. Boser, I.M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, Pennsylvania, USA, 1992. ACM.

S. Boughorbel, J.P. Tarel, and F. Fleuret. Non-mercer kernels for SVM object recognition. In *Proceedings of British Machine Vision Conference*, pages 137–146, London, England, September 2004.

C.A. Bouman. Cluster: An unsupervised algorithm for modeling gaussian mixtures. Technical report, School of Electrical and Computer Engineering, Purdue University, `https://engineering.purdue.edu/~bouman/software/cluster/`, April 1997.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 1st edition, 2004.

N. Bshouty, N. Eiron, and E. Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002.

F. Canters, W.D. Genst, and H. Dufourmont. Assessing effects of input uncertainty in structural landscape classification. *International Journal of Geographical Information Science*, 16(2):129–149, 2002.

S.E. Chick. Bayesian analysis for simulation input and output. In *Proceedings of the 1997 Winter Simulation Conference*, pages 253–260, Atlanta, Georgia, USA, 1997.

S.E. Chick. Input distribution selection for simulation experiments: Accounting for input uncertainty. *Operation Research*, 49:744–758, 2001.

M. Clyde and E.I. George. Model uncertainty. *Statistical Science*, 19(1):81–94, 2004.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

A.C. Cullen and H.C. Frey. *Probabilistic Techniques in Exposure Assessment: A Handbook for Dealing with Variability and Uncertainty in Models and Inputs*. Springer, N.Y., 1st edition, 1999.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

D. Draper. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society. Series B*, 57:45–97, 1995.

J. Droppo, A. Acero, and L. Deng. Uncertainty decoding with splice for noise robust speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 57–60, Orlando, USA, May 2002.

O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

C.W. Dunnett. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, 50:1096–1121, 1955.

R.A. Fisher. *Statistical Methods for Research Workers*. Macmillan Pub Co., 15th edition, 1970.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

A. Girard, C.E. Rasmussen, J.Q. Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, volume 15, pages 545–552. Cambridge, MA: MIT Press, 2002.

E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

S.A. Goldman and R.H. Sloan. Can pac learning algorithms tolerate random attribute noise? *Algorithmica*, 14:70–84, 1995.

G.H. Golub, Hansen P.C, and D.P. O'Leary. Tikhonov regularization and total least squares. *SIAM Journal on Numerical Analysis*, 30:185–194, 1999.

D. Gorgevik and D. Cakmakov. Combining SVM classifiers for handwritten digit recognition. In *16th International Conference on Pattern Recognition (ICPR'02)*, volume 3, pages 102–105, Quebec City, QC, Canada, 2002.

T. Graepel and R. Herbrich. Invariant pattern recognition by semidefinite programming machines. In *Advances in Neural Information Processing Systems*, volume 16, pages 33–40. Cambridge, MA: MIT Press, 2003.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 2nd edition, 1993.

S.R. Gunn. Support vector machines for classification and regression. Technical report, University of Southampton, May 1998.

M. Hauck, M.A.J. Huijbregts, J.M. Armitage, I.T. Cousins, A.M.J. Ragas, and D. Van de Meent. Model and input uncertainty in multi-media fata modelling: Benzo[a]pyrene concentrations in europe. *Chemosphere*, 72(6):959–967, 2008.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

G. Hommel. A stagewise rejective multiple test procedure based on a modified bonferroni test. *Biometrika*, 75:383–386, 1988.

J. Huang, V. Blanz, and B. Heisele. Face recognition using component-based SVM classification and morphable models. In *International Workshop on Pattern Recognition with Support Vector Machines (SVM2002)*, volume 2388 of *Lecture Notes in Computer Science*, pages 334–341, Niagara Falls, Canada, August 2002.

K. Huang, H. Yang, I. King, M.R. Lyu, and L. Chan. The minimum error minimax probability machine. *Journal of Machine Learning Research*, 5:1253–1286, 2004.

D. Huard and A. Mailhot. A bayesian perspective on input uncertainty in model calibration: Application to hydrological model "abc". *Water Resources Research*, 42(7): CiteID W07416, 2006.

D.W. Hubbard. *How to Measure Anything: Find the Value of "Intangibles" in Business*. John Wiley & Sons, Hoboken, NJ, 1st edition, 2007.

R.L. Iman and J.M.Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics A: Theory and Methods*, 9:571–595, 1980.

T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184, 1998.

M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

S.S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to platt's SMO algorithm for SVM classifier design. Technical report, Technical Report CD-99-14, Department of Mechanical and Production Engineering, National University of Singapore, 1999.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1143, 1995.

R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Department of Computer Science, Standard University, Stanford, California, USA, 1996.

S. Kotz, T.J. Kozubowski, and K. Podgorski. An asymmetric multivariate laplace distribution. Technical report, Technical Report No.367, Department of Statistics and Applied probability, University of California at Santa Barbara, `http://wolfweb.unr.edu/homepage/tkozubow/0_alm.pdf`, January 2003.

G.R.G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M.I. Jordan. Minimax probability machine. In *Advances in Neural Information Processing Systems*, volume 15, pages 929–936. Cambridge, MA: MIT press, 2002a.

G.R.G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M.I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002b.

H. Levene. In *Contributions to Probability and Statistics: Essays in Honour of Harold Hotelling*, pages 278–292. Stanford University Press, 1960.

H. Liao and M.J.F. Gales. Joint uncertainty decoding for noise robust speech recognition. Technical report, CUED/F-INFENG/TR499, University of Cambridge, 2004.

M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

D.J.C. Mackay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 24/28 Oval Road, London NW1, U.K., 1979.

MATLAB. Matlab 7 external interfaces. `http://www.mathworks.com`, 2007. The MathWorks, Inc. 3 Apple Hill Drive, Natick, MA 01760-2098, USA, 2008.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 83(559):69–70, 1909.

K.S. Miller and H. Ruben. Multidimensional gaussian distributions. *The Annals of Mathematical Statistics*, 37(1):301–307, 1966.

MinGW. Mingwiki. `http://www.mingw.org/wiki/MinGWiki`, 2008.

T.M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1st edition, 1997.

H.D. Mittelmann. An independent benchmarking of sdp and socp solvers. *Mathematical Programming*, 95(2):407–430, 2003.

H.D. Mittelmann. Socp (second-order cone programming) benchmark. Technical report, `http://plato.asu.edu/ftp/socp.html`, November 2008.

M.G. Morgan and M. Henrion. *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, N.Y., 1990.

MOSEK. The mosek optimisation toolbox for matlab manual. version 5.0 (revision 105). http://www.mosek.com, 2008. MOSEK ApS, Symbion Science Park, Fruebjergvej 3, Box 16, 2100 Copenhagen Ø, Denmark, 2008.

P.B. Nemenyi. *Distribution-Free Multiple Comparison*. PhD thesis, Princeton University, Plainsboro, New Jersey, USA, 1963.

Y. Nesterov and A. Nemirovskii. Interior-point polynomial methods in convex programming. *SIAM Studies in Applied Mathematics*, 13, 1994.

Y. Nesterov and M.J. Todd. Self scaled barriers and interior point methods for convex programming. *Mathematics of Operations Research*, 22(1):1–42, 1997.

R.M. Norton. The double exponential distribution: Using calculus to find a maximum likelihood estimator. *The American Statistician*, 38(2):135–136, 1984.

A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B*, 40:1–42, 1978.

P. Orinius. Pelles c for windows. http://www.christian-heffner.de/, 2008.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, Amelia Island, Florida, 1997.

J.C. Platt. Fast training of support vector machines using sequential minimal optimisation. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208, 1998.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. the MIT Press, http://www.gaussianprocess.org/gpml, 2006.

G. Rätsch. Benchmark repository used in soft margins for adaboost, fisher discriminant analysis with kernels and robust boosting via convex optimisation. http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm, 2001a.

G. Rätsch. *Robust Boosting via Convex Optimisation*. PhD thesis, University of Potsdam, Computer Science Dept., August-Bebel-Str. 89, 14482 Potsdam, Germany, 2001b.

M. Schmidt. Identifying speaker with support vector networks. In *Interface '96 Proceedings*, Sydney, 1996.

S.S. Shapiro and M.B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, 1965.

D.J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 2nd edition, 2000.

P.K. Shivaswamy, C. Bhattacharyya, and A.J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.

H.A. Simon. Why should machines learn? *Machine Learning: An Artificial Intelligence Approach*, 1:25–37, 1983.

R. Sloan. Types of noise in data for concept learning. In *Proceedings of the First Annual Workshop on Computational Learning Theory*, pages 91–96, MIT, Cambridge, Massachusetts, USA, 1988.

R.H. Sloan. Four types of noise in data for pac learning. *Information Processing Letters*, 54:157–162, 1995.

J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetic cones. *Optimization Methods and Software*, (11-12):625–653, 1999.

J.W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5: 99–114, 1949.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

L. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 560–566, Los Angeles, CA, 1985.

R.J. Vanderbei. Loqo user's manual-version 4.05. Technical report, School of Engineering and Applied Science, Princeton University, `http://www.princeton.edu/~rvdb/loqo/LOQO.html`, September 2006.

V. Vapnik. *The Nature of Statistical Learning Theory: Statistics for Engineering and Information Science*. Springer, N.Y., 2nd edition, 1999.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Application*, 16(2):264–280, 1971.

M. Wallace, S. Ioannou, A. Raouzaiou, K. Karpouzis, and S. Kollias. Dealing with feature uncertainty in facial expression recognition. *International Journal of Intelligent Systems Technologies and Applications*, 1(3-4):409–429, 2006.

V. Wan and S. Renals. Support vector machine speaker verification methodology. In *IEEE International Workshop on Neural Networks for Signal Processing*, volume 2, pages 221–224, Toulouse, France, September 2003.

E.W. Weisstein. Uniform distribution. From MathWorld-A Wolfram Web Resource. `http://mathworld.wolfram.com/UniformDistribution.html`.

C.K.I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, volume 8, pages 514–520. Cambridge, MA: MIT Press, 1996.

R.L. Winkler. *Introduction to Bayesian Inference and Decision*. Holt, Rinehart and Winston, 2nd edition, 2003.

W.A. Wright. Bayesian approach to neural network modelling with input uncertainty. *IEEE Transactions on Neural Networks*, 10(6):1261–1270, 1999.

J. Yang and S.R. Gunn. Uncertain input classification in support vector machines. In *Sheffield Machine Learning Workshop (Poster Session)*, Sheffield, UK, September 2004.

J. Yang and S.R. Gunn. Exploiting uncertain data in support vector classification. In *KES2007 11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Part III*, pages 148–155, Vietri sul Mare, Italy, 2007a.

J. Yang and S.R. Gunn. Iterative constraints in support vector classification with uncertain information. In *International Workshop on Constraint-Based Mining and Learning, at ECML/PKDD 2007*, pages 49–61, Warsaw, Porland, 2007b.

Y. Ye, M.J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19:53–67, 1994.

J.H. Zar. *Biostatistical Analysis*. Prentice Hall, Englewood Clifs, New Jersey, 4th edition, 1998.

F. Zouaoui and J.R. Wilson. Accounting for input model and parameter uncertainty in simulation. In *Proceedings of the 2001 Winter Simulation Conference*, pages 290–299, Arlington, Virginia, USA, 2001.