Geometric margin domain description with instance-specific margins

Adam W. Gripton

A dissertation submitted to the Postgraduate Studies Committee
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Heriot-Watt University
Department of Physics
School of Engineering and Physical Sciences

May 17, 2011

Abstract

Support vector domain description (SVDD) is a useful tool in data mining, used for analysing the within-class distribution of multi-class data and to ascertain membership of a class with known training distribution. An important property of the method is its inner-product based formulation, resulting in its applicability to reproductive kernel Hilbert spaces using the "kernel trick". This practice relies on full knowledge of feature values in the training set, requiring data exhibiting incompleteness to be pre-processed via imputation, sometimes adding unnecessary or incorrect data into the classifier. Based on an existing study of support vector machine (SVM) classification with structurally missing data, we present a method of domain description of incomplete data without imputation, and generalise to some times of kernel space. We review statistical techniques of dealing with missing data, and explore the properties and limitations of the SVM procedure. We present two methods to achieve this aim: the first provides an input space solution, and the second uses a given imputation of a dataset to calculate an improved solution. We apply our methods first to synthetic and commonly-used datasets, then to non-destructive assay (NDA) data provided by a third party. We compare our classification machines to the use of a standard SVDD boundary, and highlight where performance improves upon the use of imputation.

Acknowledgements

I would like to thank my supervisor, Dr. Weiping Lu, for constant help and guidance throughout this project and for allowing me the independence to guide my research direction whilst ensuring I remained focused. I also owe my gratitude to Prof. Michael Christie, whose critical feedback and mentoring have allowed this thesis document to do justice to the project. My thanks also go to Prof. David Corne for many helpful discussions and providing his services as internal examiner, as well as external examiner Dr. Bruce Worton for his constructive criticism.

I also acknowledge the input of Dr. Peter Clifford and Dr. Geoff Nicholls for their in-depth discussion of multi-classifier methods in that I include in §3.4.3. Thanks also to David Tax for his input during the ICPR 2010 poster session.

Finally, I offer my gratitude to the ever-helpful library staff at Heriot-Watt University, Edinburgh University and Glasgow University for providing me ease of access to the necessary material I required to produce this document.

This project was funded by a CASE award principally funded by the Engineering and Physical Sciences Research Council (EPSRC). Acknowledgement is also due to the Atomic Weapons Establishment (AWE) for providing the emissions dataset, and to the team there for providing general advice and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 What is classification?

Classification is, put simply, the basis of all decision making. It is the process whereby the nature of an inherent property of an object is analysed or predicted. The *capiscum* pepper can be classified by colour into red, yellow, green or orange. People can be classified by gender, race, religion, occupation, amongst a variety of other classes. Music could be classified by the nature of its tempo, its timbre or its mood. It extends also to analysis of utility or worth: a picture could be classified as a work of art or a child's scribbling; an old clock could be classified as a priceless antique or a worthless fake; a political policy could be classified as worthy of submission as a bill in Parliament, or considered unworthy and thrown out. Classification is at once both informative and predictive, works essentially on known data, and when employed for prediction uses *feature information* – measurable aspects of an object's known properties – to form the basis of the nature of the labelling of an object. It is both subjective and objective, and

both highly specialised and the most natural reflex known to man: juries debate over long time periods to classify defendants as guilty or otherwise based on the interpretation of complex evidential data, yet even in the womb an unborn baby can recognise a voice as belonging to its parent from instinct alone. Most significantly, it is an inherent ability bestowed upon all living things, but is something that an untrained computational system is entirely incapable of. Furthermore, as we shall see, unlike many other tasks, the task of programming a computer with *artificial intelligence*, which we broadly define here as simply the ability to classify in whichever context the term is used, is sometimes highly complicated and requires significant programming effort to attempt. This dichotomy is no more clearly seen than in the field of *image recognition*.

1.1.1 A simple exercise

(a) (b) (c) (d)

Figure 1.1: Four images: Which is the odd one out?

To demonstrate this ability inherent in a natural cognitive (and not necessarily human) mind, we shall begin this discussion with a few simple questions. Firstly, we shall ask which of the following 200×200 pixel images in Figure 1.1 is the odd one out. Next, we consider the set of sequences of numbers given in Table 1.1, and ask for each of these four sequences, which number is the odd one out.

$$\{1, 13, 47, 28, 39, 11, 17, 25\} \tag{1.1}$$

$$\{9, 121, 144, 49, 57, 64, 16, 100\} \tag{1.2}$$

$$\{e, \pi, \sqrt{2}, \frac{5}{7}, (3)^{1/3}\} \tag{1.3}$$

$$\{5, 19, -13, 37, 61, 15, 2\} \tag{1.4}$$

Table 1.1: In each sequence, which is the odd number out?

For the first problem involving images, the choice would immediately be clear to even a small child; image (b) depicts a puppy whereas the others depict kittens. Nevertheless, the high-speed decision-making computations involved in our brains when computing this seemingly trivial decision are themselves complex, and a decision which seems simple to us would perhaps not be as simple to a computer. We return to this point below. In the second exercise, it takes only a passing knowledge of primary-school arithmetic to deduce the answers to the first two problems: in the first, 28 is the only number we could call 'even', and in the second we see that 57 is the only number here which is not a square of a natural number. In the third, we may require some high-school mathematics in recognising that $\frac{5}{7}$ is the only rational number in this set. However, with the fourth problem, no amount of intuition is seemingly of help. Is it (-13), being the only number below zero? Is it 15, being the only number whose modulus is non-prime? Perhaps it is 2, for being the only even number? Suddenly we see that it is not always possible to be sure of ourselves in seemingly simple challenges such as this. We could, for example, have realistically said that in the first problem, 47 is the only number above 40; or that in the second, 64 is the only sixth-power of a

natural number in the list. Figure 1.1(a) could equally be seen as different, since it depicts a sleeping animal, whereas the others are awake. A multitude of rules could be derived, in fact, to prove that in any of the cases above, one particular number or image, or even many, could be said to 'stick out' apart from the others, although some rules make *more intuitive sense* than others. What is wrong, for example, with saying 100 is the only number in sequence 1.2 that is equal to 100?

1.1.2 Features of a decision rule

In general, the following questions could be posed of the way in which we could handle any process of classification:

- How do we choose from a seemingly infinite wide range of decision criteria to arrive at the most 'intuitive'?
- How do we know when a problem can and cannot be solved in a simple way?
- Why in this problem is it better to develop a *decision rule* based on the data given, as opposed to proposing a model *a priori* (before any evidence)?
- How can we convince ourselves that our decision rule is accurate and will hold true given *novel data* – that is, new examples drawn similarly on which to test our basis of decision-making?
- Would a computational machine be able to arrive at the same conclusions as us given the problems above?

All of these concepts are important in the fields of *classification* and *machine learning*. Machine learning can be defined[1] as:

> "The process or technique by which a device modifies its own behavior as the result of its past experience and performance."

The motivation for training a computer to perform a *classification task*, as opposed to, say, attempting to perform the same task by employing humans alone, is simple to frame. Suppose we had one million sets of images of kittens and puppies due to various types of photographs, similar to those given in Figure 1.1 above, and it was important that they were classified accurately and quickly. Although a human operator may be able to process sets of odd-one-out challenges involving discrimination between kittens and puppies with a good degree of accuracy – perhaps over 99% – the nature of the human body would ensure that the time taken to process a million sets of these images would be roughly 16 weeks of flat-out work on a 35-hour per week schedule, non-stop at the rate of one every two seconds. However, although this may cause considerable psychological stress to the unfortunate person involved, it is still true that the accuracy rate would ultimately be impressive given only the intuition imparted onto the person as a child growing up, with no special training needed in order to perform this particular task. This example leads us to the following 'philosophy' behind the difference between human learning and artificial intelligence:

> Humans, although slow and inefficient, are **incredible classification machines** in terms of accuracy, intuition and simple model comparison.

[1] McGraw-Hill Dictionary of Scientific and Technical Terms, 6th edition

The numerical examples above show this well, in that we can say it makes more sense given the first set of data to look at the remainder of each datum when divided by two, rather than to say (albeit still correctly) that only one of these numbers is above 40. Both decision rules lead to one of the datapoints being classified differently from the rest, which achieves the required aim, and yet working with odd or even numbers seems more suited to this task. If we were asked to describe our thought processes in how we may reach this decision of one model's provenance over another, however, we may be stumped; similarly, if we were asked to design a hard-and-fast rule for when the even-number criterion should be applied to a dataset and when the above-40 criterion should be applied, it is equally unclear. To give an example of this, which is the odd one out in the following dataset?

$$\{1005, 1001, 1007, 39, 1008, 1011, 1009, 1023\} \tag{1.5}$$

Although this dataset has the same properties as Set 1.1 above – namely, it has eight elements each consisting of natural numbers – it now seems more obvious which of these competing rules to apply: clearly the entry 39 'looks' more out of place, even though 1008 is again the only even number in the set. Briefly returning to the image classification problem above, we also see from the variance present amongst the images of kittens that a hard-and-fast rule is difficult to arrive at. Image (a) in Figure 1.1 does not show open eyes, restricting our knowledge of their colour; image (c) shows a different colour of cat and (d) has an off-white and non-trivial background. When we consider trying to impart this intuitive knowledge to a computer in terms of definite code it can use to arrive at a decision, therefore, the scale of the problem is daunting. Furthermore, the lack of intuition and common sense possessed by a computer means that when the images in Figure 1.1 are

considered by the machine, instead of seeing kittens and puppies, it sees:

$$\{x_1, x_2, x_3, x_4 : x_i \in \mathbb{N}_k^{200\times 200}\} \tag{1.6}$$

where here $\mathbb{N}_k$ refers to the set of natural numbers between 1 and the colour depth involved, perhaps $\{1 \dots 256\}$ or $\{1 \dots 2^{24}\}$. That is, the computer will only see datapoints consisting of lists of numbers which have 40,000 entries each. With this information alone, the machine classification process is all but futile. As the data are of image form, we know *a priori* that the pixel values will be highly correlated, with all 40,000 bits per image of information clearly not all being required; and thus moving any of the images by one pixel in any direction, while completely changing the vector of numerical values, will alter very little about the given information to the human observer. Conversely, altering the pixel values with a complex but known re-arrangement function will produce an entirely obliterated image to the human observer, but one which can be perfectly reconstructed – and can therefore be seen as equivalent – to a computer. However, suppose for a moment that all this intuitive information and training data could be given to a computer, such that it could perform the evaluation of the following decision function:

$$f : \mathbb{N}_k^{200\times 200} \rightarrow \{0, 1\}, \tag{1.7}$$

with equal accuracy to a human observer; that is, it can take an image such as those above and produce a binary result it decrees to be the most likely nature of the image concerned, with 0 and 1 referring, say, to a kitten or a non-kitten respectively (it is somewhat misleading to assume that 200×200 pixel images of white noise could be meaningfully 'classified' as being of a puppy, hence our notation here; if this is also required, we might adapt our training data and decision

function to map to $\{0, 1, 2\}$ instead). With the speed and reliability of computers, plus their lack of fatigue and ability to compute non-stop for an indefinite continuous time period, even if this function evaluation takes 0.1 seconds – itself a somewhat conservative estimate – the required classification of the four million images (equivalent to choosing the 'odd-ones-out' one million times) would take 4.62 days, meaning an operator could leave the machine running at 5pm on Friday and have results by 9am on Wednesday!

1.1.3 Stability and feature extraction

The only challenge which presents itself here, of course, is to be able to accurately construct a function f from enough training data to begin with, and ensure that the decision function is somehow *stable* – that is, it can deal with novel data which have been altered in some realistic way:

(a) Trans. (b) Stret. (c) Noisy (d) Obscured (e) Mapped (f) All

Figure 1.2: Effects of altering images

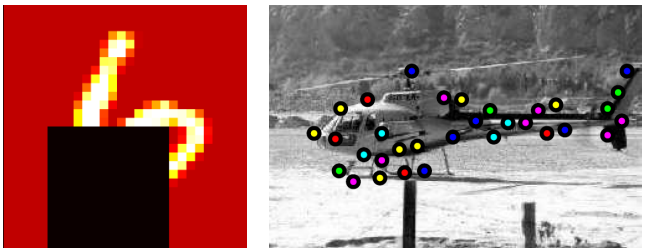
- (a) translated through a small number of pixels in any direction
- (b) stretched or skewed by a small amount
- (c) augmented by a small amount of noise (Poisson or Gaussian)
- (d) obscured over a small amount of the picture by e.g. a black box

- (e) mapped into a different colour map, or (f) a combination of these.

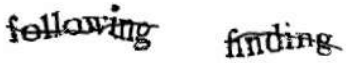
Generally, these properties can apply to a wider class of datatypes than images alone, and some form of them can often be queried of any classification process. In general, good *feature extraction* from raw data is necessary and sufficient to have a classification system exhibiting reasonable performance. For example, it is clear that if our decision rule was based solely on the value of the pixel at location (x_1, y_1), it would be incredibly unstable, whereas a sufficiently good feature extraction algorithm could find in an image, for example, the orientation of the ears or analyse the markings of the animal independently of the information corruption present in the image. A common rule, and one which a designer of an artificial system would do well always to observe, is that wherever features are extracted from raw data, it is usually more useful for them to have some *physical significance* based on the nature of the raw data which has been observed. If we know that our classifier will be dealing with pictures of animals and its task is to classify their nature, what is the use in simply querying the value of one particular pixel if we know from the outset that this pixel would be rendered meaningless by any of the corruption methods detailed above? Similarly, if we have a collection of natural numbers, it is perhaps of significance that they belong to $\mathbb{N}$ and not $\mathbb{R}$, and thus a general technique applying to real numbers in general may not always be the best one – we return to this concept of simplicity in the section §1.1.4 below.

Examples in literature This example, though in itself not very serious, is in fact indicative of many open problems in machine learning. Chechik et al [14] give some good examples of these in their paper, notably those of obscuring areas in handwritten digit recognition from the MNIST data set due to LeCun [43],

and detection of the presence of vehicle shapes in closed-circuit camera images, studied by Berg [7] amongst others. Moreover, the abilities of humans over computers to recognise heavily distorted data is put to use by the CAPTCHA system [46]. Though here, we maintain the examples as being those with respect to image analysis, it is important to remember that human recognition can also be equally astute in terms of more general numeric settings, as seen above with the numerical-based classification problems. Figure 1.3[2] shows examples of these



(a) Handwritten digits (b) Feature detection in image



(c) CAPTCHA

Figure 1.3: Examples of human vs. machine classification

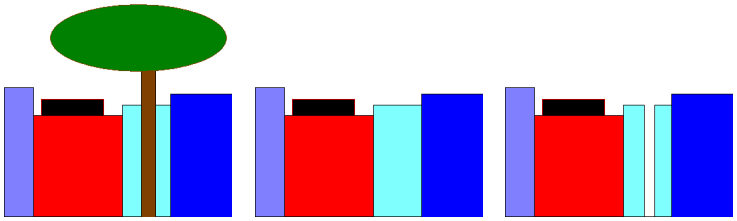
three contexts of image analysis; all three instances show situations where human-based picture classification can easily detect something, but where machines classically find it difficult to extract the correct meaningful information. Figure (a) is taken from the paper by Chechik et al and shows an example of a major feature of the number 6 being obscured. Figure (b), from the paper by Berg, demonstrates the attempted finding of the helicopter object within the image by an automated algorithm. Finally, Figure (c) demonstrates the utility of the CAPTCHA algo-

[2]Subfigure 1.3(b) is reproduced with kind permission of Alex Berg, `www.acberg.com`.

rithm, which prevents automated 'spam' entries, generated by a so-called 'bot' script, from leaking into the submission forms of websites by requiring that these words are typed in along with the submission data. As shown, the letters are often subject to re-shaping by some nonlinear contour and obstruction by lines; features that a human eye can pick out and see beyond, where a machine would find it difficult.

1.1.4 Model simplicity

> "A theory with mathematical beauty is more likely to be correct than an ugly one that fits some experimental data." – Paul Dirac



(a) Observed data (b) One scenario (c) Another scenario

Figure 1.4: Model simplicity: Which scenario is more likely?

To illustrate the importance of the simplicity of a model, consider the example given here in Figure 1.4 taken from the book by MacKay [50]. In Figure 1.4(a) we see a simulated "picture" with boxes lined up and an obstruction to the view (e.g. a tree). Figures (b) and (c) give two possible scenarios for the model, each providing their own completion of the sector of the image which we cannot see. Figure (b) provides a model that behind the 'tree' there is one large cyan building; Figure (c) says instead that two cyan buildings of equal height are positioned behind

it, with a small gap in between them smaller than the obstruction. Both models are, therefore, consistent with the observed data. Intuitively, we may say that the scenario given in Figure 1.4(b) is somehow more 'likely', because the likelihood of there being two boxes which just happen to be of the same colour and size with the gap between them falling behind the obstruction is perhaps quite slim. In a similar way, if we return to the set of numbers given in Equation 1.5 we could say that given the information that one point is to be classified differently from the others, the model of whether an entry is greater or less than (say) 1000, would intuitively be more likely than an odd vs. even model. Both of these 'hunches' rely on the assumptions we make about the model that has generated the data in the first place: with the boxes, for example, the assertion that two congruent cyan boxes next to each other is unlikely may itself assume that the distribution, colour and size of boxes is randomly distributed.

1.1.5 Occam's razor

> "It is vain to do with more what can be done with less." – William of Ockham (Occam) (c. 1288 - c. 1348)

This quote from Occam, a Franciscan friar, shows that this principle has been understood for many centuries. The principle attributed to him known as *Occam's razor* [11], namely "entities must not be multiplied beyond necessity"[3], is a significant consideration in all classification tasks. An heuristic principle alone as opposed to a rigorous mathematical theorem, it implies of many competing models used to describe a phenomenon that in general the simplest explanation

[3]Original Latin: "entia non sunt multiplicanda praeter necessitatem"

should be seen as the most likely. We can see this illustrated with the boxes example above: why assume that there are two boxes of the same colour and shape, thus complicating the model and adding extra *parameters* unnecessarily, where we could just say it is likely that only one box exists?

1.1.6 Curve fitting

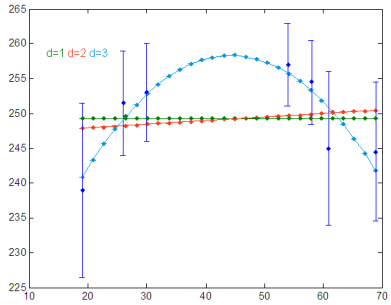


Figure 1.5: Occam's razor in curve fitting: which approximate fit is more likely?

A similar problem arises when we look at a simple curve-fitting problem to a handful of one-dimensional datapoints. The challenge, as presented in Figure 1.5, is to find firstly a member of the class of functions which are polynomials over the variable x, and furthermore a set of parameters governing a model based on this class member of functions which is optimised over a least-squares fit to best fit the data as given. To clarify, the three fits that we depict in Figure 1.5 are the following members of the polynomial class over x:

$$f_0(x;\mathbf{c}^{(0)}) = c_0^{(0)} \tag{1.8}$$

$$f_1(x;\mathbf{c}^{(1)}) = c_0^{(1)} + c_1^{(1)}x \tag{1.9}$$

$$f_2(x;\mathbf{c}^{(2)}) = c_0^{(2)} + c_1^{(2)}x + c_2^{(2)}x^2 \tag{1.10}$$

where here $\mathbf{c}^{(0)} \in \mathbb{R}$, $\mathbf{c}^{(1)} \in \mathbb{R}^2$ and $\mathbf{c}^{(2)} \in \mathbb{R}^3$ are the parameters for each of these models, and it is assumed that we have an optimisation procedure capable of finding, in each case, the *maximum likelihood* (ML) value of each of these parameter vectors over their respective domains, given the data. That is, the following objective for the error function committed by each of the functions f_k above can be minimised over $\mathbf{c}^{(k)}$ in each case:

$$\hat{\mathbf{c}}^{(k)} \equiv \arg\min_{\mathbf{c}} \phi_k(\mathbf{c}^{(k)}) = \sum_{i=1}^{7} \left(\frac{y_i - f_k(x_i; \mathbf{c}^{(k)})}{\sigma_i} \right)^2, \tag{1.11}$$

where (x_i, y_i) refer to the observed data, and σ_i refers to each of the errors involved in these measurements (this error function is known as a *weighted least-squares fit*). In the diagram, this observed data with error bars is given with the dark blue line; the fit for f_0 is green, f_1 red and f_2 cyan. It is visually obvious that f_1 fails to make much improvement on the fit provided by the constant function f_0; however, f_2 could be seen as a better model fit. However, care should be taken before we make a statement such as "f_2 is thus *clearly* the better fit to this dataset"; note that there exist seven datapoints in this example, and thus the following function could be constructed:

$$f_6(x; \mathbf{c}^{(6)}) = \sum_{k=0}^{6} c_k^{(6)} x^k, \tag{1.12}$$

providing a 'sextic' fit over the data such that for the correct value of $\mathbf{c}^{(6)}$, the mean-squared error would be zero, or less technically that the polynomial function $\phi_6(x)$ would pass *exactly* through the centre of each datapoint as shown. Clearly, it is erroneous to say that this model is therefore perfect: it only fits the data as it is so complex, and even a small change in the values of the points (x_i, y_i) may result in a wildly different function definition for f_6. This situation is analogous to that alluded to in the previous section, where in the odd-one-out exercises it is

somehow not intuitive to come up with decision rules such as:

- "In exercise 1.2, 100 is the odd one out as it is the only value x_i for which $x_i = 100$"
- "Figure 1.1(c) is a kitten as the RGB value of its pixel at (60, 40) is (212, 157, 116)"

From the examples given above, we see that the major challenge presenting a designer of an artificially intelligent classifier is twofold: to pick out the best features according to the task it is required to perform, and to process the data in the most efficient, meaningful and appropriate way, again according to the problem in hand. Good classification could be therefore seen as something of an artform; no two problems will be the same, as the types of data and the end results are so wildly varying. Mathematical methods provide the basis of how to classify in certain cases, but in general there is no one-size-fits-all way of designing a general classifier, simply because such a concept does not exist. Prior information about the required task must always be employed, and sensible, human decisions made on *a priori* knowledge must provide the foundations behind the designing of a code in order for reasonable results to be expected.

1.1.7 Remarks

By employing artificial intelligence in solving classification problems, we achieve the following aims:

- We can *categorise* objects or concepts according to their measurable quantities.

- We can make *decisions* on the nature of an object based on realistic models of how these features interact with each other.

- We can make predictions of the nature of a novel object when we have the required information about the nature of other examples we have seen before: this is known as *training data* and will be dealt with in depth in Chapter 2.

- We can *quantify certainty*, i.e. ascertain how sure we are about a decision given sufficiently good statistical methods.

In Chapter 2, we will return to techniques which ensure the proper management of data. We have observed in this chapter that classification is a natural and vital process in living things which is, in all but fairly trivial cases, a difficult problem to train an automatic computational system to reproduce well through artificial intelligence. We have also noted that the principle of Occam's razor shows that the simplicity of a model is a vital property of any context in which classification is taking place, and should be of primary consideration in any realistic attempt to classify data. Thus, Chapter 2 will build a more mathematical basis into the ways in which mathematical classification is performed with abstract datasets, through methods such as alignment techniques, use of Bayes' theorem and distribution estimation amongst others.

1.2 Verification

The problem of pattern recognition and feature extraction in gamma-ray emission spectra from decaying radioactive sources is one which, once researched,

could yield far-reaching and contextually surprising applications in many areas. A third party corporation, along with the EPSRC[4], have provided partial funding to this project on the basis that classification techniques could be used to analyse data collected by their Non-Destructive Assay (NDA) group from the decay of radioactive elements, more details on which are provided in later chapters. A major problem when dealing with radioactive material which cannot be readily inspected is that of verification. An engineer charged with the task of verification of a fissile material may be presented with the following obstacles:

- The machinery he is allowed to use during inspection may be limited in its accuracy or clarity;
- The time in which he is allowed to make observations may be limited ('short count times');
- The signal may be deliberately altered in some (possibly nonlinear) way;
- A shielding method may be imposed to make the signal weaker or noisier; and so on.

In each of these cases – and indeed when discussing any potential obstacles – we need to ask two important questions:

1. After these tactics have been applied to limit the information which the engineer may obtain, is it still mathematically possible to verify presence of a material, and if so, to what confidence level?

[4]Engineering and Physical Sciences Research Council

2. Suppose, knowing the way in which the data had been corrupted, the engineer designed a relevant 'decision-boundary' algorithm to which information was input and whose output was, for example, the most likely substance which was present. Could this machine itself be inspected to deduce its training data?

These questions are the main motivation for a mathematical study into whether a classifier can be produced in each case, and whether it can somehow be made 'singular': that is, to make the following representative function non-invertible:

$$f : (T, X) \to (\hat{c}, \alpha) \tag{1.13}$$

where here T represents the set of training data, X the set of observations, $\hat{c}$ some maximum likelihood estimate for the class of data being observed, and α the confidence level that this decision is correct. This type of analysis lends itself well to Bayesian statistical methods, since a quantitative estimate must ultimately be produced for both the outcome and its likelihood. Using tools such as principal component analysis, similar dimensional-reduction problems can be implemented for noisy images. One of the initial aims of a study into this area is to ascertain the effect each above obstacle would result in, and the ways in which this can be resolved by statistical methods.

1.2.1 Expert replacement

The usual way of performing this type of analysis – where, say, the results of a spectroscopy experiment conducted on a decaying piece of radioactive material will provide input, and a list of constituent elements/shielding provide results for output – is for experts to be employed. These scientists, usually with deep

knowledge of the idiosyncracies of the experiment in question (and therefore with significant *a priori* knowledge of the model driving the data construction), would require as much clear information as possible about the data, and they would use their experience to determine their statement about the nature of the object being studied. This project seeks to reduce the need for experts, instead concentrating on the creation of a "black box" decision making machine which can be operated by a layman and which, ideally, itself would not contain any sensitive information; but which would provide the same answers as those of an expert's opinion.

Technical issues How are experts replaced? Questions that may be raised of a black box machine would be of the form:

- Would it be possible for anyone to inspect its contents publicly?
- What sort of results would it produce?
- How trustworthy would its results be?
- How stable (insensitive to small alterations in initial conditions) would it be?
- How secure would knowledge of training data be on inspection?

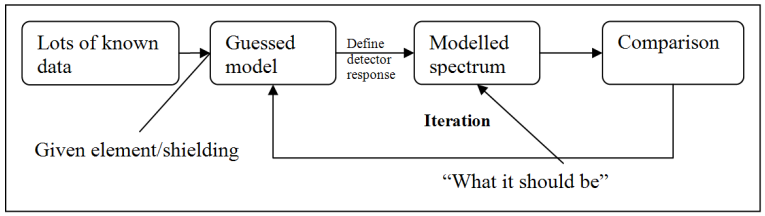


Figure 1.6: Expert replacement system

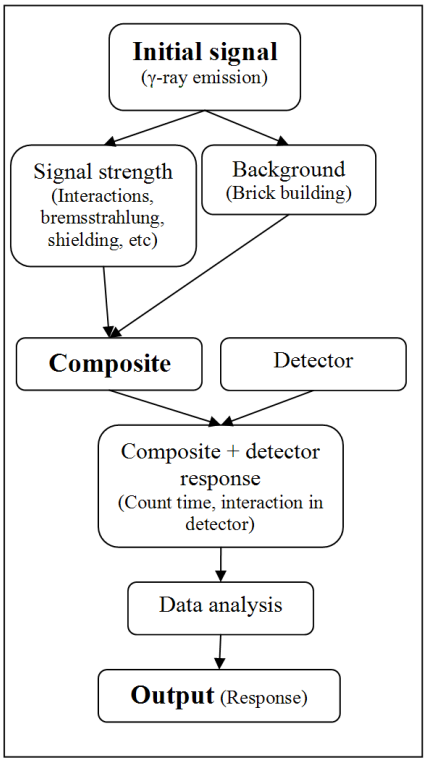


Figure 1.7: Signal construction in assigned dataset

The usual process can be modelled as an iterative flowchart, as shown in Figure 1.6. In terms of how a signal is constructed, Figure 1.7 shows the relevant flowchart for this. As a result of this complex process, there are many potential features of a spectroscopy-based signal to analyse, thus giving plenty of choice for the features to extract. In a consultation regarding this project, however, it was agreed that any classification procedure should not attempt to 're-invent the wheel' by approximating too closely the methods of an expert in the usual process: clearly, if a great deal of knowledge was gained about exactly how the processes in Figure 1.7 affected a signal and their quantitative nature in each case of a decaying material and shielding method, the project would – paradoxically – end up being of little worth, since many of these facts would already need to be known.

1.2.2 Target for this study

If this project is to have been regarded as a success, it must have achieved two aims. Firstly, it must shed light on the "expert replacement black box" classification problem as originally posed, on condition that sufficiently complete and inter-comparable training data is provided; secondly, it must break new ground in terms of an abstract classification problem. The latter aim we describe more in depth in the following section §1.3. These should be seen as two parallel aims: if we are to be successful in the mathematical theory behind techniques based on types of data found in the consignment, results gained through our process should be able to carry over to using it in this particular application.

Clearly, any classification system is only ever as good as the data it is provided with, and thus we shall dedicate a section later on to describing the various feature extraction algorithms we shall employ. As stated above, effective feature extrac-

tion in any application will lead to accurate measurements of underlying physical quantities that change the nature of the observations, leading ultimately to better separation of data by property in the feature space. This in itself is a requirement of a classifier which can give results that are more reliable, and with a greater degree of certainty.

In conclusion, the following should be seen as a "high-level" target for the study:

> A rough "black box" classifier, which can take spectral, neutron and image information and produce some sort of statement about what is going on, insofar as it will be able to tell information set A from B with a certain accuracy.

Consultation The form of data provided was always seen as crucial in order to give the project the best chance of success, in that consultations were held at an early stage to ensure that the data was complete in terms of the nature of the classification that was required to be done. Furthermore, the final dataset showed a sufficient quantity of data to perform a classification task: a classifier system can only function if there is an element of inter-comparability in the data suite it is provided with; and equally, the more repeated experiments, different timescales, etc. the better in terms of enhancing a classifier's usefulness. Another way in which the external liaison parties were of significant assistance was by providing useful pointers towards feature extraction: this will be covered more fully in a later section.

Factors beyond the remit As we shall see, there are inevitably other factors regarding the set of experimental data, such as apparatus distance, location in the room, geometry of the hemispherical shell, etc., which should be considered as beyond our remit. The priorities for the study will lie in examining the differences between fissile elements, shielding material, and timescale. Nevertheless, the study should be seen as a success if it is relatively simple to provide an extension of its method should another very significant parameter arise which was hitherto unaccounted for. Caution should again be exercised here, however, as explained in the above section regarding Occam's razor: it can sometimes be tempting to try and model extra variables, but this may only have an effect of degrading an otherwise elegantly simple system.

1.2.3 Remarks

This report will also, therefore, serve as an introduction to this area of study. A full explanation of the process of gamma-ray spectrometry and neutron data collection will be given in Chapter 7, detailing the relevant phenomena which occur, and in many cases must be overcome, in the analysis of a gamma-ray spectrum. A discussion on shielding methods will be provided, and the effects of the above corruption tactics will be considered. The theory will be used to examine the data which has been received, and finally in Chapter 8 a full set of results and analysis will be produced on the relevant data.

1.3 Project motivation

So far, we have introduced the following parts of the thesis: Chapter 2 will give a survey on statistical methods for classification given a dataset; Chapter 7 will describe the feature extraction process and consider the types of data provided; and Chapter 8 will provide the necessary results on the set we have been provided with. The remainder of the thesis we dedicate to developing a new technique in classification, based on certain specific assumptions about phenomena which could occur within an abstract dataset, most notably that of *missing data*. In order to achieve this, we shall need to base it around a certain theme, which we describe below.

1.3.1 Central theme

The following subsection describes a 'vision' for this project. Various terms used in the paragraph which follows will be terms that shall be described in later chapters, since the technical details are not within the scope of this introduction section. Thus, for now we describe the project conceptually. The thesis is as follows:

Thesis specification It is possible to create a non-parametric statistical classification machine designed for a general, abstract dataset of real values, such that where data is missing through non-applicability rather than being not measured, the classifier can use the given information for the data present without making any assumptions about the distribution of the missing data, thus avoiding imputation methods. Furthermore, the process will be readily extendable to certain types of kernel reproducing Hilbert spaces. In particular, we shall show that sup-

port vector domain description based classification, a method used in literature for multi-class analysis, can be extended to handle missing data in a way which eliminates the need for a pre-processing step via an imputation method. The method will have direct applications to the structure and type of data present in the dataset provided, and thus will achieve two aims. Firstly, it will provide a logical, rigorous extension to methods already present in literature dealing with kernels and those dealing with missing data; and secondly, it will provide some insight into a way in which the provided dataset could be analysed in this way.

1.3.2 Application to the external dataset

Writing retrospectively, we can describe the reasoning behind the choice of direction as described above. As we shall see in Chapter 7, the provided datasets consisted of multiple experiments per session using different types of detection equipment. These in themselves were incomplete, in that sometimes a particular detector would not be used. Thus, given a sensible feature extraction process which provides some features for each of the detector readings themselves, the natural state of the full dataset produced will be that of one with missing data. We wish to provide a method which does not assume the results of any detector experiment that was not performed, whilst being able to operate in a kernel space and thus provide greater flexibility with regards to the subtle non-linear features which could be exhibited by any abstract dataset.

1.3.3 Structure of thesis chapters

This thesis document will be divided into three principal sections, corresponding to three themes: firstly, we deal with the **background** information and literature reviews in Chapters 2, 3 and 4; secondly, we introduce **our two methods** in Chapters 5 and 6, including the results of application of these methods to artificial data in §5.6 and §6.7. Finally, we devote Chapters 7 and 8 to a full study of the physical processes, feature extraction and classification involved with the **dataset provided** for us, drawing our conclusions from the study in Chapter 9. The three themes are explored in more detail below.

Background and Literature As already mentioned, **Chapter 2** will deal with classification in a very general sense, observing the 'usual' methods which are used to arrange and classify abstract datapoints with statistical methods. It will describe these methods with the assumption that no data is absent within a dataset, and will thus also explore the methods of kernel spaces, describing the processes involved in extension to these more general formulations. The concept of the support vector description machine will be introduced with the intention of developing it later once combined with methods described in the following chapters. **Chapter 3** will provide a full introduction into the current statistical theory of missing data, paying particular attention to the cases where a known probabilistic-based method works on specific assumptions of an underlying parametric model driving the dataset and the modelling of the missingness case as being essentially random. We will comment on various useful methods which have been developed in this case, and show in which situations they could exhibit shortcomings or make too many assumptions. Methods dealing with *structurally missing* data will fol-

low so as to provide some basis for building a later theory specifically tuned to this type of data absence. Following on from this, **Chapter 4** will provide an in-depth description to a method developed by Gal Chechik et al for training a specific type of classifier – the *support vector machine* – to handle structurally missing data. This description will introduce the concept of instance-specific margins on which we shall later build our methods. We review the methods used and note the analysis with which Chechik et al arrived at each method. We observe the areas of this method it is possible to take forward and motivate our use of the instance-specific margin, itself taken from the principle of this method. Particular care will be taken in the analysis of how their method applies to reproducing kernel Hilbert spaces, as this is an important property a classification method can possess to generalise into an arbitrarily complicated mapping space, thus being able to combine a method designed for use on the input features into an equivalent one applied to complex products of these.

Our methods In **Chapters 5 and 6** we take ideas from the previous three chapters and show how they can be combined into a full theory of the support-vector domain description function that operates under the presence of structurally missing data. The concept of the SVDD, and the motivation for its use, will be taken from Chapter 2, as will the concepts involving the use and utility of kernel spaces; the theories already present to deal with multiple imputations will be used from Chapter 3; and the concept of instance-specific margins from Chapter 4 will also feed into the theory. Specifically, Chapter 5 will deal with the introduction of our 'exact-centre' (XC) method, and Chapter 6 with our 'dual-optimisation' (DO) method. We provide results in both of these chapters from analysis of datasets

with structurally absent features, and identify the situations in which they outperform an SVDD with imputed data in terms of sphere volume.

Provided data **Chapter 7** will introduce the physical concepts behind radiation emissions data, giving a full background into the structure of the dataset and the processes which produced the various kinds of data. This information will then be used in **Chapter 8** to motivate the choice of physically significant features we decided to extract; we shall then provide a full account of the classification activities we performed on the dataset, giving our conclusions to the whole study in **Chapter 9**.

Chapter 2

Classification with full data

2.1 Conventions

Datasets We define a *dataset* $X = \{x_i\}_{i=1}^N$, typically $x_i \in \mathbb{R}^d$, as being a collection of N observations (known as *datapoints*) of some phenomenon, each of these being furnished with d distinct characteristics, known as *features*. The natural way to describe a structure such as this is in matrix form, with the matrix X being conventionally of dimension $(N \times d)$. A well known example is Fisher's iris data [23]: this dataset contains $N = 150$ points corresponding to observations made of different iris flowers. Each point contains $d = 4$ features, denoting the different observations made: namely the sepal length and width, and the petal length and width. Each flower was measured in all of these ways, producing a dataset with 600 pieces of information in total.

Classes In many cases, the nature of the observed phenomenon being recorded by the data X may exhibit the ability to be divided into *classes*. In the case of

Fisher's iris data, these classes denote the 3 different species of iris (*setosa*, *versicolor* and *virginica*) being measured, and this known information was recorded at the same time as the measurements. In general, where it is appropriate to include this information, the dataset may be accompanied by some *class data* $\mathbf{y} = \{y_i\}_{i=1}^{N}$, $y_i \in \mathbb{N}$, the purpose of which is to assign a label to each datapoint x_i describing its particular nature. Often we find that $y_i \in \{0, 1\}$: this is known as *binary classification*, and specialised techniques exist to make full use of this assumption. Where there exist more than two distinct classes, the zero class is usually omitted by convention: correspondingly, the iris data would be accompanied by $y_i \in \{1, 2, 3\}$.

Training and testing data We can define a *classifier*, or *classification machine*, as follows:

> A classification machine is a system which, once provided with suitable training information furnished with class data, can make an informed decision as to the likely class of a given testing point it is required to analyse.

Returning again to the iris dataset, a good classifier trained on Fisher's data will be able to accurately output the likely species of an iris flower we provide, given only the same four measurements obtained from it as detailed above. The data bank required for preliminary analysis by a classifier in order to make an informed decision is known as the *training* data, and will consist of datapoints X_{TR} furnished with class data $\mathbf{y}_{TR}$. The information subsequently used to challenge the classifier is known as the *testing (novel)* data, and will consist only of datapoints X_{TE}, the classifier's task being the accurate output of the unknown $\mathbf{y}_{TE}$. The training

information typically affects a classifier's behaviour; the testing information does not. Occasionally a classifier will also be able to accurately reject a sample as not being likely to reside in any of the trained classes, and in this case a *reject class* is sometimes permitted in the output $\mathbf{y}_{TE}$, rather than force the classifier to make a guess it considers to be inaccurate.

Cross validation In many cases, the ideal output $\mathbf{y}_{TE}$ is known a priori, and the task is instead to assess a classifier's performance. *Cross validation* is often used for this purpose: a small, representative sample of data X_R is removed, and the classifier is trained on the remaining points. The classifier is then shown X_R as testing data and required to give its estimation of the corresponding class information $\mathbf{y}_R$, which is known by the examiner. The classifier's estimations are then recorded alongside the true class information and the process is repeated with a different section of X being used as X_R. Usually this is done in such a way that all data is included in X_R at some stage; stochastic cross validation employs a random approach instead. When this evaluation process finishes, various statistics can be derived for each class from comparison of the classifier results with the 'true' class values:

true positive values Data in class k, classified correctly as within k.

true negative values Data not in class k, classified correctly as outwith k.

false positive values Data not in class k, classified incorrectly as within k.

false negative values Data in class k, classified incorrectly as outwith k.

Two important values can be calculated from these:

$$\text{Sensitivity: } \frac{TP}{TP+FN}\text{; Specificity: } \frac{TN}{FP+TN}$$

The former is maximised for a class if the classifier manages to correctly classify all those points whose true value is within the class; the latter is maximised if no points outside the class are misinterpreted as being within it. Often a classifier will be trained to produce a minimum sensitivity, and its specificity will be taken as the measure of efficacy.

A popular, publicly available collection of datasets is maintained by the University of California, Irvine: the UCI repository [3] contains both binary and multi-class data for use in machine learning. In the literature, comparisons between classification methods are often derived using sets from this suite as a benchmark. It includes, amongst others: Fisher's iris data, BUPA breast cancer data and an Aldermaston glass experiment. In §6.5 we will describe this in further detail and introduce a study linked to this thesis project based on a cross-validation procedure applied to the provided data.

2.2 Classification methods

2.2.1 Bayes' Theorem

The basis of most Bayesian statistical methods is that of *Bayes' theorem*:

$$\Pr(Y|X) = \frac{\Pr(X|Y) \cdot \Pr(Y)}{\Pr(X)}. \tag{2.1}$$

When applied to datasets, with X as a proposed model (parameters) and Y as the event of making a certain set of observations, this theorem states that "what we know now is proportional to the influence of our observations and what we knew before". Bayesian classification theory is based on the use of $\Pr(Y|X)$, the posterior probability of observing the data Y given that the model is X. $\Pr(Y)$,

the *a priori* probability of observing Y, must be set subjectively, but would be dominated by a sufficient mass of *evidence* $\Pr(X|Y)$, perhaps to the contrary. We may therefore think of Equation 2.1 as follows:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \tag{2.2}$$

$$\implies \text{posterior} \propto \text{prior} \times \text{likelihood}. \tag{2.3}$$

This approach towards probability involves using known data to arrive at a *maximum likelihood* estimate of what the parameter(s) driving the system were. Generation of a posterior *probability density function* (PDF) makes use of Bayes' Theorem, recasting it in the form:

$$P(\omega_k|v) = \frac{P(\omega_k)p(v|\omega_k)}{\sum_{i=1}^{K} P(\omega_i)p(v|\omega_i)}. \tag{2.4}$$

Here, the ω_k are the events corresponding to a testing datapoint being in class k, and v denoted the model information with which the classifier has been trained. The capital P refer to the a priori probabilities of ω_k, and the lowercase p refer to some measure of the degree of evidence linking this point to class k. The denominator here normalises over all possible classes; the reject class as mentioned above may be included here and made equal to a constant. This formula results in an *a posteriori* probability being assigned to the testing point's chance of being in class k.

2.2.2 Naïve Bayes classification

We can use Equation 2.4 to form a simple classification system to estimate the nature of testing data based on known training data X. Let there be K classes in a set of training data, and let $X^{(k)} = \{x_i^{(k)}\}_{i=1}^{N_k}, \sum N_k = N$ be training data

representing the class $k \in \{1 \dots K\}$; let y be our novel point and let the event ω_k read "y is in the class k"; we recast Equation 2.4 using Equation 2.3 as the following:

$$P(\omega_k|y, X) = P(\omega_k|X)P(y|\omega_k, X) \cdot \text{constant}. \tag{2.5}$$

Firstly, the constant term is simply the normalisation term in Equation 2.4 and may, for now, be discarded. The first term, $P(\omega_k|X)$, is the *prior* probability that *any* new datapoint will belong to the class k, without knowledge of the datapoint itself but knowing the training data X. We may have prior information on this; however, it is usually safe to take this as $\frac{N_k}{N}$. Estimating the term $P(y|\omega_k, X)$, the *likelihood* function of a realistic observation of y given that it is purported to be in class k and the relevant training information X, is more difficult and requires certain assumptions. If we let y (and thus x_i) to have dimension d, this term is the following:

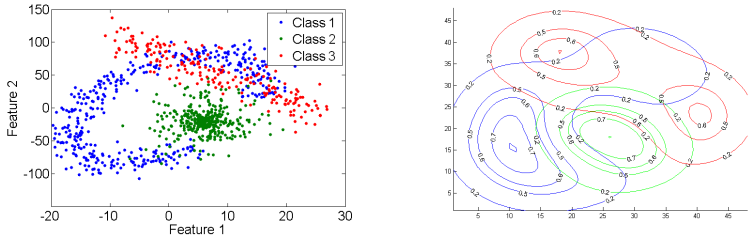
$$\begin{aligned} P(y|\omega_k, X) &= P(y \text{ a realistic member of } X^{(k)}) & (2.6)\\ &= P((y_1, y_2, \dots, y_d) \text{ a.r.m. } (X_1^{(k)}, X_2^{(k)}, \dots, X_d^{(k)})) & (2.7) \end{aligned}$$

We now make the *naïve independence assumption*; that the realistic membership of each y_j to the set $X_j^{(k)}$ are all independent. Thus we can recast these equations as:

$$\begin{aligned} P(y|\omega_k, X) &= \prod_{j=1}^{d} P(y_j \text{ a.r.m. } X_j^{(k)}) & (2.8)\\ &= \prod_{j=1}^{d} P(y_j|\omega_k, X) \text{ by notation} & (2.9)\\ \implies P(\omega_k|y, X) &\propto P(\omega_k|X) \prod_{j=1}^{d} P(y_j|\omega_k, X). & (2.10) \end{aligned}$$

All that remains is to choose the one-dimensional function $P(y_j|\omega_k, X)$ given the set $X^{(k)}$. A popular method is using a Gaussian function with mean equal to that of the $X^{(k)}$ and a known variance, $\frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-1}{2\sigma^2}|y_j - \mu_j^{(k)}|^2)$. This function can thus be computed over all classes, and then normalised according to Equation 2.4 to provide a valid *posterior PDF* (PPDF) for any given novel point (with perhaps a $K + 1^{th}$ reject class of constant probability added), with the highest probability of class membership being decided as the *maximum a posteriori* (MAP) class of choice.

2.2.3 Parzen windows



(a) Dataset (b) PPDF contours

Figure 2.1: Parzen windows: (a) the dataset; (b) the probability contours

A particularly useful derivative of the Bayes classifier is that of *Parzen windows* [60] [37], which uses similar techniques to those described above to estimate the distribution of (a class of) a dataset. Referring again to our training data X split into constituent classes, for each element $x_i^{(k)} \in X^{(k)}$ we perform the following operation:

$$f(\mathbf{y}; \mathbf{x}_i^{(k)}) = \frac{1}{\sqrt{2\pi\sigma_K^2}} \exp(\frac{-1}{2\sigma_K^2}\|\mathbf{y} - \mathbf{x}_i^{(k)}\|^2) \tag{2.11}$$

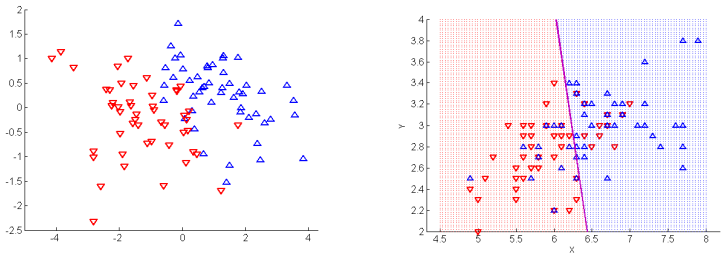
In principle, $f(\mathbf{y};\mathbf{x})$ (the Parzen *kernel*) could be any bounded function of two variables, but as before the Gaussian kernel is often used since it describes a probability-based proximity measure of the two vectors. Once a kernel has been created for each testing point relative to the known data of class k, the sum is taken and normalised to form a PPDF:

$$f(\mathbf{y};\mathbf{x}^{(k)}) = \frac{1}{N_k}\sum_{i=1}^{N_k} f(\mathbf{y};\mathbf{x}_i^{(k)}) \tag{2.12}$$

This PPDF is then provided priors and normalised over all classes in the same way as with naïve Bayes classifiers. As shown in diagram 2.1, it can generate elegant PPDF contours, but has the drawback that over multiple dimensions, instability may occur and computational time is a great deal more than with the naïve Bayes classifier, especially for large datasets. This method is often used effectively in lower-dimensional classification problems; for example, Eftestol et al. [21] use the Parzen windowing method to try and improve prediction of the occurrence of a cardiac arrest in medical patients given defibrillation readings. Improvements have also been made on this simple method, for example that due to Babich [4] who improves on the computational time needed to compute a Parzen window kernel classifier by a weighting process.

2.2.4 Principal component analysis (PCA)

In many situations where more than one feature is measured for each datapoint, the features chosen for measurement may be highly correlated. This may not be desirable, and a transformed dataset may be sought for which the features are statistically independent. This is known as *principal component analysis*, or the *Karhunen-Loève (KL) transform*. Conceptually, it identifies the main set of or-



(a) PCA (b) LDA

Figure 2.2: Alignment: (a) PCA; (b) LDA

thogonal axes over which the dataset is distributed, producing a small set of coordinates representing the data well, and which themselves are uncorrelated - and hence more useful as parameters. To perform PCA, we require the covariance matrix of the centred data to be diagonal. Define the *centred dataset* thus:

$$\hat{X} = \{\hat{X}_i\} = (X_i - \mu^X), \text{ where } \mu^X = \{\mu_j^X\} = \left\{ \frac{1}{N} \sum_{i=1}^{N} X_{ij} \right\}_{j=1}^{d} \tag{2.13}$$

The *covariance matrix* is then $C = \hat{X}\hat{X}^T$. This array captures the relation between the features of the data, and must be diagonalised to ensure non-correlation: thus we solve:

$$\text{find } V, \Lambda \text{ such that } V^{-1}CV = \Lambda, \Lambda \text{ diagonal} \tag{2.14}$$

This is equivalent to the eigenvalue problem $\lambda v = Cv$, which can be solved for a set of *eigenvalues* λ_k (the diagonal values of Λ), sorted by magnitude, and corresponding *eigenvectors* v_k (the columns of V). At this stage, only those columns of V, Λ and rows of Λ with nonzero λ_k are retained. In order to normalise the eigenvectors and thus maintain the distance preserving nature of the transformation, the

condition is introduced that:

$$\lambda_k(\hat{v}_k \cdot \hat{v}_k) = 1 \Longrightarrow \hat{V} = \{\hat{v}_k\} = \frac{v_k}{\sqrt{\lambda_k}} \tag{2.15}$$

The transformed data can now be expressed as:

$$X^*_{ij} = \sum_{k=1}^{N} C_{ik}\hat{V}_{kj} \Longrightarrow X^* = C\hat{V}. \tag{2.16}$$

Novel data Y can, in turn, be processed thus:

$$Y \rightarrow Y - \mu^X = \hat{Y} \rightarrow \hat{Y}\hat{X}^T = C^Y \rightarrow C^Y\hat{V} = Y^*. \tag{2.17}$$

That is, centring is performed with respect to the original data X. If the λ_k are sorted in the process such that λ_1 is greatest, this process produces an orthonormal transformation that re-aligns the axes about the dataset in such a way that the new first axis (known as the first *principal component* of the data) experiences the greatest variance within the dataset, and subsequent axes successively less variance.

In practice, the formulae above will often not be used as shown, since the main aim of PCA is usually one of *dimension reduction*, to alleviate the problem as stated above of many highly-correlated features. This process is now easily achieved since we have optimally realigned the data: a simple projection of data onto the first $l < d$ principal components is computed by replacing V with a matrix made up of those columns k for which $\lambda_k > \epsilon$, with ϵ some tolerance parameter. The more columns kept, the more true to life the projected results, but also the more difficult and/or unstable the problem will be of accurate classification - this is termed the *Curse of Dimensionality*. For example, PCA is useful in "identikit"-style projection of faces [88] onto their basic features, and hence in distinguishing

"obviously" different face shapes, colours, layouts, etc. from each other, but is not always the most efficient method for more subtle feature extraction (e.g. scars).

2.2.5 Linear discriminant analysis

Although PCA is suitable for use in a situation where class information is not present or unimportant, an alignment system may instead be required to rearrange the data to maximise separation between known classes. Here, as demonstrated by Zhuang and Dai [89], PCA may no longer be helpful. Linear discriminant analysis (LDA) [26], also often used in image-classification procedures [20], addresses this problem by instead of looking at the variance of each class over the whole dataset, substituting this for a comparison between two quantities, calculated for each class: the between-class scatter S_b and within-class scatter S_w, defined thus:

$$S_b = \frac{1}{N}\sum_{k=1}^{K} N_k \|\mu_k - \mu\|^2, \tag{2.18}$$

$$S_w = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} [X_i^{(k)} - N_k]. \tag{2.19}$$

Here K is the number of classes, N_k is the number of datapoints in each class, μ_k denote class-specific mean vectors and $X_i^{(k)}$ denotes the i^{th} datapoint belonging to the k^{th} class of data. These equations have the effect of comparing the "directions" of each class with each other; computing the eigenvectors for this transformation then involves minimising the ratio of these two quantities, namely the Fisher criterion:

$$W_{LDA} = \arg\max \frac{|W^T S_b W|}{|W^T S_w W|}.$$

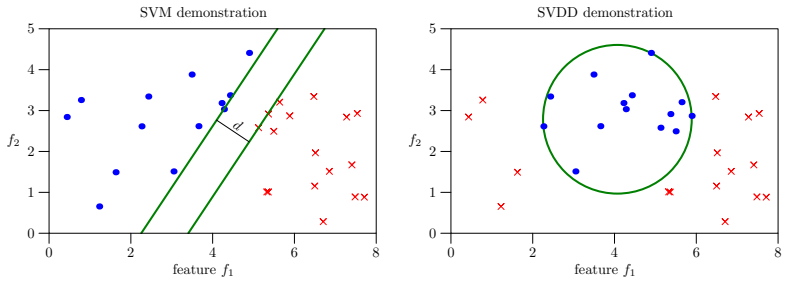
This quantity is undefined if S_w is singular, which often occurs when the dimension of each datapoint is far higher than the number of training points (e.g. face

recognition): this is known as the *small sample size* problem. Thus a combination of PCA followed by LDA is often used in applications, so dimensionality of a problem is reduced beforehand (see e.g. [32]). Martinez et al. [53] show that the choice of which method to use is sometimes complex, and that LDA is not necessarily more effective despite its incorporation of class-specific information. Belhumeur [6] also alludes to this in a study of face recognition. In its raw form, LDA handles binary-class data by design; however, studies such as that of Ma [49] show that its use can occasionally be extended to multi-class datasets. In certain situations, it can be adapted to kernel spaces (see §2.3), as shown by Mika [56], although this paper shows also that this method can lead to an ill-posed problem.

Local hyperplane classification Widely used in classification literature, with examples due to Chen and Liu [15], Mika [57], Tao Yang [87] and Jian Yang [86], localised Fisher discriminant analysis is a popular method for combating the non-singular form often encountered when performing discriminant analysis in a kernel space. These papers broadly proposed localised methods of separation of data with binary class information, to render tractable the problem of kernel discriminant analysis at a smaller scale.

2.2.6 Support vector machines

Originally proposed in its original form by Vapnik [80], a widely-used method of binary classification, the *support vector machine*, involves constructing a maximally separating hyperplane between data distributed in two distinct subspaces. The full derivation of the SVM can be found in the book by Schölkopf and Smola [69]; we give a brief derivation here. Consider the geometric implication



(a) SVM (b) SVDD

Figure 2.3: Demonstration of (a) a support vector machine; (b) support vector domain description.

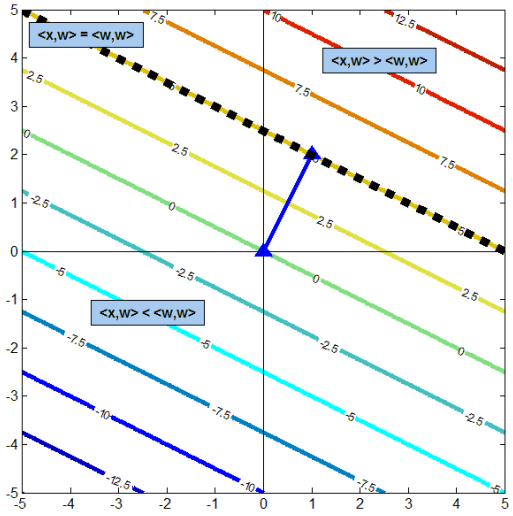


Figure 2.4: Inner product contours

of the set of points in an inner product space V, defined as follows:

$$\mathcal{H}(\mathbf{w}) = \{\mathbf{x} \in V : \langle \mathbf{w}, \mathbf{x} \rangle - b = 0\} \tag{2.20}$$

Figure 2.4 shows an example of this in $\mathbb{R}^2$: the contours for this function describe a set of parallel lines, with the particular path describing those points for which $\langle \mathbf{w}, \mathbf{x} \rangle \equiv \langle \mathbf{w}, \mathbf{w} \rangle$ passing through the point $\mathbf{w}$ and being perpendicular to the vector $(O\mathbf{w})$. Thus, equation 2.20 is the general equation in an inner product space of a *hyperplane*: that is, if the dimension of the vector space is d, it describes a $(d-1)$-dimensional surface through the space, separating points according to the side of the boundary on which they fall, and thus inducing a classification function as follows:

$$f_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b). \tag{2.21}$$

The purpose of the scalar b should be obvious from Figure 2.4: it acts as a scalar according to which of the parallel lines characterised by our choice of $\mathbf{w}$ we wish to assign the boundary. Choosing $b \equiv \langle \mathbf{w}, \mathbf{w} \rangle$ clearly recovers the hyperplane passing through the point $\mathbf{w}$ itself. However, note that this could also be achieved by scaling $\mathbf{w}$ by a constant K, and accordingly b by K^2. We thus have a redundant degree of freedom, which is corrected by adding a further requirement. The *canonical* form of the hyperplane $\mathcal{H}$ defined in Equation 2.20 is one of the two unique planes which satisfy:

$$\min_i |\langle \mathbf{w}, x_i \rangle| = 1. \tag{2.22}$$

This requirement is equivalent to demanding that the point closest to the hyperplane has a distance of $1/\|\mathbf{w}\|$. The purpose of a *support vector machine* classification is as follows: Given a dataset X, to find the optimal hyperplane, which will

separate two classes of data on either side with maximal margin [81]. Should this margin be found subject to the canonical constraints given in Equation 2.22, it is clear that, since it will fall between datapoints on either side (see Figure 2.3(a)) with margin $1/\|\mathbf{w}\|$, we have that the total margin is $\rho = 2/\|\mathbf{w}\|$; to be clear, this is the quantity we wish to *maximise*. Thus, values of $\mathbf{w}$ which *minimise* the quantity $\|\mathbf{w}\|$ will be advantageous. If we assume that the partitions of the data which are thus classified will be those values on one side with labels $y_i \equiv 1$ and those on the other with $y_i \equiv -1$, as a result of Equation 2.22 we have the following formulation:

$$\begin{aligned} \mathbf{w} \cdot x_i + b \geq 1 \qquad & (y_i = 1), \\ \mathbf{w} \cdot x_i + b \leq 1 \qquad & (y_i = -1) \\ \implies y_i(\mathbf{w} \cdot x_i + b) \geq 1. & \end{aligned} \tag{2.23}$$

By solving an optimisation problem, the SVM can find a decision boundary such that the proportion of data in each class on each side of the boundary is maximised. The method was later adapted by Cortes and Vapnik [17] to include kernel methods (discussed in §2.3), making the process more flexible when the boundary between the classes is non-linear. In the linear formulation, the separating boundary may be derived by solving the following optimisation problem:

$$(\mathbf{w}, \xi, b) = \arg\min_{\mathbf{w},\xi,b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \tag{2.24}$$

$$\text{such that } y_i(\mathbf{w} \cdot x_i + b) \geq 1 - \xi_i \forall i, \xi_i \geq 0, \tag{2.25}$$

where here we note that minimising $\|\mathbf{w}\|$ is equivalent to the (easier) problem of minimising $\frac{1}{2}\|\mathbf{w}\|^2$. Here, $y_i \in \{-1, 1\}$ is the binary class information, $C \in [0, 1]$ is the parameter which allows for a *soft margin* in the case that not all data may

be separated linearly, thus allowing some data to be classified on the opposite side to the majority of its respective class, if this solution ends up more optimal; ξ_i are the corresponding error terms that allow this soft margin to be implemented. The choice of C is usually chosen on the particular basis of the problem in question; however, various papers have attempted to address the problem for choosing the optimal value of this pay-off parameter in an SVM context, most notably that of Hastie [33] and, in the context of SVDD as we shall introduce below, the paper due to Sjöstrand et al. [71].

2.2.7 Support vector domain description

Tax and Duin [77] proposed a variant on the standard support vector machine, the *domain description* (also known as *one-class SVM*), dealing with the problem of providing a compact spherical description of (one class of) a dataset. This method is useful where only status of presence in a well-defined region is necessary; it has also been used in hybrid multi-class classification methods (see below). The formulation is somewhat similar to standard SVM; however, instead of producing a hyperplane boundary, a sphere is fitted around the data:

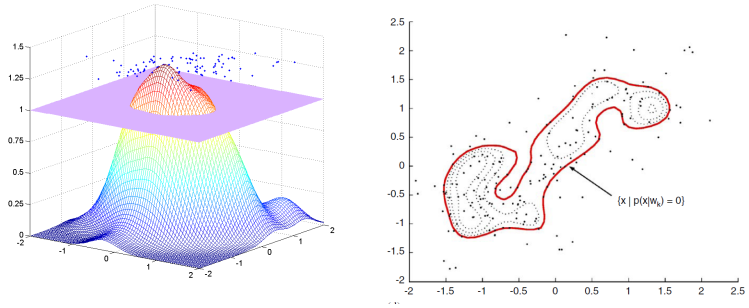
$$(\mathbf{a}, \xi, R) = \arg\min_{\mathbf{a},\xi,R} R^2 + C\sum_{i=1}^{N} \xi_i \tag{2.26}$$

$$\text{such that } \|x_i - \mathbf{a}\|^2 \leq R^2 + \xi_i \forall i, \xi_i \geq 0. \tag{2.27}$$

The ξ_i terms here again have the effect of providing a soft-margin classifier, with C the parameter governing the penalty on making some of these terms nonzero, thus allowing some data outside the sphere if there exists a more optimal solution this way. Note the case $C = 1$ is known as a *hard margin* classifier and is equivalent to not allowing the ξ_i terms to be non-zero.

In §2.4, we shall see how both these formulations can be solved using a *quadratic program* (see §2.4.5) and how the solution depends entirely on the position of those data nearest the boundary itself, making the methods desirably insensitive to outliers. Note also that the solutions above do not involve the data x_i explicitly; they may be arrived at using only the inner products between data values, with no necessity to include the data itself in the calculation. Both of these concepts are important in the use of *kernel spaces*.

2.2.8 Domain described SVC



(a) DSVC PPDF (b) DSVC contour

Figure 2.5: Domain described SVC in the linear case

Lee and Lee [44] describe a system, the domain described support vector classifier, that uses ideas from the Parzen window classifier (§2.2.3) and the SVDD (§2.2.7) to create a powerful multi-class classification machine with the information gleaned from using the latter to process each class in turn. A similar process to achieve the same aim is also described in Kang and Choi [38]. Lee and Lee process each class independently via SVDD and construct the following function to simulate a Parzen window classifier based on distances from the sphere centres

(cf. Equation 2.12):

$$p(y; X^{(k)}) = \frac{1}{2}(R_k - f_k(y)) \tag{2.28}$$

$$\text{where } f_k(y) = 1 - 2\sum_i \alpha_i^{(k)} \exp(-q\|y - x_i^{(k)}\|^2) \tag{2.29}$$

$$+ \sum_{i,j} \alpha_i^{(k)} \alpha_j^{(k)} \exp(-q\|x_i^{(k)} - x_j^{(k)}\|^2).$$

where here R_k (see §2.4.4) and $\alpha_i^{(k)}$ refer to the solutions of an SVDD problem on the constituent classes. This function is then used for classification according to Bayes' theorem as a likelihood function, with the posterior being calculated to within a multiplicative constant (evidence) for each novel point and each class by adding a prior and taking the maximum over all classes:

$$k = \arg\max_{k=1\ldots K} \frac{N_k}{N}(R_k - f_k(y)). \tag{2.30}$$

We return later in §2.4.3 to this method of classification, as well as showing how to derive R_k, in the context of kernel spaces.

2.2.9 Using multiple classifiers

Many classification methods are based around the concept of a dataset with only two classes. The question naturally arises, therefore, as to ways of dealing with multi-class data, where it is unrealistic to rearrange into two classes. A method often used is to aggregate the results of several two-class classifiers, novel data then being classified according to some (possibly weighted) pre-defined decision rule once results have been obtained from the two-class combinations. Two main ways of achieving this are known as *one-against-one* and *one-against-all*. One-against-one classification works on a dataset involving K classes and applies $\frac{K(K-1)}{2}$ two-class classifiers according to the following method:

- Split the dataset X up into K constituent classes, $\{X_k\}_{k=1}^{K}$
- For $k_1 = 2 \dots K$:
- For $k_2 = 1 \dots k_1$:
- Create a classifier C_{k_1,k_2} to distinguish between X_{k_1}, X_{k_2}
- Increment k, and repeat the loop.

We therefore have $(K-1)$ classifiers referring to each constituent class of the dataset, and thus for a novel datapoint, to consider its candidacy in each class we must weight the results of these $K-1$ classifiers. Clearly, the way in which this is done is important as it affects the final decision. It will also be affected by the form of output involved in the binary classification step: e.g. using an SVM will produce $K-1$ decision boundaries for each class.

Since this method produces so many different results, which may be highly contradictory, the true nature of a novel point is often unclear. *One-against-all* classification uses a slightly different method to reduce the number of binary classifiers:

- For $k = 1 \dots K$:
- Split X into two parts: X_k (entries of class k), Y_k (otherwise)
- Create a classifier C_k to distinguish between X_k and Y_k
- Increment k, and repeat the loop.

This method produces K separate binary classifiers, but has the disadvantage that in the case where classes describe very different properties of a dataset, it will

not always be reasonable to combine them for a binary classifier to analyse, and thus the results may lack in rigour what they gain in simplicity. Other methods which can be used to combine these separate 'weak' classifiers into a more robust machine include boosting [25] and bagging (bootstrap aggregating).

2.3 Kernel spaces

When we classify data, the appropriate method we use - whether naïve Bayesian posterior-PDF analysis, SVM, SVDD or other methods - will often depend on the structure inherent in the dataset. For example, a multiclass dataset with well-defined clusters may require little more than a simple PPDF classification machine, whereas two-class data with a complex boundary of separation may require a support vector based method.

In this section, we will introduce the concept of a kernel space. This concept may be applied where a linear standard method fails to identify classes within a dataset with a sufficiently descriptive boundary, producing too many classification errors. Its main purpose is to map datapoints into a higher-dimensional superspace, within which a linear classification system may be applied to produce more desirable results. We will also show how, due to the structure of linear formulations of classifiers, the explicit computation of this mapping is not necessary, thus providing a tool for exploring mappings of data into very high, even infinite, dimensional superspaces. Furthermore, we will show that despite not having an explicit mapping, functions such as mean and variance can be standardised over the feature space, allowing for more stable results.

2.3.1 Definitions

Recall the definition of a dataset X as given in §2.1. The *input space* is the vector space in which all members of this dataset, $x_i \in X$, reside. For convenience we shall assume elements of X are real and have d features each; we thus refer to the input space as $\mathbb{R}^d$.

Kernel spaces Define a mapping as follows:

$$\phi : \mathbb{R}^d \to \mathbb{R}^D$$

Usually we have $D > d$ – also, D is not required to be finite, the only condition being that $\forall x \in \mathbb{R}^d, \|\phi(x)\|$ is finite. The mapping ϕ is not defined explicitly; rather, it is defined implicitly to conform to the following relation:

$$\exists k(x, y) \text{ such that } \forall x, y \in \mathbb{R}^d, k(x, y) = \phi(x) \cdot \phi(y). \tag{2.31}$$

If ϕ satisfies Equation 2.31, then k is known as a *kernel function*; we refer to ϕ as its respective *kernel mapping* and its domain $\mathbb{R}^D$ is known as a *reproducing kernel Hilbert space* (RKHS). Hereafter, we refer to this domain simply as *kernel space*. More generally we shall refer to the space in which algorithms are applied as *feature space*, to distinguish it in the case that kernel space is the same as input space (hereafter referred to as the *linear case*). Some examples of valid kernel functions $k(x, y)$ are given below.

- Linear kernel $k(x, y) = x \cdot y$
- Quadratic kernel $k(x, y) = (1 + x \cdot y)^2$
- Polynomial kernel $k(x, y) = (1 + x \cdot y)^p$ for some p

- Hyperbolic tangent (sigmoid) kernel $k(x, y) = \tanh(\kappa(x \cdot y) + \Theta)$
- Radial basis function (RBF) kernel $k(x, y) = \exp(-\|x - y\|^2/2\sigma^2)$

Note that these functions do not tell us the structure of the underlying map ϕ: indeed, in the case of RBF the explicit mapping is notoriously intractable [74]. Smola et al [73] provide some insight as to the connection between Green's functions on regularization operators and support vector kernels; for now we simply assume that our kernels have the above forms and we operate from a viewpoint of not requiring the explicit form to be known, or 'simple'. Other situations may also be identified where the exact choice of kernel will not be known *a priori*, but instead will be due to the result of a statistical training procedure; examples of this are due to Weinberger et al. [83], Lanckriet [42] and de la Torre and Vinyals [79]. Chen [15] instead uses an optimisation procedure based on the Fisher criterion similar to that used in LDA §2.2.5. In some cases, however – notably that of the quadratic kernel – it is possible to visualize the explicit mapping ϕ as well. For example, let $d = 3$ so $x = (x_1, x_2, x_3) \in \mathbb{R}^3$, and consider the following mapping:

$$\phi : (x_1, x_2, x_3) \rightarrow (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

It is straightforward to verify that if $k(x, y) = (1 + x \cdot y)^2$, then condition 2.31 is satisfied. This example also serves to show that ϕ need not be unique: if we replace the final three elements of $\phi(x)$ with $(x_1x_2, x_1x_3, x_2x_1, x_2x_3, x_3x_1, x_3x_2)$ the same result is achieved.

2.3.2 The kernel trick

Fortunately, the explicit computation of ϕ is rendered unnecessary if condition 2.31 holds; as we shall see, knowing ϕ exists is sufficient. *Mercer's theorem*,

mentioned by Schölkopf [68] gives a simple condition for a valid mapping ϕ to exist: that of *positive definiteness*, or equivalently, non-negative eigenvalues of the following matrix:

$$K = \{K_{ij}\} = \{\phi(x_i) \cdot \phi(x_j)\} = \{k(x_i, x_j)\} \tag{2.32}$$

The matrix K is known as the *kernel matrix* of the mapping and plays a crucial part in the analysis of the domain $\mathbb{R}^D$. Indeed, if we recall the method of PCA introduced in §2.2.4, we can identify the number of non-infinitesimal eigenvalues of the matrix K as being the number of meaningful principal axes on which a PCA method would project data from $\mathbb{R}^D$. In general, with a well chosen kernel this will be greater than the dimension of the original data d, thus motivating the use of the space ϕ; the use of kernel methods allow deeper patterns within the distribution of a dataset to be studied. The *kernel trick*, as popularly known in literature, is the concept that in certain cases, knowledge of K (and the existence of ϕ) can allow us to study classification methods applied to the *projected* dataset $\phi(X)$, without having to compute ϕ explicitly. Note that the matrix K, while not giving the explicit values of each $\phi(x)$, does give values of all the inner products $\{\phi(x_i) \cdot \phi(x_j)\}$, and therefore if a classification method can be formulated in a way such that values of datapoints are only mentioned implicitly through their dot products with each other, then this data can also be analysed in a kernel space $\mathbb{R}^D = \phi(\mathbb{R}^d)$ characterised uniquely by its kernel function k satisfying Equation 2.32.

2.3.3 Kernel centring

Suppose we have a simple, explicit mapping $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^4$, with $\phi(x_1, x_2) = (1, x_1, x_2, x_1x_2)$[1], and suppose that we have a dataset $X = \{x_i \in \mathbb{R}^2\}$ which we have *standardised*; i.e. we have $\mu_j = 0, \sigma_j = 1 \forall j \in \{1, 2\}$. Clearly, the projected data $\phi(X)$ does not have this property. In general, we will not know ϕ; however, standardisation is still possible. Schölkopf [68] notes the following relation: If

$$X = \{x_i\}_{i=1}^{N}, K = \{K_{ij}\} = \{k(x_i, x_j)\},$$

then the *centred* kernel K^{cen}, equivalent to ensuring that $\mu_j = 0 \forall j \in \{1 \dots D\}$ in kernel space, is given by:

$$K^{cen} = K - OK - KO + OKO, \tag{2.33}$$

$$\text{where } O \in \mathbb{R}^{n \times n}, O_{ij} \equiv 1/n. \tag{2.34}$$

Similarly, if we now wish to map novel data Y into this centred kernel space, we compute $U = \{U_{ij}\}_{(i,j)=(1,1)}^{(m,n)} = \{k(y_i, x_j)\}$, and apply the following:

$$U^{cen(K)} = U - O^*K - UO + O^*KO \tag{2.35}$$

$$\text{where } O \in \mathbb{R}^{m \times n}, O_{ij} \equiv 1/n. \tag{2.36}$$

To see this process in action with the given mapping function, consider two vectors x and y. The kernel matrix K is therefore of dimension (2×2) and has the following entries:

$$K_{xx} = 1 + x_1^2 + x_2^2 + x_1^2x_2^2, \tag{2.37}$$

$$K_{yy} = 1 + y_1^2 + y_2^2 + y_1^2y_2^2, \tag{2.38}$$

$$K_{xy} = K_{yx} = 1 + x_1y_1 + x_2y_2 + x_1y_1x_2y_2. \tag{2.39}$$

[1] Note this is a valid kernel as we know ϕ, thus k trivially exists.

We see that O is a (2×2) matrix with entries $\frac{1}{2}$, and thus:

$$OK = \frac{1}{2}\begin{pmatrix} K_{xx} + K_{xy} & K_{xy} + K_{yy} \\ K_{xx} + K_{xy} & K_{xy} + K_{yy} \end{pmatrix}, \tag{2.40}$$

$$KO = \frac{1}{2}\begin{pmatrix} K_{xx} + K_{xy} & K_{xx} + K_{xy} \\ K_{xy} + K_{yy} & K_{xy} + K_{yy} \end{pmatrix}, \tag{2.41}$$

$$OKO = \frac{1}{4}\{K_{xx} + K_{yy} + 2K_{xy}\}. \tag{2.42}$$

If we let $x = (2, 3), y = (5, 7)$, we have:

$$\phi(X) = \begin{pmatrix} 1 & 2 & 3 & 6 \\ 1 & 5 & 7 & 35 \end{pmatrix} \Longrightarrow \phi^{(c)}(X) = \begin{pmatrix} 0 & -1.5 & -2 & -14.5 \\ 0 & 1.5 & 2 & 14.5 \end{pmatrix} \tag{2.43}$$

The kernel matrices are:

$$K = \begin{pmatrix} 50 & 242 \\ 242 & 1300 \end{pmatrix}, K^{(c)} = 216.5 \begin{pmatrix} +1 & -1 \\ -1 & +1 \end{pmatrix}. \tag{2.44}$$

Following the above equations yields:

$$OK = (KO)^T = \begin{pmatrix} 146 & 771 \\ 146 & 771 \end{pmatrix}, \{OKO\}_{ij} = 458.5, \tag{2.45}$$

yielding the required centred kernel as described above.

2.3.4 Kernel standardisation

Kernel PCA We now have all the tools necessary to perform principal component analysis (§2.2.4) in a kernel space; just as the covariance matrix C was the inner product matrix of the centred dataset given in the matrix $\hat{X}$, $C = \hat{X}\hat{X}^T$, we now replace C with K^{cen} as described in this section:

$$\text{find } V, \Lambda \text{ such that } V^{-1}K^{cen}V = \Lambda, \Lambda \text{ diagonal} \tag{2.46}$$

It is then a simple case of following Equations 2.15 and 2.16 to find the principal components of a dataset in kernel space, and Equation 2.17 (using U in place of C^Y) to map onto these axes a set of given novel data:

$$Y \to k(Y, X) = U \to \text{centred} = U^{cen(K)} \to U^{cen(K)}\hat{V} = Y^*. \tag{2.47}$$

Kernel whitening The method of PCA, and its kernel equivalent, ensure a distance preserving rotation of axes in feature space which aligns directions in a dataset with greatest variance up against the principal axis, with subsequent directions of decreasing variance against each subsequent axis in turn. However, this may not be our aim: in order to perform a technique such as support vector description(§2.2.7), in both linear and kernel feature space, it is useful to map data to principal axes with *equal* variance, i.e. perform some meaningful standardisation. This cannot be done in the general case with kernels, as the concept is somewhat meaningless in cases where one implicit dimension may intrinsically have zero variance (consider the valid kernels of the form $\phi : (x_1, x_2) \to (1, \cdots)$ above); however, the method of *kernel whitening*, introduced by Tax and Juszczak [78], achieves a similar aim to data projected with KPCA. Recall Equation 2.15; this condition was introduced to ensure the transformation merely rotated the data (i.e. it preserved distances). Replacing this condition by:

$$\lambda_k^2(\hat{v}_k \cdot \hat{v}_k) = 1 \Longrightarrow \hat{V} = \{\hat{v}_k\} = \frac{v_k\sqrt{N-1}}{\lambda_k} \tag{2.48}$$

ensures instead that when we reconstruct the data onto the principal axes in the same fashion as Equation 2.16, we achieve transformed data X_W with zero mean and unit variance. Creating the requisite kernel matrix is achieved with $K_W =$

$X_W X_W^T$, and mapping novel data to these axes is done according to Equation 2.35 as per normal.

2.3.5 Kernel SV methods

As a consequence of the fact that the objective functions for both the SVM method (§2.2.6) and the SVDD method (§2.2.7) are reliant only on the values of the inner products of data points, as opposed to using the explicit values of the data itself, they can be adapted for use in kernel spaces. A comprehensive review of kernel-based learning algorithms can be found in Muller et al. [58]. For SVM, the kernel analogue of Equations 2.24 and 2.25 is as follows:

$$\begin{aligned}(\mathbf{w}, \xi, b) &= \arg\min_{\mathbf{w},\xi,b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{such that } y_i(\mathbf{w}\cdot\phi(x_i)+b) &\geq 1-\xi_i \forall i, \xi_i \geq 0. \end{aligned} \tag{2.49}$$

The kernel formulation for SVDD (Equations 2.26 and 2.27) is:

$$\begin{aligned}(\mathbf{a}, \xi, R) &= \arg\min_{\mathbf{a},\xi,R} R^2 + C\sum_{i=1}^{N}\xi_i \\ \text{such that } \|\phi(x_i)-\mathbf{a}\|^2 &\leq R^2+\xi_i \forall i, \xi_i \geq 0. \end{aligned} \tag{2.50}$$

Note the subtle differences between the linear-space and kernel-space formulations: the only changes are that x_i have been replaced by their mapped equivalents $\phi(x_i)$. In the next section, we use these general forms to show that quadratic programming can help solve the formulations in the more general, kernel case. The linear case is a special case of this where $\phi(x) = x$ and $k(x, y) = x \cdot y$.

2.4 Solving optimisation problems

It is possible to greatly simplify the procedure of solving Equations 2.49 and 2.50 by the method of Lagrange multipliers. We will show that the solutions have a simple representation as a weighting vector and ultimately depend only on a few points in the dataset, thus making computations with the solution possible even for large datasets.

Lagrangian multipliers The *strong Lagrangian principle* states that if we wish to minimise a function $f(\mathbf{x})$ (called the *primal* objective) subject to the constraints $g_i(\mathbf{x}) \leq 0$, $g_j^=(\mathbf{x}) = 0$, we may construct the following function:

$$\Lambda(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum_{i=1}^{N_1} \alpha_i g_i(\mathbf{x}) + \sum_{j=1}^{N_2} \beta_j g_j^=(\mathbf{x}) \tag{2.51}$$

with the property that the minimum will occur at a point where $\nabla_x \Lambda = 0$; furthermore, the *Wolfe dual problem* [28] states that if we can successfully *maximise* Λ over all $\mathbf{x}, \alpha \geq 0$ we have an equivalent problem (the *dual* objective) and thus this (hopefully less intractable) $\mathbf{x}$ will lie at the same point. We now apply this method to SVM and SVDD.

2.4.1 SVM

Suppose we wish to perform the equivalent of a linear SVM on a dataset $X \in \mathbb{R}^{N \times d}$ that has been mapped to another space according to some function $\phi : \mathbb{R}^d \to \mathbb{R}^D$ such that $\exists k(x, z) \forall x, z \in \mathbb{R}^d, k(x, z) = \phi(x) \cdot \phi(z)$. Let $\{y_i\} \in \{-1, 1\}$ be the relevant binary class information. The constraints in Equation 2.49 can be

rearranged thus:

$$\begin{aligned} g_i, h_i &\leq 0 \\ \text{where } g_i(\mathbf{w}, \xi, b) &= 1 - \xi_i - y_i(b + \mathbf{w} \cdot \phi(\xi_i)), \quad (2.52) \\ h_i(\mathbf{w}, \xi, b) &= -\xi_i. \quad (2.53) \end{aligned}$$

Thus the Lagrangian for SVM (with $\alpha_i, \beta_i \geq 0 \forall i$) is as follows:

$$\Lambda(\mathbf{w}, \xi, b, \alpha, \beta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i\left[1 - \xi_i - y_i(b + \mathbf{w} \cdot \phi(\xi_i))\right] - \sum_{i=1}^{N}\beta_i\xi_i. \quad (2.54)$$

Taking derivatives with respect to the primal variables $(\mathbf{w}, \xi, b)$ we find:

$$\forall k \frac{\partial \Lambda}{\partial w_k} = 0 \implies \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \phi(x_i), \quad (2.55)$$

$$\frac{\partial \Lambda}{\partial b} = 0 \implies \sum_{i=1}^{N}\alpha_i y_i = 0, \quad (2.56)$$

$$\forall i \frac{\partial \Lambda}{\partial \xi_i} = 0 \implies C = \alpha_i + \beta_i. \quad (2.57)$$

Equation 2.57, together with the requirement that $\alpha_i \geq 0, \beta_i \geq 0$, implies $\alpha_i \in [0, C]$. Using these relations we find:

$$\|\mathbf{w}\|^2 = \left\|\sum_i \alpha_i y_i \phi(x_i)\right\|^2 = \sum_{i,j}\alpha_i\alpha_j Q_{ij} = \alpha^T Q \alpha \quad (2.58)$$

where $Q_{ij} = y_i y_j \phi(x_i)\phi(x_j) = y_i y_j k(x_i, x_j)$. Thus:

$$\alpha = \arg\max_{\alpha} \Lambda = \frac{1}{2}\alpha^T Q \alpha + \sum_i \alpha_i - \sum_i \alpha_i y_i (b + \mathbf{w} \cdot \phi(x_i)) \quad (2.59)$$

where the second term here uses $\alpha_i = C - \beta_i$. The third term, using relation 2.55, can be expanded:

$$\begin{aligned}\sum_i \alpha_i y_i (b + \mathbf{w} \cdot \phi(x_i)) &= b \sum_i \alpha_i y_i + \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) \\ &= 0 + \sum_{i,j} \alpha_i \alpha_j Q_{ij} \\ &= \alpha^T Q \alpha. \end{aligned} \tag{2.60}$$

Thus the solution can be expressed (with $L = -\Lambda$) as:

$$\alpha = \arg\min_\alpha L(\alpha) = \frac{1}{2}\alpha^T Q \alpha - \mathbf{e}^T \alpha \tag{2.61}$$

where $\mathbf{e}$ is a vector of all ones, under the conditions $y^t \alpha = 0$ (from Equation 2.56) and $\alpha_i \in [0, C] \forall i$. This is now in a general quadratic form (see §2.4.5), and can be solved using standard optimisation solvers. Furthermore, the Karush-Kuhn-Tucker (KKT) conditions (see p. 13 and Ch. 6 of [69]) ensure that under certain continuity conditions, this solution for α is globally optimal:

> **KKT conditions:** If, and only if, the functions $f : \mathbb{R}^d \to \mathbb{R}$ and $g_i : \mathbb{R}^d \to \mathbb{R}$ are continuously differentiable and convex and the functions $g_j^= : \mathbb{R}^d \to \mathbb{R}$ are affine, and a minimum is found (that is, a point with coefficients α, β satisfying $\nabla L = 0$ where L is defined as in Equation 2.51), then this point is a global minimum of the minimisation problem as stated.

Once an optimal α is known, the feature-space position relative to the margin can be computed for any novel point z to within the constant factor b by the following formula:

$$\delta_z = \sum_{i=1}^{N} \alpha_i y_i k(x_i, z) \tag{2.62}$$

In order to classify z on a side of the margin we need to know the value of the constant factor b. This is derived by consulting the margin distances of particular points, known as *support vectors*, described below.

2.4.2 SVDD

A similar approach, as introduced by Schölkopf et al [67], is used to analyse the SVDD objective function. The constraints in Equation 2.50 can be rearranged thus:

$$\begin{aligned} g_i, h_i &\leq 0 \\ \text{where } g_i(\mathbf{a}, \xi, R) &= \|\phi(x_i) - \mathbf{a}\|^2 - R^2 - \xi_i, \qquad (2.63) \\ h_i(\mathbf{a}, \xi, R) &= -\xi_i. \qquad (2.64) \end{aligned}$$

Thus the Lagrangian for SVDD is as follows:

$$\Lambda(\mathbf{a}, \xi, R, \alpha, \beta) = R^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[R^2 + \xi_i - \|\phi(x_i) - \mathbf{a}\|^2\right] - \sum_{i=1}^{N}\beta_i\xi_i. \qquad (2.65)$$

Taking derivatives yields:

$$\frac{\partial\Lambda}{\partial R} = 0 \implies \sum_{i=1}^{N}\alpha_i = 1, \qquad (2.66)$$

$$\forall k \frac{\partial\Lambda}{\partial a_k} = 0 \implies \mathbf{a} = \frac{\sum_{i=1}^{N}\alpha_i\phi(x_i)}{\sum_{i=1}^{N}\alpha_i} = \sum_{i=1}^{N}\alpha_i\phi(x_i), \qquad (2.67)$$

$$\forall i \frac{\partial\Lambda}{\partial\xi_i} = 0 \implies C = \alpha_i + \beta_i \implies \alpha_i \in [0, C]. \qquad (2.68)$$

Using Equation 2.68 and cancelling terms in $\sum\xi_i, \sum\alpha_i\xi_i$ we find:

$$\Lambda = R^2\left(1 - \sum_i \alpha_i\right) + \sum_i \alpha_i K_{ii} - 2\mathbf{a}\cdot\sum_i \alpha_i\phi(x_i) + \|\mathbf{a}\|^2\sum_i \alpha_i \qquad (2.69)$$

where $K_{ij} = \phi(x_i) \cdot \phi(x_j)$. Using Equations 2.66 and 2.67 allows simplification:

$$\Lambda = \sum_i \alpha_i K_{ii} - \sum_{i,j} \alpha_i \alpha_j K_{ij} \tag{2.70}$$

Finally, putting $\mathbf{d} = \text{diag}(K)$ and $L = -\Lambda$ as before, we have another solvable QP form:

$$\alpha = \arg\min_{\alpha} L(\alpha) = \alpha^T K \alpha - \mathbf{d}^T \alpha, \tag{2.71}$$

where $\mathbf{e}^T \alpha = 1$ (from Equation 2.66) and as before, $\alpha_i \in [0, C] \forall i$. To find the distance from the centre in feature space of a novel point denoted z, the following equation is used:

$$R_z^2 = k(z, z) - 2 \sum_i \alpha_i k(x_i, z) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j). \tag{2.72}$$

In §2.4.4 we shall discuss how this equation may be used to determine the effective radius of our enclosing sphere, and how it is related to the cost parameter C. Once the radius of the sphere is calculated, it is a simple matter of comparing it to the radii of novel points; similarly, the radii of the training points themselves have an inherent connection to the values taken by the coefficients α_i.

2.4.3 Domain described SVC

Recall the DDSVC method as described in §2.2.8, which allows a shape similar to a posterior PDF function - related to density of datapoints - to be cast over the support contour of a dataset in linear space, with the purpose of providing information as to the distribution of the dataset within the sphere described by the SVDD method; Lee and Lee show [44] that creating one of these for each class and comparing results across classes can result in elegant, accurate decision boundaries for

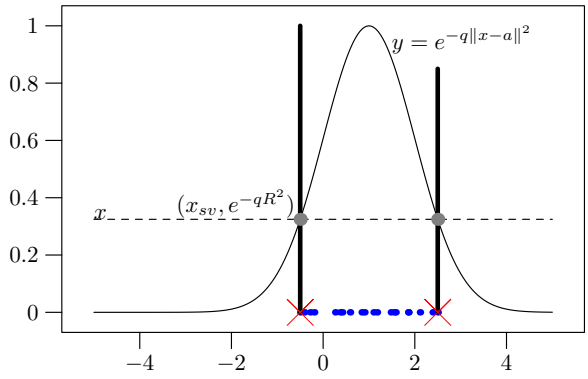


Figure 2.6: Domain-described SVC: adapted method ($a = 1, q = 1/2$)

novel data. Here we adapt this method to provide a kernel analogue, substituting kernel distance in feature space for Euclidean distance in linear space. For data belonging to each class k in turn, known as $X^{(k)}$ as before, consider the following function:

$$p(y; X^{(k)}) = \exp\left(-q\left\|\phi(y) - \mathbf{a}^{(k)}\right\|^2\right) - \exp\left(-qR_k^2\right) \quad (2.73)$$

$$\text{where } \left\|\phi(y) - \mathbf{a}^{(k)}\right\|^2 = k(y,y) - 2\sum_i \alpha_i^{(k)} k(y, x_i^{(k)}) + \sum_{i,j} \alpha_i^{(k)} \alpha_j^{(k)} k(x_i^{(k)}, x_j^{(k)}). \quad (2.74)$$

Figure 2.6 shows the motivation for this formula: support vector points, defining the feature-space distance R away from the centre point $\mathbf{a}$, will take the value e^{-qR^2} when the windowing function is used; setting this as the threshold then ensures that these points lie on the decision boundary and anything "inside" will have a closer distance in feature-space to the centre.

2.4.4 Support vectors

The reason for the term "support vector" in the names of these two classification methods comes from the phenomenon encountered in most cases when these quadratic programs are solved. In both cases, the coefficients α_i will split up into a trichotomy of classes:

- $\alpha_i = 0$: *inner* points
- $\alpha_i = C$: *outer* points
- $0 < \alpha_i < C$: *support vector* points

The relationship between these three classes is as follows: The solutions to the above QP problems will usually exhibit a small number ($o(1)$ in many cases) of support vectors, and the classification boundary will be continuously dependent on these points alone. In most simple cases, most points will be *inner* and have null coefficient. This corresponds to their error terms ξ_i, mentioned in the primal but not dual problems, not needing to be used. if $C < 1$ it may be the case that some points are outside the boundary; for these points, $\alpha_i = C$ and they also play no part in the classification boundary. This is equivalent in the primal formulation to their ξ_i coefficients being non-zero. In the SVDD case, for example, this will be the case if the optimal solution denotes that all points except these can be fit in a sphere around a certain point, such that this sphere's primal objective function (related to its radius) is smaller than if all points were included (thus all ξ_i terms were zero). In both SVM and SVDD cases, the boundary is *defined* as being the line on which only the *support vectors* lie.

SVM classification To calculate which side of the boundary lies a novel point z, we need to know the shift b used in the primal formulation. We calculate, using Equation 2.62, the shifted positions δ_i, relative to the margin, of those data for which $0 < \alpha_i < C$. By design of the SVM algorithm, it should be the case that all δ_i for which $y_i = -1$ are "separate" numerically from the values taken by those data with $y_i = 1$. For example, it may be the case that:

$$\max_{\{i:y_i=-1\}} \delta_i < \min_{\{i:y_i=1\}} \delta_i.$$

We can therefore take b to be in the centre to provide a decision boundary:

$$b = \frac{\sum_{i \in S_v} \alpha_i \delta_i}{\sum_{i \in S_v} \alpha_i} \tag{2.75}$$

We may now use the following equation to classify a novel point:

$$\omega_z = \text{sgn}\left(b + \sum_{i=1}^{N} \alpha_i y_i k(x_i, z) \right). \tag{2.76}$$

SVDD classification For the domain description, the process is largely similar. The optimal radius R is defined as being the feature-space distance (see Equation 2.72) between *any* support vector and the centre of the sphere. We thus use this equation to provide a decision boundary for novel data:

$$z \in S \iff R^2 \geq k(z,z) - 2\sum_i \alpha_i k(x_i, z) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j). \tag{2.77}$$

The fact that usually the number of support vectors is very low compared to the size of the dataset makes these methods particularly popular in simple classification machines due to the simple computation of a decision boundary of novel points which only needs to use the values of a handful of the training data.

2.4.5 Quadratic programming and LibSVM

LibSVM [13] is a package designed by Chih-Jen Lin et al. at National Taiwan University, for the express purpose of solving the optimisation problems as derived in §2.4.1 and §2.4.2. The observation made in this paper is that SVM, SVDD and various other formulations of problem similar to this can all be reduced to the general form of a *quadratic program*. This definition is given below.

Definition A *quadratic program* is an optimisation problem that takes the following form:

$$\mathbf{x} = \arg\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}, \tag{2.78}$$

$$\text{such that } A\mathbf{x} \leq \mathbf{b}, A^{=}\mathbf{x} = \mathbf{b}^{=}. \tag{2.79}$$

Here, $\mathbf{x}, \mathbf{c} \in \mathbb{R}^N$, $Q \in \mathbb{R}^{N\times N}$, $A, A^{=} \in \mathbb{R}^{M\times N}$, $\mathbf{b}, \mathbf{b}^{=} \in \mathbb{R}^M$. The name arises from the analogue it provides in $\mathbb{R}^d$ with solving a quadratic equation in one dimension: the Q term denotes how products between two values in the vector $\mathbf{x}$ contribute to the objective function, and c the linear terms in $\mathbf{x}$. Technical details of the solution methods given are provided in [22]; this paper, included as an accompaniment to the LibSVM package, describes an *active-set* method for solving the quadratic programming problems. In practice, we have usually used algorithms built into the MATLAB Optimization Toolbox [55] to solve each instance; these programs use a similar method and have proved adequate for the requirements of this study. A reference to LibSVM is included here as its active-set approach, though beyond the remit of this document, is a commonly-used approach for convex quadratic program solving.

Chapter 3

Dealing with missing data

3.1 Background

Until now we have considered only methods that operate on datasets such that, for all datapoints $\{X_i\}_{i=1}^N$ and all considered features $\{f_k\}_{k=1}^d$, we can identify a unique value (usually a real number) pertaining to the measurement taken of datapoint i with respect to feature k. We have called the relevant entry X_{ik} or similar to identify with an element in a matrix; consequently, it has been convenient to think of the full dataset X as being a "normal" matrix over $\mathbb{R}$, that is $X \in \mathbb{R}^{N\times d}$. We will henceforth refer to this situation as being the *full data case* (FDC). However, in many cases there will be *missing data* present in either training or testing datasets. If it is attempted to naïvely use the classification methods described above on such a dataset, usually nonsensical results will follow; it is thus important to have a knowledge of techniques in dealing with missing data. Comprehensive reviews of statistical methods commonly used in the analysis of datasets in the presence of missing data include the book by Schafer [65] and that

of Little and Rubin [47]. Finally, for completeness, some statistical methods are used to analyse the case where data is not missing, but rather *uncertain*, such as the study due to Bi and Zhang [9] which provides an SVM style classifier in this case.

3.1.1 Notation

In the following chapter we shall make use of the following notation. As before, we define the dataset $X \in \mathbb{R}^{N\times d} = \{x_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}^d$, as our dataset, and we introduce the idea of a *presence matrix* $\Pi = \{\pi_{ij}\}$ for $i \in \{1 \dots N\}$, $j \in \{1 \dots d\}$, possessing binary values $\pi_{ij} \in \{0, 1\}$. This matrix will denote the presence $\pi_{ij} = 1$ or missingness $\pi_{ij} = 0$ of the value X_{ij} denoting the j^{th} feature of the datapoint x_i. We hereafter use the algebraic notation $\mathbb{Z}_2$ to denote $\{0, 1\}$, whence $\Pi \in \mathbb{Z}_2^{(N\times d)}$.

3.1.2 Models

In order to perform statistical analysis with missing data, we need to be sure of the data framework in which we are working. To this end, it is useful to define an underlying statistical *model*, assumed to be a probability distribution with certain parameters, driving the sampling of the data and the missingness pattern. Care must be taken when defining the most general case, since there will most likely be elements of dependency between the values of the dataset and their presence status, thus precluding the analysis of the dataset X and presence matrix Π separately. For example, consider a survey in which respondents are encouraged to provide their annual salary, or else denote a lack of response; clearly there is po-

tential in this situation for the probability of non-response to vary according to the underlying values which would have been observed (for example, those on very low or very high incomes may be more reticent to declare this in the survey). In general, we can define the following probability density model for each datapoint $\mathbf{x} \in \mathbb{R}^d$ and presence row $\pi \in \mathbb{Z}_2^d$:

$$f : \{\mathbb{R}^d \times \mathbb{Z}_2^d\} \to \mathbb{R}, \tag{3.1}$$

$$\Pr(\mathbf{x} = \hat{\mathbf{x}}, \pi = \hat{\pi}) = f(\hat{\mathbf{x}}, \hat{\pi}; \zeta). \tag{3.2}$$

Here, the function of the model parameter ζ is a very general one, providing information for the model both in the sense of the dataset and the distribution of missingness. As this is a probability distribution, we have the following relation:

$$\forall \zeta, \int_{-\infty}^{\infty} dx_d \cdots \int_{-\infty}^{\infty} dx_1 \sum_{\pi_d=0}^{1} \cdots \sum_{\pi_1=0}^{1} f(\mathbf{x}, \pi; \zeta) \equiv 1. \tag{3.3}$$

Dependence between features is an important concept: for example, consider the following two-dimensional Gaussian distribution:

$$f(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(\frac{-1}{2(1-\rho^2)}\left[\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - 2\rho\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2}\right]\right). \tag{3.4}$$

Note that this distribution, as we expect, is of type $\mathbb{R}^2 \to \mathbb{R}$ in the full data case, and it has five parameters $(\rho, \sigma_1, \sigma_2, \mu_1, \mu_2)$. In particular, the correlation index ρ allows for correlation between the two features. This simple example illustrates that dealing with the models feature-by-feature is also not general enough.

Distinctness A common assumption is often made, however, which makes analysis significantly more tractable. This is the principle of *distinctness*, described in

Little and Rubin [47] amongst others, and requires that there exists some partition of the features of the model-governing parameter ζ, which we will hereafter refer to as $\zeta = (\theta, \psi)$ in keeping with the notation of this book, such that:

$$f(\pi|\theta, \psi, \mathbf{x}) = f(\pi|\psi, \mathbf{x}) \forall \mathbf{x}, \psi, \tag{3.5}$$

$$f(\mathbf{x}|\theta, \psi, \pi) = f(\mathbf{x}|\theta, \pi) \forall \theta, \pi. \tag{3.6}$$

The above equations denote that θ is the model for the data alone, and likewise ψ for the missingness structure. The parameter space in which the original model ζ resides must thus be equal to the Cartesian cross product of those spaces for θ and ψ. Schafer [65] observes that:

> *...in many situations this is intuitively reasonable, as knowing θ will provide little information about ψ and vice versa.*

In §3.1.4 we will cover various common terms when dealing with properties which can be possessed by the p.d.f given by f. For now, it is important to reiterate that even with the (fairly weak) assumption of distinctness, we have a very general model, in which feature correlations are catered for as well as inter-dependency of the distributions of $\mathbf{x}$ and π on the values taken by one another, even if we make the above assumption that the model parameters themselves do not interfere.

3.1.3 Assumptions

Throughout this chapter, we will make it clear under which assumptions certain techniques of dealing with missing data operate. A list of common assumptions is provided below for reference. This is not an exhaustive list, and neither should it be implied that these assumptions in any way discredit various methods: their

review in the following chapter is primarily to provide completeness with the literature, and show situations in which other methods can, rightly, be used. It should also be noted that by definition, a review of existing methods would not be required if there existed one particular method which out-performed all others in every case. Disclaimer aside, methods often operate under the following conditions, all of which are given also in converse form to illustrate the wide range of properties which may be assumed for a given application:

1. Data are missing (completely) at random (MAR or MCAR).

2. Conversely, data are not missing at random (NMAR); perhaps a model is known relating the values of datapoints to the distribution of missingness. Chapter 1 of the Little and Rubin book gives a good example of this, where data are conditionally missing on their value.

3. There is a known, or calculable, *model* of the distribution of the data θ (a *parametric* method).

4. Conversely, data must be filled in without any assumptions on model type.

5. The parameters of a data model θ are distinct from those governing the mechanism behind the pattern of missingness. That is, in the notation used above in §3.1.2, we have the joint parameter space of θ and ψ being the direct product of the parameter spaces of θ and ψ.

6. Conversely, there may be a causation relation between the values which could be taken by a datapoint and its missingness, meaning the models f^θ and f^ψ were in fact only marginals of a non-trivial joint distribution $f^{(\theta,\psi)}$ giving extra information about the correlation between the models.

7. The missingness of data in a dataset indicates true values that are *meaningful for analysis* (the primary assumption made in the book by Little and Rubin, numbered Assumption 1.1 in the *Second Edition*).

8. Conversely, missing data is known to be *structurally missing* (see §3.1.5). It may also be unknown - see §3.4.3 for a brief discussion on ways of dealing with this.

A commonly-used term to denote the case of the two assumptions of MAR and data model distinctness is *ignorable* [16]. These definitions are discussed more fully in the following section.

3.1.4 Common types of missingness

Missing at random A common type of missingness used in literature related to data with incomplete observations is the phenomenon of being *missing at random* (MAR), which we mentioned above in §3.1.3. A concise definition of this is given by a website[1] maintained by the London School of Hygiene and Tropical Medicine [12]:

> *...given the observed data, the missingness mechanism does not depend on the unobserved data.*

Thus, two datapoints which share values within their mutually *observed features*, under the condition of being MAR, would have the same models for the distribution of their missing values. An example would be:

$$X = \begin{cases} x_1 = & a & b & \cdot \\ x_2 = & a & \cdot & c \end{cases} \tag{3.7}$$

[1] Available at `www.missingdata.org.uk`.

Here, since x_1 and x_2 share their value a corresponding to feature f_1, we can say that the missing value x_{22} will be drawn from the *same distribution* (but will not necessarily be the same value) as the value x_{12}, here labelled b, and likewise x_{13} distributed as c: that is, if this system is MAR, by virtue of the first datapoint x_1, we know that the missingness mechanism can only ever depend on features f_1 and f_3, and by a similar argument with x_2 it depends only on f_1 and f_2, whence it can only ever depend on the value of f_1 (although it is not guaranteed it does – it is simply the only option available). Note also that this very specific statement can only be made as the points share exact values of features, in their values for feature f_1. Thus, the only conditioning (under MAR) on the missingness inherent in the points x_{13} and x_{22} was that $x_{i1} = a$. Rubin [64] provides the following technical definition:

> The missing data are missing at random if for each, possible value of the parameter ϕ [*which we refer to as* ζ], the conditional probability of the observed pattern of missing data, given the missing data and the value of the observed data, is the same for all possible values of the missing data.

This definition can be cast into the following form: Each $(\mathbf{x}, \pi)$ combination can induce a partition of the vector $\mathbf{x}$ into an 'observed' and a 'missing' portion, denoted and defined hereafter as:

$$\begin{aligned} x_{obs} &= \{x_i : \pi_i = 1\}, \\ x_{mis} &= \{x_i : \pi_i = 0\}. \end{aligned}$$

We define data as being missing at random if the model has the following property for every $\mathbf{x}$ and π:

$$f(\pi|\zeta, x_{obs}, x_{mis}) \equiv f(\pi|\zeta, x_{obs}) \tag{3.8}$$

This definition requires a careful interpretation. It is now possible to see why the above example required the values x_{13} and x_{22} to have the same distribution:

$$\begin{aligned}
f(\pi|\zeta, x_1 = a, x_2 = b, x_3 = x_{13}) &= f(\pi|\zeta, x_1 = a, x_2 = b) \forall x_{13}, & (3.9)\\
f(\pi|\zeta, x_1 = a, x_2 = x_{22}, x_3 = c) &= f(\pi|\zeta, x_1 = a, x_3 = c) \forall x_{22}, & (3.10)\\
\rightarrow \{\text{with } x_{22} = b &\quad , \quad \text{by (3.10)}\} & \\
f(\pi|\zeta, x_1 = a, x_2 = b, x_3 = c) &= f(\pi|\zeta, x_1 = a, x_3 = c) & (3.11)\\
\rightarrow \{\text{with } x_{13} = c &\quad , \quad \text{by (3.9)}\} & \\
f(\pi|\zeta, x_1 = a, x_2 = b, x_3 = c) &= f(\pi|\zeta, x_1 = a, x_2 = b) & (3.12)\\
\Longrightarrow f(\pi|\zeta, x_1 = a, x_3 = c) &= f(\pi|\zeta, x_1 = a, x_2 = b) & (3.13)\\
&= f(\pi|\zeta, x_1 = a). & (3.14)
\end{aligned}$$

Missing at random, then, does *not* imply that the missingness pattern is completely unrelated to the data values. In fact, it does not even preclude the distribution of π being unrelated to the *parameters* driving the model; however, as we mentioned above, if it is true that data are MAR and furthermore the parameters θ and ψ making up the general model ζ are distinct, we have the following equation for **ignorability**:

$$f(\pi|\zeta, x) \equiv f(\pi|\psi, x_{obs}). \tag{3.15}$$

This is a common framework under which to operate, and implies that the mechanism governing Π cannot itself tell us anything about the data model θ. Unfortunately, all of these conditions (distinctness, MAR and thus ignorability) are

in general very difficult to infer from a given dataset with no prior knowledge about the model driving the data collection process [12]. One important property of ignorability, used extensively in statistical methods for missing data, is that the expectation of a model parameter is unbiased. Suppose, for example, we have a set $\{x_1 \dots x_n\} \in \mathbb{R}$ of one-dimensional data and our model was $f(x|\theta) \sim N(\theta_1, \theta_2)$, i.e. $\theta \equiv (\mu, \sigma^2)$: if this set can now be split into an observed part $x_{obs} = \{x_1 \dots x_p\}$ and a missing part $x_{mis} = \{x_{p+1} \dots x_n\}$, and we assume that the missingness mechanism is ignorable, it is true to say that:

$$x_{obs} \sim N(\hat{\mu}, \hat{\sigma^2}) \Longrightarrow x_{mis} \sim N(\hat{\mu}, \hat{\sigma^2}) \tag{3.16}$$

for any estimator $(\hat{\mu}, \hat{\sigma^2})$ we care to infer from the observed data x_{obs}; in particular this works for the maximum likelihood estimator for μ, i.e. the sample mean of x_{obs}.

Missing completely at random The model can be simplified further still if we make the highly restrictive assumption that where missingness exists, it has *nothing whatsoever* to do with either the model governing the data collection nor the actual values taken by the data: it is simply a model all of its own, and could in theory be applied to any similar datasets of the same dimension. This is the *missing completely at random* (MCAR) assumption, and implies:

$$f(\mathbf{x}, \pi; \zeta) \equiv g(\mathbf{x}; \theta) \cdot h(\pi; \psi) \tag{3.17}$$

$$\Longleftrightarrow f(\pi|\zeta, x) \equiv f(\pi|\zeta) \forall x \tag{3.18}$$

(compare Equation 3.8 for the definition of MAR). Note that even this assumption provides for the different features to have varying distributions of missingness,

perhaps as Bernoulli trials with different success probabilities; it also allows correlation between missingness features (e.g. if f_1 is missing then there is a high probability that f_2 is as well, etc.). This model is often, however, seen as too restrictive, because the nature of many data collection scenarios may often imply a causation relation between the data values and whether they will be present. A succinct review of these methods is also given in [24]. Finally, if the following relation is true:

$$f(\pi|\zeta, x_{obs}, x_{mis}) \text{ is non-trivially dependent on } x_{mis} \tag{3.19}$$

then the data is said to be **not missing at random** (NMAR), as may be expected. Scheid [66] provides some insight as to how model selection could be performed in some of these cases. An example of this is covered in detail in Little and Rubin, where a system is investigated such that the absence of features means that they are above or below a certain value, resulting in expressions such as the following (c.f. Little and Rubin p. 13):

$$f(\pi_j|\zeta, x_j) = \begin{cases} 1, & x_j > \alpha, \\ 0, & x_j \leq \alpha. \end{cases} \tag{3.20}$$

Here, the distribution of π clearly can depend on the missing values, as the missingness indicator is itself entirely governed by the position of the data features. Thus, if $x_j \leq \alpha$ here, then it is true that $x_j \in x_{mis}$ and $f(\pi|\zeta, x)$ is clearly dependent on its value.

3.1.5 Structurally missing data

In the previous section, when describing common model structures inducing incomplete data, we have consistently used the assumption numbered as '7' in

§3.1.3 (hereafter referred to as the *validity* assumption): that is, these models make the implicit assumption that missing data values hide meaningful completions of data features with some valid physical interpretation. That is, the model works on the following basis:

> *We shall do the best we can with the data available to us in terms of understanding its model; however, at every point we would prefer to be working with the full dataset, involving the inclusion of those features hidden by the missing data structure.*

However, quite a different situation results when we relax the validity assumption. Thus, we progress to the converse Assumption 8, hereafter referred to as the *structurally missing* assumption. Chechik et al [14] motivate their paper on instance-specific margins in SVM (discussed in full in Chapter 4) with structurally missing data. They describe it as per the following four quotes:

- "Unlike the case where a feature exists but its value is not observed . . . a feature may not even exist (structurally absent) for some of the samples."
- ". . . samples have varying subsets of observable features due to the *inherent* properties of the instances."
- ". . . in some cases, features are known to be nonexisting, rather than have an unknown value."
- ". . . each data instance reside in a lower dimensional subspace of the feature space, determined by its own existing features."

Although a fairly recent consideration, work has been done with the structural missingness assumption in areas of classification theory, such as the study on

clustering due to Wagstaff [82] and second-order cone program based regression analysis due to Shivaswamy et al [70]; these papers all describe a similar system with regards to structurally missing data. A rigorous definition of the properties of such a dataset is, however, complex by nature. Since it is an assumed property of data rather than something which can be made statistically obvious, a certain mathematical definition is not obvious. However, as is apparent from the above quotes, Chechik et al define such a phenomenon as every point lying in its own instance-specific subspace of the combined feature space. The best way, perhaps, would be to define a prior probability for the missingness pattern and then provide some model for each. If we let $F = \{1 \dots d\}$, and define $\mathcal{F} = \mathcal{P}(F)$, the power set of F denoting all possible subsets, and $D = 2^d$ so that $|\mathcal{F}| = D$, define the missingness *selection prior* as:

$$\Phi_k = \Pr(\mathcal{I}_\pi = \mathcal{F}_k) \forall k \in \{1 \dots D\}, \tag{3.21}$$

$$\sum_{k=1}^{D} \Phi_k = 1, \tag{3.22}$$

where $\mathcal{I}_\pi$ denotes the indices of π equal to 1 (or equivalently 0). Thus, for every possible missingness pattern, we can assign a probability Φ_k that it will occur in the 'next' datapoint for analysis. For a given dataset X, this induces a partition $\{X^{\langle k \rangle}\}_{k=1}^{D}$ of datapoints $\{x_i^{\langle k \rangle} \in \mathbb{R}^{d_k}\}_{i=1}^{N_k}$, with the relevant features present as given in the index set $\mathcal{F}_k$, where $d_k = |\mathcal{F}_k|$: that is, the number of present features given in the missingness arrangement k. Next, for every k a *specific* probability model $f_k(\mathbf{x}; \theta_k)$ must be produced:

$$\theta_k \in \mathbb{R}^{M_k}, f : \mathbb{R}^{d_k} \to \mathbb{R}$$

$$\text{such that } \forall \theta_k \in \mathbb{R}^{M_k}, \int_{\mathbf{x} \in \mathbb{R}^{d_k}} f(\mathbf{x}; \theta_k) d\mathbf{x} \equiv 1. \tag{3.23}$$

At first this formulation may seem entirely intractable, with a dataset with even 10 dimensions producing $2^{10} = 1024$ potential partitions, each with their own probability model of M_k variables dealing with very different types of datapoints with features ranging from null sets to full complements. However, things can be made at least conceptually simpler in developing the methods to deal with this case, if a binding philosophy can be introduced to the problem of classifying such a seemingly disparate set of points:

> *In all subsequent methods, with structurally missing data it is important that every datapoint feature present in a dataset must contribute to a procedure, with no pre-processing or assumption being made on any basis involving 'true' values.*

Thus, to deal with this type of data, an exclusive-and-exhaustive approach must be adopted: we work with the data that is there, and assume nothing else. This philosophy underpins the entirety of the work described in Chapter 5 as we seek to improve on existing methods using an imputation-avoiding approach.

3.2 Simple methods

In this section, we briefly overview some simple ways of dealing with datasets with missing features, assuming no prior knowledge of the model concerned, either for the data distribution θ or the missingness pattern ψ. We provide this here as an introduction to various ways available to a method for which no useful information is either available or desirable, partially motivated by providing a context in which to develop methods for structurally-missing data, described above in §3.1.5. Chapter 6 of the thesis due to Marlin [52] also provides a good review in

this area. Further analyses are given by Weir [84], Marwala [54] and Allison [1]. We shall note some trivial methods and provide a brief overview of imputation methods.

3.2.1 Trivial methods

There are two simple methods, largely too basic to be used much in literature, which we mention here: case deletion and marginalisation. The former could be applied in any relevant context; the latter could potentially be used in a classification task.

Case deletion Also known as *complete-case analysis*, we define case deletion as the pre-processing of a dataset $X \in \mathbb{R}^{N \times d}$ via the deletion of *any* datapoint $x_i \in \mathbb{R}^d$ which does not have its entire complement of features: equivalently, the decision might be made to delete from analysis any feature $\{x_{ij}\}_{i=1}^{N} \in \mathbb{R}^N$ which is not shared by all datapoints in the set, depending on the ratio of N to d according to how much information would be lost in either case. Apart from the cases where data is both missing completely at random and largely complete, such that case deletion did not impact significantly on the inherent properties of the distribution of the data, this method is generally too cavalier with the rejection of data to be suitable. For example, consider the following dataset:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdot \\ x_{21} & x_{22} & \cdot & x_{24} \\ x_{31} & \cdot & x_{33} & x_{34} \\ \cdot & x_{42} & x_{43} & x_{44} \end{pmatrix} \tag{3.24}$$

This dataset would be completely obliterated by case or feature deletion, despite only a quarter of its values being missing. As we let $(N, d) \to \infty$ in a similarly structured example, we end up with datasets whose missingness ratio $N_m/N \to 0$ but all of which would be left as null sets under deletion.

Marginalisation When computing the distance of a point x with missing data away from either another point y (which may be incomplete) or a centre point $\mathbf{a}$ (which is assumed to be of full data), it could be argued that the missingness could be worked around by only considering those features shared by both points. Although an assumption similar to this is used as the starting point from where methods in later chapters shall be derived, great care must be taken over its use in a 'normal' classification procedure as the distance metric which is produced will not obey the triangle inequality. In particular, if a distance is required between two points x and y which do not share features, their distance will be calculated this way as being zero. If Parzen windowing (§2.2.3), for example, is used with this instance-specific distance metric, probability contours will result with 'waves' propagating to infinity in each feature dimension, and the resulting map will not be normalisable over all feature space as a result. Thus, the use of instance-specific margins (which we shall return to later) in this way with standard classification methods should also be discouraged.

3.2.2 Imputation

The process of *imputation*, the verb *to impute* literally meaning 'to attribute or assign as a characteristic', refers to the process of the pre-filling of a dataset with missing data values, so that a complete (full-data case) dataset is obtained, to

which standard methods, usually of classification, can be applied. It is a widely used method in many areas, including the use by Armknecht [2] in price imputation in finance. Clearly, there exist many different ways of doing this, all with their own models and assuming different prior information – sometimes more, sometimes less or none – about the distribution of the extant data. It is important to note that although choice of imputation method itself may well depend on the empirical distribution of the dataset, the overriding feature of imputation methods is that, as mentioned in this section's introduction, *no model is required* to be explicitly stated in terms of the distribution of either data or presence matrix. Once chosen, any imputation method should be able to work given a dataset alone. We review a few methods here, which we shall use later on in this account as benchmarks with which to compare the methods we will devise for instance-specific geometric margin analysis.

Zeros Imputing a dataset by zeros is a very simple, if rather rudimentary, way of completion of missing data. This process, although basic, performs reasonably well when the dataset has been *centred*; that is, for each feature j with data entries $\{x_{ij}\}_{i=1}^{N}$, over all present data $x_{ij} \in X_{obs(j)}$, the mean is already zero. Where this is not the case or centring is inappropriate, this method can cause unnecessarily inaccurate completions of a dataset.

Means A slight generalisation of the imputation by zeros method is to impute by the restricted-data *mean values* of each feature in turn, thus avoiding the problem where data has not been centred. Thus, for each j, we split the specific feature vector $X_{(j)}$ into $X_{obs(j)}$ and $X_{mis(j)}$ according to those points where this feature

was observed or missing, and define for all i such that $x_{ij} \in X_{mis(j)}$:

$$x_{ij} \leftarrow \frac{1}{N_{obs(j)}} \sum_{i=1}^{N_{obs(j)}} x_{obs(j)}^{(i)}, \tag{3.25}$$

where $N_{obs(j)}$ denotes the number of datapoints with feature j present, and $x_{obs(j)}^{(i)}$ (for $i \in \{1 \dots N_{obs(j)}\}$) as being the j^{th} feature of the i^{th} datapoint of X with this property.

Nearest neighbours Algorithm 1 describes a simple approach related to clustering methods, which imputes datapoints using a nearest-neighbours algorithm.

3.2.3 Adding extra features

In their paper, Chechik et al [14] noted a simple method of completing a dataset that involved the creation of a new set of augmented features, to display the presence or absence of data as appropriate. This could be done in many ways, the most basic being the augmentation of the data matrix X with the $(0, 1)$ realisation of the presence matrix Π, thus resulting in the $(N \times 2d)$ matrix, then imputing in some other manner. The advantage of this is that it is made more obvious where a feature has been filled in, setting it apart in feature space from its fellow points which were perhaps complete before processing; the obvious disadvantage of this is that the dataset itself grows, which (apart from issues regarding computational complexity) does not easily allow a direct comparison of efficiency with a method that has only been set to work with the original d-dimensional dataset once completed. A refinement of this method would be to apply an analytical algorithm to the presence matrix Π to ascertain the number of *distinct* patterns of missingness

Algorithm 1 Nearest-neighbour imputation algorithm

for $x_i : i \in \{1 \dots N\}$ **do**

Let obs_i be the index of observed features of x_i, mis_i the missing features index, with $x_{obs(i)}$, $x_{mis(i)}$ the relevant vectors of values.

if $obs_i = \emptyset$ (i.e. x_i is entirely null) **then**

Fill with feature averages as per 'mean imputation' above

else

$X_{obs} \leftarrow$ (columns of X indexed by obs_i)

$\mathcal{Q}_X \leftarrow$ (index of points in X_{obs} with present features)

$X'_{obs} \leftarrow$ (X_{obs} restricted to non-null points)

Make vector of distances d between each point in X'_{obs} and original vector $x_{obs(i)}$, with distances calculated only in dimensions with mutually-present features

Remove original datapoint, take indices of K nearest datapoints

Impute $x_{mis(i)}$ with the feature means of these K points

end if

end for

present, thus reducing the number of flags. For example, if it was deduced that only 3 missingness patterns occurred within a dataset (which may well be more likely to naturally be the case than the maximum of 2^d), it would be perfectly reasonable to assign a 'classification' flag as an extra dimension of, say, $\{-1, 0, 1\}$ depending on the particular pattern of each datapoint.

3.3 Expectation-maximisation (EM)

The so-called Expectation-Maximisation (EM) method (see Schafer [65], Borman [10] among others) is a method that has been applied for decades in dealing with the best way to perform imputation on a dataset with missing data. The central assumptions are that data is missing due to meaningful data being omitted, and generally operates under the ignorability assumption of a MAR pattern and independence of model and presence distributions; however, it is designed for use in a general case. A model $f(\mathbf{x}|\theta)$ of the underlying process from which the *complete* dataset X would have been drawn is a pre-requisite, at least in a well-estimated form. A reasonable estimate of the parameter vector θ itself is also useful as a starting point. No prior information about the missingness distribution or its parameters ψ is assumed, so we hereafter work only with θ. The technique was introduced for maximum likelihood estimates from incomplete data by Dempster et al [18], and later Ghahramani and Jordan [27] apply the technique to supervised learning from incomplete datasets. It is also covered in depth in the book by Little and Rubin [47]. Essentially, the purpose of the EM algorithm is to optimise both data completion and model parameter selection, by alternating between iterative steps concerning both these objectives one after the other.

3.3.1 Algorithm

The process involved in computing an EM-based estimate of the most likely distribution of data in a dataset can be depicted in a two-step iterative process, commonly known as the 'E-step' and the 'M-step'. In simple terms, these calculations can be described thus:

1. Calculate a general expression for the *complete-data* likelihood $L(\theta|\mathbf{x})$, and thus log-likelihood function $l(\theta|\mathbf{x})$, for a set of fully observed data $\mathbf{x}$ and any given model parameters θ. Identify the *sufficient statistics* $S^{(1)} \ldots S^{(m)}$ which would be required in this expression to evaluate its expectation. Thus $l = l(\theta, S^{(j)})$ should be a function of the model parameters θ and statistics $S^{(j)}$, but should not otherwise explicitly depend on the data values.

2. Form an expression for the expectation of each of the statistics $S^{(1)} \ldots S^{(m)}$ conditional on the observed data, using regression analysis to compute these if necessary[2].

3. Decide on an initial estimate of the parameter θ_0. Let $k = 0$.

4. Calculate the current values $S_k^{(1)} \ldots S_k^{(m)}$ of the sufficient statistics identified in step 1, based on the observed data and the current estimate of the model θ_k.

5. **E-step:** Identify a *particular* likelihood objective function based on the *current* values of these statistics S_k, $l_k(\theta; S_k)$ being a function of θ only.

6. **M-step:** Calibrate the current estimate of θ by maximising this particular objective with respect to θ; increment k and return to step 4.

[2] See the bivariate normal example on p. 170 of Little and Rubin for an example of this.

These steps can be summed up by the following iterative relation (see [10]):

$$\theta_{k+1} = \arg\max_{\theta} \left\{ \mathbb{E}_{Z|\mathbf{x},\theta_k} \left\{ \log \mathbf{p}(\theta|\mathbf{x},\mathbf{z}) \right\} \right\}, \tag{3.26}$$

$$\text{where } \mathbb{E}_{Z|\mathbf{x},\theta_k}(\cdot) \equiv \int_{\mathbf{z}} \mathbf{p}(\mathbf{z}|\mathbf{x},\theta_k)(\cdot)d\mathbf{z}. \tag{3.27}$$

Here, we use $\mathbf{z}$ as shorthand for the data which is missing, and use the integral sign over $\mathbf{z}$ to conventionally describe that we are conditioning over all possible completions of the data in $\mathbf{x}$. Thus, these equations state that at each iterative step, we should search for the optimal θ which maximises the expression given in Equation 3.26, based on our previous assumptions for the values of $\mathbf{z}$ and θ. This is best understood given a worked example, from Little and Rubin (page 168), described in the next section.

3.3.2 Example

Suppose we are aware that, in a set of data $X = \{x_1 \dots x_n\}$, that the x_i are independently and identically distributed (i.i.d), under a normal distribution $x_i \sim N(\mu, \sigma^2)$, so that here $\theta = (\mu, \sigma^2)$; furthermore, $x_1 \dots x_r$ are present and observed, with $x_{r+1} \dots x_n$ being missing. The expression for the p.d.f. is the following familiar form:

$$f(x_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-1}{2\sigma^2}(x_i - \mu)^2 \right) \tag{3.28}$$

Thus, the *complete-data* likelihood function is as follows:

$$\begin{aligned} L(\mu, \sigma^2; X) &= \prod_{i=1}^{n} f(x_i; \mu, \sigma^2) \\ &= (2\pi\sigma^2)^{\frac{-n}{2}} \exp\left(\frac{-1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right) \\ \implies l(\mu, \sigma^2; X) &= -n\log|\sigma| - \frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}x_i^2 - 2\mu\sum_{i=1}^{n}x_i + n\mu^2\right) \\ &= -n\log|\sigma| - \frac{1}{2\sigma^2}\left(S_2 - 2\mu S_1 + n\mu^2\right), \end{aligned} \tag{3.29}$$

where the values:

$$S_1 = \sum_{i=1}^{n} x_i \text{ and } S_2 = \sum_{i=1}^{n} x_i^2$$

are the sufficient statistics for this problem: given full data, they are directly calculable from this and known values. The next step is to derive the calculation of these statistics in the presence of missing data, so that the proper *expectation* of this expression can be calculated. Clearly, under this model and any estimate $(\mu', \sigma^{2(\prime)})$ for each missing data value $\{x_i\}_{r+1}^{n}$, we have $S_1 = \mu'$. For the expectation of x_i^2, we refer to the definition of σ^2:

$$\begin{aligned} \sigma^{2(\prime)} &= \mathbb{E}(x_i - \mu')^2 \\ &= \mathbb{E}(x_i^2) - 2\mu'\mathbb{E}(x_i) + (\mu')^2 \\ &= \mathbb{E}(x_i^2) - (\mu')^2 \\ \implies \mathbb{E}(x_i^2) &= (\mu')^2 + \sigma^{2(\prime)}. \end{aligned}$$

Thus:

$$S_1(X; \mu', \sigma^{2(\prime)}) = \sum_{i=1}^{r} x_i + (n-r)\mu', \tag{3.30}$$

$$S_2(X; \mu', \sigma^{2(\prime)}) = \sum_{i=1}^{r} x_i^2 + (n-r)\left[(\mu')^2 + \sigma^{2(\prime)}\right]. \tag{3.31}$$

Thus, by Equation 3.29, every time we estimate (μ, σ^2) and thus compute at each step the requisite values – say (e_1, e_2) – of these statistics, we can create a particular objective function $l(\mu, \sigma^2; e_j)$ (by substituting these values in appropriately to the general objective given above) which can be maximised for the parameter values (μ, σ^2), thus giving a calibrated value for these parameters for passing to the next step.

3.4 Methods for structural missingness

Up until now, we have provided methods which either assume that the missingness of data is either ignorable (or similar) and/or can be modelled by some statistical distribution ψ. However, in order to deal with the complex concept of structurally missing data, as described above in §3.1.5, we require new tools. Recall that the binding philosophy must always be one of using the maximum available amount of information in a dataset without making any assumptions as to the 'true' nature of the 'hidden' data, as in this case this information may be meaningless or unhelpful. In preparation for a study in later chapters of our own methods for classifying data under this assumption, we briefly review the study by Dick [19] in §3.4.1. This process is designed to work in a kernel-induced feature space, and later we shall build on the concepts which shall be introduced here, in the design of our system. More information on *model-based* multiple imputations, which are not necessary for this study, are to be found in the book by Rubin [63], Chapter 10 of Little and Rubin [47], and Chapter 3 of Schafer [65]. These methods are based on the EM algorithm described above, and work on a similar basis – that of multiple completions of a dataset resulting in a full matrix to which a standard method

can be applied. Papers by Banasik and Crook [5] and Tanner and Wong [76] show how the data augmentation method can be used for classification. Where a model exists, this method may also be of some empirical help, although the infinite-imputation method covered here is more general. After explaining this method, and identifying some other ways in which the problem of nonparametric missing data has been tackled in literature, we turn attention in §3.4.3 to a review of a consultation entered into regarding the potential use of multi-classifier methods for structurally missing data, and the selection procedures which would be required should it not be clear whether the missing data was ignorable or structural.

3.4.1 Infinite imputations

Uwe Dick [19] asserts that learning from incomplete data in a kernel-induced feature space is possible via a process of so-called 'infinite imputations', the model of which we shall now describe. Allow the dataset $X \in \mathbb{R}^{N\times d}$ to have missing data, so that the presence matrix $\Pi \in \mathbb{Z}_2^{N\times d}$ has some values for which $\pi_{ij} = 0$. Since the total number of features in the dataset is equal to Nd, we can view the space Ω_X of all possible imputations as being isomorphic to $\mathbb{R}^M$, where $0 \leq M \leq Nd$ is a measure of the number of missing features in the entire dataset. Dick defines the following:

$$\omega_X \in \Omega_X \sim \mathbb{R}^M \tag{3.32}$$

such that ω_X is a single realisation of an imputation of X, induced by a completion vector $\mathbf{q} \in \mathbb{R}^M$: thus, ω_X itself is the same size as X (and can thus be indexed similarly) but has the constraint that $\omega_{ij} \equiv x_{ij}$ for all (i, j) such that $\pi_{ij} = 1$. Thus, Ω_X has M free parameters and is, as required, isomorphic to the parameter space $\mathbb{R}^M$ inhabited by $\mathbf{q}$. Dick observes that this relation induces a family of

kernels:

$$K(\omega)(x_i, x_j) = k(\omega_i, \omega_j) \tag{3.33}$$

Thus, if some probability measure $p : \mathbb{R}^M \to \mathbb{R}$ can be deduced from prior information regarding the likelihood of different completions, we can define the kernel product *with respect to the measure* p as follows:

$$K(p)(x_i, x_j) = \int_{\omega \in \Omega_X} K(\omega)(x_i, x_j) dp(\omega) \tag{3.34}$$

As this kernel function itself can produce the required kernel matrix $\{k(x_i, x_j)\}$ for the given imputation distribution prescribed by p, the general learning problem for any optimisation criterion $R(\alpha, K)$ dependent on the kernel matrix K (and regularising function Q) can be formulated as follows:

$$\hat{R}_{(K,\gamma)} = \arg\min_{\alpha,p} R_{(K,\gamma)}(\alpha, p) = R(\alpha, K(p)) + \gamma Q(p) \tag{3.35}$$

$$\text{such that } \forall \omega p(\omega) \geq 0, \int_{\omega \in \Omega_X} p(\omega) d\omega \equiv 1. \tag{3.36}$$

Dick observes that this more general case of optimisation procedure integrates over *infinitely* many realisations of the completion of the dataset X. Since the functional parameter p is allowed to vary, in theory any imputation can be reached, and the process is 'guided' by the regularising function $Q(p)$, with the parameter $\gamma \in (0, \infty)$ acting very similarly to the payoff-cost parameter C used in a traditional SVM approach (see §2.2.6). Although this is an intractably large space over which to search, Dick shows that there must always exist an optimal distributional parameter $\hat{p}$ which is supported on at most $n + 2$ different imputations, and thus uses a greedy algorithm to solve the optimisation procedure.

3.4.2 Other methods

Here we briefly review some other methods which have been adopted for tackling missing data problems. Three lines of study are covered: using second-order cone programming for handling an *uncertain* data structure, Smola's kernel extension of support vector methods into problems with missing data, and finally the work on geometric margin clustering due to Wagstaff.

Cone programming with uncertain data Shivaswamy et al [70] propose a method, based on previous observations by Bhattacharyya et al [8] regarding second-order cone programming, to prove that the support vector machine classifier (§2.2.6) can be adapted in the case where measurements are uncertain. The paper operates under the framework that the following familiar constraint in the 'certain data case' of:

$$y_i(b + \langle \mathbf{w}, x_i \rangle) \geq 1 - \xi_i \tag{3.37}$$

(compare Equation 2.25), is replaced by the following:

$$\mathbb{P}_X \left\{ y_i(b + \langle \mathbf{w}, x_i \rangle) \geq 1 - \xi_i \right\} \geq 1 - \kappa_i. \tag{3.38}$$

Thus, the deterministic constraint is replaced by a probabilistic one, with a further attribute κ_i to allow for the softness of the probabilistic margin to be altered. They formulate results assuming both a *worst distribution* and a normal distribution of each x_i under assumptions of a given mean and variance, and show that in this case (using Chebyshev inequalities) the classification via a support vector machine of the x_i can be successfully formulated into a second order cone program (see Equation 4.9 for the definition of this in the context of Chechik et al's SVM study).

Kernel extensions with missing data Smola and Hofmann [72] make insights into the problem of 'systematically dealing with incomplete data', observing that there is no unified framework to tackle the general case which does not entirely rely on a pre-imputation step. They define an *exponential family* of functions of datapoints as a vector $\phi : \mathbb{R}^d \to \mathbb{R}^m$ of *sufficient statistics* for a dataset $X = \{x_i \in \mathbb{R}^d\}$ with a probabiltiy density model θ (see §3.3 for use of sufficient statistics in the E-M method above); thus, in the full data case, we may define a probability density involving exponential terms in an *exponential-normal* form as:

$$p(x;\theta) = p_0(x) \exp\left(\langle \phi(x), \theta \rangle - g(\theta)\right), \tag{3.39}$$

where g is a specifically-chosen function based on the logarithm of the integral of $p_0 e^{\langle \phi(x), \theta \rangle}$. They extend this in the incomplete-data case using a model based on θ in a similar way to that of EM, and thereafter set out a framework of kernel classification methods which would be able to adapt in this case. They mention that these methods are very susceptible to non-convexity and that the integrals required in computing the requisite functions g for the incomplete-data analysis may be intractable. However, they show that the system can successfully be used in some cases to perform parameter estimation and support vector classification in the presence of the data-driving model θ.

Geometric margin clustering Mentioned above, the paper by Wagstaff [82] proposes a clustering system to deal with essentially the same generality of data structure as the Dick paper [19] mentioned in the previous section. He observes that much clustering work in literature relies heavily on either imputation or marginalisation, and to deal with missing data in a non-parametric, general manner he proposes a form of K-means clustering utilising the following model: $\exists$ a set of

soft constraints $\langle x_i, x_j, s \rangle$ such that:

$$\forall x_i, x_j, s = -\sqrt{\sum_{f_k \in Q_i \wedge Q_j} (x_{ik} - x_{jk})^2}, \tag{3.40}$$

where Q_i refers to the set of present features in datapoint x_i. Thus, no constraints are made in any dimension not shared by the two. The parameter s represented by this expression refers to the *strength* of the constraint; that is, s represents the (geometrical instance-specific) distance of least separation which should be achieved between x_i and x_j. In most cases not all these constraints will be satisfied on an initial iteration, and thus Wagstaff proposes an optimisation procedure for clustering which minimises the total weight of violated constraints present in each iteration. This study shows that the idea of using a geometric margin to separate values with missing data has applications in other areas of classification; later on we shall present it in kernel form and show that results can be achieved with a support vector description system.

3.4.3 Discussion of multi-classifier methods

In studies involving the classification of missing data, the nature of missingness of any absent features, and hence the best method to use, may not always be obvious; imputation methods are in general more suitable for use where the assumption regarding absent datapoints being meaningful analysis, and more advanced methods suited to multiple imputations are intuitively more relevant where there exists structurally missing data. It is not always optimal to consider every single piece of missing data as being structurally absent. Furthermore, in the case of datapoints with features we do consider to be structurally absent, it does not make sense to train a classification system on imputed data and automatically expect its results

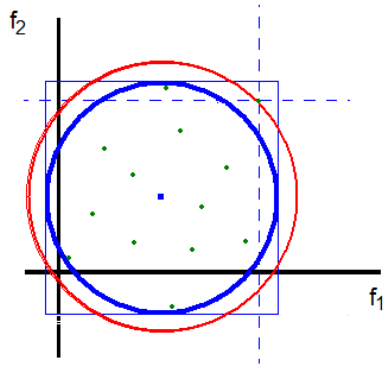


Figure 3.1: Subspace sphere expansion. New outside point added dimension by dimension would fit within existing one dimensional bounds, suggesting monocentric method using projection is flawed.

to be meaningful when projected back into the subspace in which the structurally missing data resides; in fact, it is more useful to see the structural missingness as defining a different 'type' of data altogether. The problem which needs to be addressed is that which arises where we have data which is partly unobserved, but we do not know *a priori* whether this is because it is MAR or structural. We wish to be able to consider a dataset and find a best-fit model which can tell us the likelihood of a given feature being structurally absent. If we are to avoid issues regarding the assigning of parameters to incorrect subtypes of data, such as measuring the distance between a projected d-dimensional centre and structurally absent $(d - p)$ dimensional datapoints (see Figure 3.1), we must build a whole new classifier for each different combination of statuses of features present in the training set. The status of each feature may be considered to be one of two types: "structurally absent where missing" or "omitted (MAR) when missing", giving 2^d types for a d-dimensional feature space. To apply a multi-classifier method,

we must resolve the question of when to consider features as being each of the following:

- Structurally absent: this leads to a more rigorous classifier, but the potential issue arises that we will have less data with which to train, as we have further split the full set into 'subtypes' of different patterns of presence and absence;

- Missing at random: meaning we can approach a more general classifier which has been trained on a higher quantity of more varied data but leading to the issue that we must again use imputation, and of course come up with a certain way of doing this.

Processing data where the categorisation of 'missingness type' is unclear thus requires a conclusion, given a training set, on the most likely nature of each dimension: whether we can say that, when a datapoint is missing a certain feature, it is MAR or structural. However, a more fundamental question in the presence of structurally absent data is how to process the data once it has been split into subtypes of 'missingness pattern', with a separate classifier trained on every subset. Ranking these 'mini-classifiers', which may be giving conflicting results given a test point, against each other to form a more rigorous conclusion is a difficult problem. Another potential issue is that of small sample size, and that not enough data may be present to properly train a structural classifier; in certain cases (particularly that of image processing) this may be overcome by creating a critical mass of data above and beyond the given training set, by producing extra 'prototypes': Perez [62] provides an example of this method on handwritten digits. This is the point at which prior information comes into play: we may know an important fact

Table 3.1: Types of missing data

Model	$\surd\surd$	$\surd\times$	$\times\surd$	$\times\times$	No. of types
RR	PP	PP	PP	PP	1
RS	PP	PA	PP	PA	2
SR	PP	PP	AP	AP	2
SS	PP	PA	AP	AA	4

Legend	
$\surd$	This feature is present for this training point
$\times$	This feature is absent for this training point
R	When missing, consider as omitted but underlying
S	When missing, consider as structurally absent
P	Classify with assumption of existence
A	Omit with assumption of structurality

about the nature of where the training data came from which may eliminate some possibilities when we attempt to deduce the nature of each feature type. It is possible, for example, that a classifier may "learn as it goes along" by employing a cross validation technique, thus always being open to the idea of changing its view of the nature of each feature but in a continuous way; this method would be near-optimal if 'subsequent' training data was similar to what a classifier had already seen. A weighting method could also be used to rank classifier results. As an example, consider a training set in two dimensions. Table 3.1 shows our options for classifying each pattern of missingness which could be present in a training set, depending on which model we choose. In general, the following relation applies:

$$f : (\mathbb{Z}_2^d, \mathbb{Z}_2^d) \rightarrow \mathbb{Z}_2^d$$
$$\text{where } f(\mathbf{p}, \mathbf{q}) = \mathbf{p} \vee \mathbf{q}, \tag{3.41}$$

where a value of 'true' is assumed by the glyphs of R, P and $\surd$ as depicted by Table 3.1. From the table above, we can see the choice between four separate models, some of which give 2 or 4 (in general up to 2^d) perhaps conflicting opinions as to the class of a given test point. Not only do we have to deduce the best model, but in the case where this model involves presence of structurally absent data, how to deal with the multiple classifier results. Rigorous answers to both of these questions are not clear; however, there is perhaps scope for using a method such as reversible-jump MCMC [30] in terms of the latter question of multiple classifier interpretation. Cross-validation (see §2.1) could be used to choose the best model, combined with the extra data simulation technique described above if this were appropriate. We could condition on the nature of the model and obtain results under each assumption, thus ultimately making the optimal result a function of the correlation between each feature variable. Another open question is

on whether we ultimately come up with one optimal model, or whether we consider each model in turn for a given testing set, perhaps with a weighting vector already assigned, or even allowing the classification system to morph as it went along, thus providing flexibility but perhaps less surety as to the true nature of the underlying process.

> *This discussion is an excerpt from a consultation held with Dr Peter Clifford and Dr Geoff Nicholls, University of Oxford, 17th July 2008.*

3.5 Discussion of statistical imputation

In this chapter, we have reviewed various different methods employed widely in literature for dealing with datasets with missing data. As described in §3.1.3, all of them either work best, or exclusively operate by design, when based on a set of assumptions about the nature of both the dataset and its missingness pattern. Often, a statistical method for missing data will rely on the ignorability assumption as given in Equation 3.15, or some probability-based model ζ for the distribution behind the data, which may itself partition as $\zeta = (\theta, \psi)$, based on facts that are known *a priori* about the nature of the measurements. Others make assumptions about the relevance of performing certain procedures: the EM algorithm (§3.3) is an example of where at the outset, the method intends to only consider the case where data-filling is a reasonable thing to do. It has advantages, however, in the effective estimation of these parameters in many cases, and the selection of an optimal model to fit the observed data. Dick's method of infinite imputations (§3.4.1) extends the ideas of EM to work over whole probability distributions of potential data completions, making it an important tool in the analysis of incomplete

data using an imputation-avoiding approach. Furthermore, Wagstaff's geometric-margin clustering algorithm makes in-roads into systems where no prior distribution is assumed and that the practice of imputation (see §3.2.2) is too rudimentary; in subsequent chapters we will consider further this paradigm, and devise a system to work under the assumptions of *structurally* missing data, notably the binding philosophy as mentioned in §3.1.5: that all present data should be used, and no absent data should be arbitrarily infilled at any point during the algorithm. Furthermore, we will pursue a system which blends this philosophy in with kernel methods: as Smola [72] correctly observes, precious little has been considered in the literature about a realistic extension of missing-data methods into kernel-induced feature spaces, with many statistical methods necessarily working on what would be termed 'input space' alone. Smola's system based on the underlying sufficient statistics is a good start, but we seek a system to deal with this case non-parametrically. In Chapter 4, we will consider the useful study performed by Gal Chechik et al into performing an SVM classification procedure with structurally missing data, using their method of geometrical instance-specific margins to only consider the data present in each feature as being relevant for its decision boundary calculation. By considering the models presented in this chapter and next, we can identify a system which will bring together parts of each and provide a classification system for the support vector domain description (§2.2.7) which is capable of the following:

- Classifying data in a *non-parametric* method: that is, assuming nothing about the underlying distribution θ of the dataset or its missingness pattern ψ.

- Where data are missing, to consider them as being structurally absent, and

adopting an approach towards classification that every present instance of every datapoint will contribute to the classification procedure, but nothing will be arbitrarily assumed about the in-filling of absent data.

- Working effectively in a kernel-space, again in a way which is not equivalent to any arbitrary imputation of a dataset.

The following chapters will outline the basis of the design of this classification machine, starting with a full study into the work done by Chechik on geometric margin classification. The realisation of these targets, although ambitious, will thus be able to build on the many developments in various areas already made in the area of missing data analysis.

Chapter 4

Geometric SVM with instance-specific margins

4.1 Introduction

In Chechik et al's paper of 2008 [14], they consider the problem of performing a support vector machine computation on a dataset with missing features, whilst avoiding the pre-imputation of the partial dataset. Their paper is motivated by the idea of *structurally missing* data, which we introduced in the previous chapter §3.1.5; to re-cap from the previous chapter, they define this type of data as occurring when the following is true:

> "... unlike the case where a feature exists but its value is not observed, here we focus on the case where a feature may not even exist (structurally absent) for some of the samples ... features that are known to be non-existing, rather than have an unknown value."

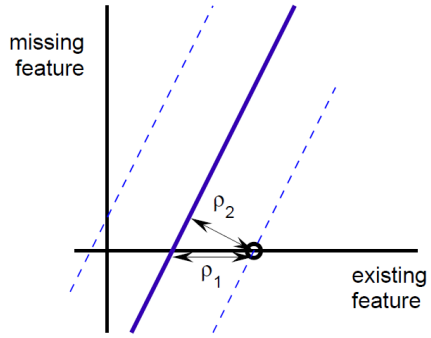


Figure 4.1: Underestimation of the margin. Note that ρ_2 underestimates the margin as it measures to an imputed point over both dimensions, whereas ρ_1 only measures in the dimension for which features are valid.

They reason that, instead of viewing datapoints as vectors of features in the same space, we should see them as residing only in the *subspace* of the main frame of reference for which features are present. This is to avoid the scenario where poor imputation of data may make those datapoints with missing features end up too close to a margin, and therefore be seen by a classifier as being 'too important' due to the data that has been filled in (see Figure 4.1). For example, for $\mathbf{x} = (x_1, \cdot, x_3)$, we see that the main input space is $\mathbb{R}^3$ but that the datapoint resides in the subspace isomorphic to $\mathbb{R}^2$. In a usual SVM, the full-data geometric margin is given by:

$$\rho(\mathbf{w}) = \min_i \frac{y_i(\mathbf{w} \cdot x_i)}{\|\mathbf{w}\|}. \tag{4.1}$$

However, this is ill-defined where x_i does not have full features. Thus Chechik et al recommend replacing this by a different metric, known in the paper as an *instance-specific margin*:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \left(\min_i \rho_i(\mathbf{w}) \right), \tag{4.2}$$

$$\text{where } \rho_i(\mathbf{w}) = \frac{y_i(\mathbf{w}^{(i)} \cdot x_i)}{\|\mathbf{w}^{(i)}\|}, \tag{4.3}$$

where here $\mathbf{w}^{(i)}$ is equivalent to the projection of $\mathbf{w}$ into the subspace for which x_i has valid features, and:

$$\|\mathbf{w}^{(i)}\|^2 = \sum_{k:f_k \in Q_{x_i}} w_k^2, \tag{4.4}$$

and $Q_{x_i} \subseteq \{1 \dots d\}$ denotes the indices for which x_i has present features. At this stage, it should be noted that the derivation of a solution of an SVM in the usual, full-data case, requires the following function to be simplified:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \left(\min_i \frac{y_i(\mathbf{w} \cdot x_i)}{\|\mathbf{w}\|} \right), \tag{4.5}$$

by removing the denominator $\|\mathbf{w}\|$ from the inner minimisation step, since it does not depend on the instance i. This is not possible in the new formulation, since the denominator is as given in Equation 4.3, precluding the use of a simple quadratic programming method for solution. Essentially, this is because the QP depends on a system which uses only the *inner products* between the datapoints and the margin $\mathbf{w}$, and their classes. Recall that this property is what makes the SVM a powerful machine to use in kernel-induced feature space. However, the computation of $\mathbf{w}^{(i)}$ requires knowledge of the dimensional structure of each of the x_i.

4.2 GMSVM formulations

In their paper, Chechik et al describe three separate algorithms to solve this more complex problem formulation given above by Equations 4.2 and 4.3, which we briefly review here. They first provide a simple, exact formulation of the problem

in the hard-margin, linear case and show that this can be reduced to a convex algorithm structure known as a *second-order cone* program (SOCP); secondly, a method is developed to provide an approximation to the margin and reduce to a familiar quadratic programming form, solvable similarly to the SVM as stated in §2.4.1. Finally, they develop an exact method that uses scalings of the full norm to approach a constrained optimisation problem that can take into account soft margins for the non-separable case.

4.2.1 Method 1: hard margin formulation

This method uses a familiar approach towards minimax optimisation, by introducing an auxiliary parameter γ as a bound, and transferring the objective into the constraints, meaning that for each given γ a solution can be sought, then γ can be optimised itself to provide a desirably tight bound. Thus, the above problem is transformed into the following:

$$\begin{aligned}(\mathbf{w}, \gamma) = \max_{\mathbf{w},\gamma} \quad & \gamma \\ \text{such that } \gamma \quad \leq \quad & \min_i \frac{y_i(\mathbf{w}^{(i)} \cdot x_i)}{\|\mathbf{w}^{(i)}\|}. \end{aligned} \tag{4.6}$$

The constraint given by Equation 4.6 is equivalent to γ being less than the quantity given *for all* i, whence:

$$\gamma \quad \leq \quad \frac{y_i(\mathbf{w}^{(i)} \cdot x_i)}{\|\mathbf{w}^{(i)}\|} \forall i \in \{1 \dots N\} \tag{4.7}$$

$$\implies \gamma\|\mathbf{w}^{(i)}\| \quad \leq \quad y_i(\mathbf{w}^{(i)} \cdot x_i) \forall i \in \{1 \dots N\}. \tag{4.8}$$

If we fix γ as constant, this program now obeys the general form of a second-order cone program, which is solvable in a convex manner (see [48] and [70]

for examples of its use in the context of missing and uncertain data) and has the following general form:

$$\mathbf{x} = \arg\min_{\mathbf{x}} \mathbf{f}^T\mathbf{x} \tag{4.9}$$

$$\text{such that } \forall i, \|A_i\mathbf{x} + \mathbf{b}_i\| \leq \mathbf{c}_i^T\mathbf{x} + d_i. \tag{4.10}$$

Chechik et al make the comments that although this method is perfectly reasonable in the hard-margin (i.e. separable SVM) case, there is no obvious way of extending the method to include a soft-margin formulation, since the derivation tends to lead to trivial solutions of $\mathbf{w} \equiv 0$ when these are included in the usual manner.

4.2.2 Method 2: quadratic approximation

In §4.1 we discussed how the denominator present in Equation 4.3 precludes the use of a quadratic program, as it cannot be taken out of the minimisation step of Equation 4.2. The next method presented in the discussed paper provides a workaround to this, via approximation of this norm. Recall the definition of the original denominator, given in Equation 4.4:

$$\|\mathbf{w}^{(i)}\|^2 = \sum_{k:f_k \in Q_{x_i}} w_k^2.$$

The second algorithm replaces this with an approximate norm thus:

$$\overline{\|\mathbf{w}\|}^2 = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{w}^{(i)}\|^2, \tag{4.11}$$

where $\|\mathbf{w}^{(i)}\|^2$ is given as above. To see how this works, consider the following dataset:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & \cdot \\ x_{31} & \cdot & x_{13} \\ \cdot & \cdot & x_{43} \end{pmatrix}, \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}. \tag{4.12}$$

Then, for example, $\|\mathbf{w}^{(2)}\|^2 = w_1^2 + w_2^2$, and:

$$\overline{\|\mathbf{w}\|}^2 = \frac{1}{4}(3w_1^2 + 2w_2^2 + 3w_3^2).$$

From here it is simple to form an objective of the form of a quadratic program (see §2.4.5) with soft margins:

$$(\mathbf{w}, b, \xi) = \arg\min_{\mathbf{w},b,\xi} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}\|\mathbf{w}^{(i)}\|^2 + C\xi_i \tag{4.13}$$

$$\text{such that } \forall i, y_i(b + (\mathbf{w}^{(i)} \cdot x_i)) \geq 1 - \xi_i. \tag{4.14}$$

Chechik et al point out that this method is expected to do well if there is a reasonable similarity between the instance-specific norms; this is the case for datasets where there is only a small degree of missingness. However, for datasets where many features are absent, the *ansatz* norm given in Equation 4.11 will not be a good approximation.

4.2.3 Method 3: soft margin formulation

Finally, a method is introduced to address the exact formulation with soft margins. The complex instance-specific norms $\|\mathbf{w}^{(i)}\|$ are replaced with scalings of the full norm, by way of coefficients $s_i = \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\|$. Thus Equations 4.2 and 4.3

become:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|} \left(\min_i \frac{y_i(\mathbf{w}^{(i)} \cdot x_i)}{s_i} \right), \tag{4.15}$$

$$\text{where } s_i = \frac{\|\mathbf{w}^{(i)}\|}{\|\mathbf{w}\|}. \tag{4.16}$$

Thus, the problem of the SVM process with soft margins can be reduced to the following equations (take $\xi \equiv 0, b = 0$ to recover the separable approach):

$$(\mathbf{w}, b, \xi, \mathbf{s}) = \arg\min_{\mathbf{w},b,\xi,\mathbf{s}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \tag{4.17}$$

$$\text{such that } \forall i, \frac{1}{s_i}\left(y_i(b + (\mathbf{w}^{(i)} \cdot x_i))\right) \geq 1 - \xi_i. \tag{4.18}$$

Chechik et al note that this program is not in a QP form. In fact, it is non-convex in $\mathbf{w}$ and requires an iterative dual-optimisation procedure to approach a reasonable solution. We first take the s_i to be constant and create the familiar Wolfe dual problem with respect to the Lagrangian multipliers $\alpha_i \in \mathbb{R}^N$, as outlined in §2.4:

$$\alpha = \arg\min_{\alpha} \frac{1}{2}\alpha^T B\alpha - \mathbf{e}^T\alpha, \tag{4.19}$$

$$\text{where } 0 \leq \alpha \leq C, B = \{B_{ij}\} = \frac{y_i y_j}{s_i s_j}(x_i \cdot x_j), \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{4.20}$$

Note here that B, analogous to the class-endowed kernel matrix used in §2.4.1 to solve the usual, full-data case SVM problem, is dependent on our choice of s_i, framing the problem as needing a dual-optimisation procedure; every iteration of $\mathbf{s}$ will produce a different SVM problem to solve. It is important to note that the term $(x_i \cdot x_j)$, which would not usually make sense outside the FDC (see §5.3), must be replaced by the analogous dot-product but where the x_i have been *pre-imputed by zeros*, thus allowing the exclusion of the product terms (by assigning

them zero value) where either x_i or x_j are not present. The optimal margin for classification can then be computed thus:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \frac{y_i}{s_i} x_i, \tag{4.21}$$

again analogously to the usual SVM formulation. Once the minimal value of this optimisation step has been computed, the s_i can be updated by the simple relation:

$$s_i^{(k+1)} = \frac{\|\mathbf{w}^{(i),(k)}\|}{\|\mathbf{w}^{(k)}\|}(s^{(k)}) \tag{4.22}$$

This relation makes sense as the optimal solution $\mathbf{w}$ will inevitably depend continuously on the choice of $\mathbf{s}$ held constant; also, as a result of Equation 4.21, if the minimal value for $\mathbf{w}$ is also a minimum over choices of $\mathbf{s}$, then clearly $s_i^{(k+1)} \equiv s_i^{(k)} \forall i$. The authors reason that this guarantees convergence to some minimal value, albeit sometimes local. Multiple starting points were used over several runs to ensure a good result; they also point out that this method means that the greatest computational burden in computing the B_{ij} can be derived beforehand, as the B is just a vector scaling of the kernel matrix K^0 for the zero-imputed dataset.

4.3 Discussion of GMSVM method

In this section, we discuss the third method mentioned above, as this is the process for which we will attempt to create an analogue with our SVDD classifier. Note firstly that the objective function depends on the datapoints x_i only in terms of their (zero-imputed) dot-products with each other, rendering it as a suitable form to which kernel methods can be applied (see §2.3). The formulation in kernel

space is straightforward, and takes the form of Equations 4.19 and 4.20 being replaced by the following pair of kernel-based equations:

$$\alpha = \arg\min_{\alpha} \frac{1}{2}\alpha^T B\alpha - \mathbf{e}^T\alpha \text{ (as before)},$$

$$\text{where } 0 \leq \alpha \leq C, B = \{B_{ij}\} = \frac{y_i y_j}{s_i s_j} k^0(x_i, x_j), \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{4.23}$$

Here, k^0 refers to the kernel function $k(x, y)$ working only on those features for which both points x and y have data present. We consider those kernels that are based on the inner products of datapoints, that is $k(x, y) = f(x \cdot y)$; thus, the formulation k^0 can be easily achieved by applying k to zero-imputed data, as described above. The reliance of k on the dot product between points in this way ensures that this step is equivalent to ignoring non-present feature elements.

4.3.1 Weighting terms

This method relies on the creation of kernels that can handle missing data. As we will see later in §5.3, this is not a natural property of a kernel space, and accordingly the GM-SVM method provides a way in which kernel spaces can be adequately dealt with. The paper by Chechik et al uses the the following map, effectively from input space to itself but able to deal with missing features:

$$h : \{\mathbb{R}, \text{NaN}\}^d \to \mathbb{R}^d, h_k(x) = \begin{cases} x_k & : \quad x_k \text{ present;} \\ 0 & : \quad x_k \text{ absent.} \end{cases} \tag{4.24}$$

Thus, when the map h is applied to each datapoint in turn, the effect on a kernel based on the dot product will be that if either feature of a given pair is absent, that feature shall be ignored. Chechik et al argue that this approach is different to pre-imputation by zeros (which would produce an identical kernel to that given above)

by virtue of the s_i terms ensuring that the datapoints are weighted correctly. Let us return to the definition of these terms:

$$s_i = \frac{\|\mathbf{w}^{(i)}\|}{\|\mathbf{w}\|}, \text{ where } \|\mathbf{w}^{(i)}\|^2 = \sum_{k: f_k \in Q_{x_i}} w_k^2. \tag{4.25}$$

The major assumption behind this extension into kernel space is that the s_i derived this way will continue to perform the same rôle once the main objective equations have been kernelised, à la Equation 4.23. However, we see above that the definition of $\|\mathbf{w}^{(i)}\|$ is itself dependent on an explicit depiction of the margin vector $\mathbf{w}$, which is not a luxury available to us in kernel-induced feature space. The latter observation is derived from the only meaningful representation of a kernel analogue of $\mathbf{w}$ in feature space (compare the definitions as per §2.4.1):

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \frac{y_i}{s_i} \phi(x_i), \tag{4.26}$$

which itself is intractable: this is the original motivation for the dual problem of the α_i Lagrangian multipliers to begin with. Thus, we find that in general, we cannot hope to provide meaningful scaling terms s_i if the explicit form of the mapping function is not known. However, occasionally it may be the case that it can be found. Let us again consider the quadratic kernel, $k(x, y) = (1 + (x \cdot y))^2$. In a two-dimensional case we have the following explicit form for ϕ:

$$\phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2). \tag{4.27}$$

We now show how the s_i can vary from the linear to the quadratic case. Take a datapoint x_1 with a present feature in f_1 and absent in f_2; we write $x_1 = (x_{11}, \cdot)$. It is trivial to verify that in the linear case:

$$s_1^2 = \frac{w_1^2}{w_1^2 + w_2^2}.$$

We see that ϕ has six dimensions, thus we model our boundary vector $\mathbf{w} \in \mathbb{R}^6$ and define the following:

$$\|\mathbf{w}^{(i)}\|^2 \equiv \sum_{k \in G_i} w_k^2, \tag{4.28}$$

$$\text{where } k \in G_i \iff \{[\phi(x_i)]_k \text{ does not involve any features not in } Q_{x_i}\} \tag{4.29}$$

This seemingly convoluted definition, which itself relies on the explicit form of ϕ, is entirely consistent with the linear case and includes dimensions of ϕ not dependent on any features of x_i (i.e. constants). Under this definition we have:

$$\begin{aligned} \|\mathbf{w}^{(1)}\|^2 &= \sum_{k \in \{1,2,4\}} w_k^2 && (4.30) \\ &= w_1^2 + w_2^2 + w_4^2 \\ \implies s_1^2 &= \frac{w_1^2 + w_2^2 + w_4^2}{w_1^2 + w_2^2 + w_3^2 + w_4^2 + w_5^2 + w_6^2}. && (4.31) \end{aligned}$$

This is plainly different from the linear case mentioned above; the definitions of the w_i are also different, so there is no correspondence between the two definitions.

Adapted input-space method We do, however, find that given a special case the s_i can prove explicitly computable. Suppose the following is true: For kernels based on the dot product $k(x, y) = f(x \cdot y)$ as above, let the vector $\mathbf{w}$ be *restricted to input space*. We now seek its equivalent mapping $\phi(\mathbf{w})$, and perform a similar analysis. In this case, Equation 4.3 becomes:

$$\rho_i(\mathbf{w}) = \frac{y_i(\phi(\mathbf{w}^{(i)}) \cdot \phi(x_i))}{\|\phi(\mathbf{w}^{(i)})\|} \tag{4.32}$$

where, if we continue the above assumption, we can reasonably define the quantity $\mathbf{w}^{(i)}$ itself (compare Equation 4.24) as:

$$(\mathbf{w}^{(i)})_k = \begin{cases} w_k & : \quad x_k \text{ present;} \\ 0 & : \quad x_k \text{ absent.} \end{cases} \tag{4.33}$$

Under the above assumptions, Equation 4.32 is equivalent to:

$$\rho_i(\mathbf{w}) = \frac{y_i(\phi(\mathbf{w}^{(i)}) \cdot \phi(x_i))}{\|\phi(\mathbf{w}^{(i)})\|} \tag{4.34}$$

A similar analysis to the linear case can now be performed if we let:

$$s_i^2 = \frac{f(\|\mathbf{w}^{(i)}\|^2)}{f(\|\mathbf{w}\|^2)}. \tag{4.35}$$

to arrive at an analogous relation to Equation 4.18. We have adapted this case in §5.4; it is subtly different from the usual case where the objective vector ($\mathbf{w}$ here, $\mathbf{a}$ in SVDD) resides in a usually singular point in kernel feature space and has no preimage in the input space. Here, if we assume that $\mathbf{w}$ itself resides in input space This example shows that great care should be taken with definitions of scaling terms such as these, as working in a kernel-induced feature space is a complex extension and scalar values, in general, cannot be expected to behave in the same way when transformed.

4.3.2 Kernel extension

The above examples show that great care should be taken with definitions of scaling terms such as the s_i, as working in a kernel-induced feature space is a complex extension and scalar values, in general, cannot be expected to behave in the same way when transformed. For example, it is also simple to verify that even the

adapted form of the exact method described above fails when the widely-used radial basis function (RBF) kernel is used:

$$k(x, y) = \exp\left(-q\|x - y\|^2\right) \tag{4.36}$$

As this equation is not dependent on the dot product, missing out features is *not* equivalent to pre-imputation with zeros. In fact, even if the 'geometric' distance metric (see §5.3.2 and Equation 5.20) is used, we find that as the RBF kernel has the effect of mapping *all* data onto a hypersphere in feature space of radius 1, the s_i terms have no effect: they merely cancel down to $s_i \equiv \frac{1}{1} \equiv 1$. In general, the approach towards kernel space used in these methods does not take into account the parametrisation of curves and surfaces that would be the result of including free parameters in input space (i.e. missing data without imputation): we hope to make inroads into this with our algorithms developed for the SVDD case. The following chapter will present a method designed to work in purely input space, before outlining the obstacles that will, in general, stand in the way of an analysis of kernel spaces when dealing with missing data.

Chapter 5

Exact centre domain description

5.1 Motivation

In this chapter, we describe methods we have devised to provide a solution to the problem of performing support vector domain description (SVDD) with a dataset with missing data. We take our lead from the studies of Chechik et al [14] on performing an SVM on structurally missing data; as described in §4, the motivation for their method came from a study of fitting data to an optimal margin with respect to the subspace in which each datapoint had present features. This itself was inspired by the challenge of working with *structurally missing* data, where imputation methods may provide spurious, meaningless completions of a dataset in which data may be missing for a good reason, rather than sub-optimal collection methods. In §2.2.7, we observed that SVDD is a similar method to the SVM classification process that instead provides a description of the domain of a dataset; it is a reasonable question, therefore, to ask whether the ideas put forward by Chechik et al about measuring in feature-specific subspaces could be extended

in this case to a similar support-vector based method. In §2.3, we learned about the power of kernel functions, which have the ability to extend classification methods such as these into arbitrarily complex feature spaces, with the 'kernel trick' (§2.3.2) ensuring that under reasonable conditions, the explicit mapping does not have to be itself computed. We can thus identify four major aims of this study. Our system must be able to:

- Perform a one-class domain description of a dataset, in a way in which the inclusion status of novel data can in some way be ascertained, and if necessary pave the way for use of a hybrid method such as DSVC (described in §2.2.8) for multi-class classification problems.

- Deal rigorously with missing data, in a way which avoids imputation methods: that is, it must use all of the data present in input space, without adding or guessing at the data absent from the dataset.

- Work effectively with kernel-induced feature spaces in a way which is rigorous and conforms to the principles described in §2.3, and avoids the need to explicitly address the mapping function ϕ of any given kernel.

- Cancel down in the 'full data case' to the same solution that would be given by a kernel SVDD.

The following chapter will describe the derivation and properties of our first method, which we term our *exact-centre* system. It should be seen as a forerunner to our main algorithm, known as our *dual-optimisation* system, that we will cover later in Chapter 6, and the material in this chapter should therefore be seen as describing the necessary preliminary work to prepare the ground for the latter method.

Section structure In §5.2, we demonstrate a way in which all but the third aim as stated above can be achieved, by considering a simple method of performing and solving a SVDD machine in linear input space only. We then identify the main challenges which arise in generalising this method to non-spherically distributed data by introducing kernel functions in §5.3, and outline ways in which these challenges can be overcome. In §5.4, we demonstrate an algorithm designed to find a centre *in input space* which provides a kernelisable domain description solution with respect to missing data whilst avoiding the use of imputation: this is the method we refer to as our exact-centre method. We test this method in the subsequent sections, with §5.5 showing the first tests we performed on MCAR data and our experiments with different missingness levels and kernel functions; finally, in §5.6 we propose a more rigid comparison structure for this and the following chapters for comparing incomplete datasets side-by-side, and use this structure to analyse our exact-centre system on synthetic data.

5.2 Re-derivation of linear SVDD for GM

5.2.1 Introduction

In the paper of Chechik et al [14] regarding a geometric margin approach to SVM, they argue that their system always provides larger margins in the missing data case, unless the classifier is orthogonal to the missing feature in question, and compare this to the imputation case which underestimates the margin, giving the point with missing data too much weight on the classification process. This is the main motivation for the approach described in this section – since it can be similarly argued that in SVDD, we try and *minimise* the radius of the enclosing sphere

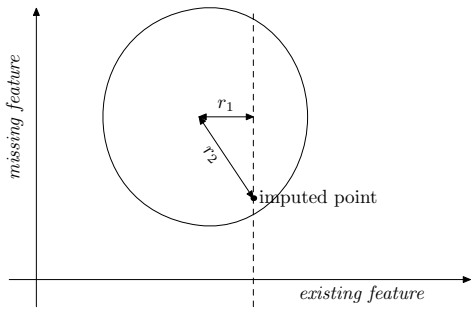


Figure 5.1: SVDD with missing data

(see Figure 5.1); given an imputation of a point with missing data, we find that its distance from the centre is always *larger* than if a geometric margin approach had been used. Thus, the same risk is encountered but with the opposite objective: that if a point is badly imputed it will have too great a distance from the centre, and thus again be seen as too important by a classifier. Using a geometric margin approach would ensure that the distances are only measured from points with respect to the features they have present – Chechik et al describe this as measuring only in the *subspace* of input space for which we have mutual feature presence. We will show that in the linear case, using dimensional information from input space, we can re-formulate the SVDD method to arrive at an optimisable quantity.

5.2.2 Formulation

In this section, we use a method similar to the SVDD method we described in §2.4.2 to investigate the formulation of the linear SVDD in the missing data case. Let $X \in \mathbb{R}^{N\times d}$ be the dataset with missing features, and let $\Pi = \{\pi_{ij}\}$ be the

dataset's unique *presence matrix*: that is, the binary matrix such that:

$$\pi_{ij} = \begin{cases} 1 & \text{if feature } X_{ij} \text{ is present;} \\ 0 & \text{if feature } X_{ij} \text{ is absent.} \end{cases} \tag{5.1}$$

In order to compare the distances of points based on their perpendicular distance, it is necessary to incorporate the matrix Π into an adapted distance metric. Suppose we had a point $x = (x_1, x_2) \in \mathbb{R}^2$ and another point $y = (y_1, \text{NaN})$ with missing data in the second dimension; an ideal distance metric will compare the distance of y from x only in the subspace for which both x and y have features (that is, only the first dimension in this case) – yielding $\|y - x\|^2 = (y_1 - x_1)^2$. In general, we can define the *linear geometric distance*[1] thus:

$$\|\mathbf{a} - \mathbf{b}\|_G^2 = \sum_{j=1}^{d} \pi_{aj}\pi_{bj}(a_j - b_j)^2, \tag{5.2}$$

where π_{aj}, π_{bj} are binary values denoting the presence or absence of the j^{th} dimensions of $\mathbf{a}, \mathbf{b}$ respectively, and the multiplication of them acts as an 'AND' gate: the main part of the summation will be ignored if either $\mathbf{a}_j$ or $\mathbf{b}_j$ is missing. By convention in this definition, we take the value of zero multiplied by an undefined missing value to equal zero; to add rigour, we may envisage both vectors to be pre-imputed (e.g. by zeros) initially. In order to compare distances using a geometric margin approach, we must adapt the problem as stated in Equations

[1]Note this the input-space definition; later in the chapter we will show that another, more general, formulation can yield the same result.

2.26 and 2.27 to the following:

$$(\mathbf{a}, \xi, R) = \arg\min_{\mathbf{a},\xi,R} R^2 + C\sum_{i=1}^{N}\xi_i \tag{5.3}$$

$$\text{such that } \left[\sum_{j=1}^{d}\pi_{ij}(x_{ij}-a_j)^2\right] - R^2 - \xi_i \leq 0, \tag{5.4}$$

$$-\xi_i \leq 0 \forall i. \tag{5.5}$$

Note the only difference between these equations and the abovementioned linear formulation of SVDD in the full data case is that we have included the values of π_{ij} in the constraints; everything else is left the same, therefore we still seek a 'ball' around a unique, full-data centre point $\mathbf{a}$ and a dimensionally homogeneous radius R. Intuitively, in terms of the distances between this centre point and data with missing features, the 'ball' will have more of a hypercube structure. We construct a Lagrangian for this system:

$$\Lambda(\mathbf{a}, \xi, R, \alpha, \beta) = R^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[R^2 + \xi_i - \sum_{j=1}^{d}\pi_{ij}(x_{ij}-a_j)^2\right] - \sum_{i=1}^{N}\beta_i\xi_i. \tag{5.6}$$

Proceeding as in §2.4.2 we take derivatives with respect to the primal variables to derive new conditions:

$$\forall i \frac{\partial\Lambda}{\partial\xi_i} = 0 \implies C = \alpha_i + \beta_i \implies \alpha_i \in [0, C] \text{ (as before);} \tag{5.7}$$

$$\frac{\partial\Lambda}{\partial R} = 0 \implies \sum_{i=1}^{N}\alpha_i = 1 \text{ (as before);} \tag{5.8}$$

$$\forall k \frac{\partial\Lambda}{\partial a_k} = 0 \implies a_k = \frac{\sum_{i=1}^{N}\alpha_i\pi_{ik}x_{ik}}{\sum_{i=1}^{N}\alpha_i\pi_{ik}}. \tag{5.9}$$

Equation 5.9 shows that this system is more complex, as we can no longer simply express $\mathbf{a}$ as a weighted sum of the x_i as in the full data case. We put these

relations back into Equation 5.6 to derive our objective function:

$$\begin{aligned}
\Lambda &= R^2 + C\sum_i \xi_i - C\sum_i \xi_i - \sum_i \alpha_i \left[R^2 + \xi_i - \sum_j \pi_{ij}(x_{ij} - a_j)^2 \right] \\
&= R^2 + \sum_i \alpha_i \left\{ \left[\sum_j \pi_{ij}(x_{ij} - a_j)^2 \right] - R^2 \right\} \\
&= \sum_i \alpha_i \left\{ \sum_j \pi_{ij}(x_{ij} - a_j)^2 \right\} \\
\Lambda &= \sum_i \alpha_i \left\{ \sum_j \pi_{ij}(x_{ij})^2 - 2\sum_j \pi_{ij} x_{ij} a_j + \sum_j \pi_{ij} a_j^2 \right\}. \qquad (5.10)
\end{aligned}$$

We can now introduce the definition of the a_j as per Equation 5.9, to arrive at our optimisable quantity:

$$\Lambda = \sum_{j=1}^{d} \Lambda_j \text{ where } \Lambda_j = \sum_{i=1}^{N} \alpha_i \pi_{ij} x_{ij}^2 - \frac{\left(\sum_i \alpha_i \pi_{ij} x_{ij}\right)^2}{\sum_i \alpha_i \pi_{ij}}. \qquad (5.11)$$

The form of this latter equation is similar to that used in statistics to compute variance in a dataset. Suppose we have a probability mass function (p.m.f.), defined over a list of one-dimensional datapoints x_i, with corresponding probability masses p_i; if $\sum p_i = 1$, as would usually be the case for a valid p.m.f., we define the variance thus:

$$\text{Var } X|_{p_i} = \sum_i p_i x_i^2 - \left(\sum_i p_i x_i \right)^2 \qquad (5.12)$$

Suppose, however, that $\sum p_i \neq 1$; to maintain the same units we introduce a normalising factor. Define the *generalised variance* as:

$$\text{VarG } X|_{p_i} = \frac{\sum_i p_i x_i^2}{\sum_i p_i} - \left(\frac{\sum_i p_i x_i}{\sum_i p_i} \right)^2 \qquad (5.13)$$

Returning to the definition of Λ above, if we let $w_j = \sum_{i=1}^{N} \alpha_i \pi_{ij}$, we have:

$$\Lambda = \sum_{j=1}^{d} w_j \text{VarG } X_j|_{\alpha_j : \pi_{ij}=1} \tag{5.14}$$

where here X_j denotes the j^{th} dimension (column) of the dataset X. That is, if we take each dimension j in turn, we can create a one-dimensional probability mass function over those points in that dimension of input space i where $\pi_{ij} = 1$, taking probability mass of α_i at each present point. We use the generalised variance since, in general, the sum of these points will not be unity outside the FDC, since some data in each dimension will be missing (and $\sum \alpha_i = 1$ only over all datapoints). For example, if:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ \cdot & x_{22} & x_{23} \\ x_{31} & \cdot & x_{33} \\ x_{41} & x_{42} & \cdot \end{pmatrix}, \alpha = \frac{1}{10} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \tag{5.15}$$

then Λ is calculated thus:

$$\Lambda = \frac{8}{10} \text{VarG } X_1 + \frac{7}{10} \text{VarG } X_2 + \frac{6}{10} \text{VarG } X_3. \tag{5.16}$$

This method makes intuitive sense in terms of the behaviour of support vectors in linear space. To maximise Λ, we must choose α such that the weighted sum of the projections of the dataset onto each axis, adorned with point-masses designated by the coefficients α_i, is maximal. When using the classical FDC form of SVDD, we find that those X_{ij} for which α_j is largest are the support vectors and those outside the sphere: thus maximising the variance of a p.m.f. when the data is projected onto each axis.

5.2.3 Objective function

In order to be able to maximise the quantity Λ effectively, it is an advantage if we can provide an optimisation solver with a gradient function. We return to the original, data-based definition of Λ as given in Equation 5.11:

$$\Lambda = \sum_{j=1}^{d} \Lambda_j \text{ where } \Lambda_j = \sum_{i=1}^{N} \alpha_i \pi_{ij} x_{ij}^2 - \frac{\left(\sum_i \alpha_i \pi_{ij} x_{ij}\right)^2}{\sum_i \alpha_i \pi_{ij}}.$$

Since we are specifying the problem in terms of the coefficients α_i, we compute the corresponding gradient of each of the Λ_j, with a view to their summation once this has been derived. For all $k \in \{1 \dots N\}$:

$$\frac{\partial \Lambda_j}{\partial \alpha_k} = \pi_{kj} x_{kj}^2 - 2\pi_{kj} x_{kj} \left(\frac{\sum \alpha_i \pi_{ij} x_{ij}}{\sum \alpha_i \pi_{ij}} \right) + \pi_{kj} \left(\frac{\sum \alpha_i \pi_{ij} x_{ij}}{\sum \alpha_i \pi_{ij}} \right)^2 \tag{5.17}$$

$$\implies \frac{\partial \Lambda_j}{\partial \alpha_k} = \pi_{kj} \left(x_{kj} - \frac{\sum \alpha_i \pi_{ij} x_{ij}}{\sum \alpha_i \pi_{ij}} \right)^2. \tag{5.18}$$

The second term here can be equated with the centre of mass of datapoints along a dimension j with their features present; in particular, this gradient function computes the distance of x_{kj} away from this dimensional centre of mass. Since the constraints on α given by Equations 5.7 and 5.8 are the same as in the SVDD case, an optimisation solver, for example MATLAB's Optimization Toolbox function `fmincon`, can now be used to find a constrained minimal value of the additive inverse of the value $\Lambda = \sum \Lambda_j$.

Comments In itself, this system thus provides an effective – if rather computationally intensive – way of deriving an accurate weighting vector α for an SVDD in linear (input) space, with geometric margins: that is, the values of any missing data are not 'guessed' at any point during the calculation, and in §5.5 we show that

on simple problems, this method can be effective. Thus, on its own the method is a reasonable self-contained algorithm for dealing with spherically-distributed data. However, its utility is limited given that the exact values of datapoints, along with their presence/absence status, must be known; this precludes the use of kernel methods, in which the full-data case SVDD method gains its degree of versatility. Unfortunately, the processes necessary in moving from input space to a generalised feature space are not simple: in the next section, we describe the obstacles which must be overcome to gain a rigorous algorithm which does not need to address feature-space values explicitly.

5.3 Development of kernelisable system

5.3.1 Introduction

If we are to produce a system which caters for non-spherically distributed data whilst maintaining a geometric radius formulation, a few observations must be made. Firstly, in the full data case, domain description, along with many other classification techniques, is rarely used in the linear formulation, relying instead on the kernel methods introduced in §2.3 and described in detail in §2.3.5 and §2.4.2. Secondly, that the very formulation of a rigorous inner product $k(x, y) = \phi(x) \cdot \phi(y)$, whether in input space or feature space, requires use of products between corresponding features of the datapoints involved, since:

$$(x \cdot y) = \sum_{k=1}^{d} x_k y_k.$$

If we wish to avoid using any imputation methods on missing data expressions such as $x_k y_k$ must be dealt with properly in cases where, say, x_k is present and

y_k is absent. Thirdly, that Mercer's theorem of function analysis [68] implies that the mapping ϕ always exists in a well-defined fashion if $K = \{K_{ij}\}$ is positive definite. The following observations lead to the following theorem.

Theorem There does not exist a suitable positive definite kernel function

$$k(x, y) = \phi(x) \cdot \phi(y),$$

which uses all dimensions of input data in some way, and can be applied to a dataset with missing features for a use in a kernelisation of a classification technique using a geometric radius approach; that is, avoiding the use of pre-imputation of the dataset.

Proof By contradiction; suppose there were a positive definite kernel $k(x, y)$. Then there exists a well-defined mapping ϕ such that $k(x, y) = \phi(x) \cdot \phi(y)$; let us assume ϕ has dimension D[2]. Thus, for a given pair of datapoints (x, y), $k(x, y) = \sum_{t=1}^{D} \phi_t(x)\phi_t(y)$ for some convergent series of functions $\phi_t(x)$. Either we are in the full data case (i.e. all features of both x and y are well defined), a degenerate no-data case (where some features are defined for neither x nor y), or there exists some $k_0 \in \{1 \dots d\}$ such that x_{k_0} (without loss of generality) exists and y_{k_0} is absent. Let T be the subset of $\{1 \dots D\}$ such that the functions $\{\phi_t(x) : t \in T\}$ are dependent on the k_0^{th} dimension. Either T is empty (in which case the k_0^{th} dimension is never used, violating the principle of full use of the present data in input space) or each of the corresponding functions $\{\phi_t(y) : t \in T\}$ are functions of a non-existent variable, which cannot be made well-defined without imputation

[2] D is allowed to be infinite without loss of generality, since ϕ must converge to a unique value $\forall x \in \mathbb{R}^d$ in the input space.

of a value for y_{k_0}. Thus, a suitable ϕ cannot exist to apply to all data, and K cannot itself be positive definite.

Comments As well as the implication that any candidate K cannot be positive in anything but the full data case, thus implying a non-convex problem and precluding the use of many standard methods for solving a quadratic program (see §2.4) with a positive definite kernel, this theorem also shows us that a suitable ϕ cannot exist. We return to the quadratic kernel for an example:

$$K(x, y) = (1 + x \cdot y)^2$$

Here, as mentioned in previous chapters, an explicit depiction of ϕ is well known. For example, for $x \in \mathbb{R}^2$:

$$\phi(x) = \{1, \sqrt{2}x_1\sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2\} \tag{5.19}$$

Dimensions 3, 5 and 6 are meaningless here if x_2 is absent. Indeed, the term $\sqrt{2}x_1x_2$, despite providing information about x_1, is rendered meaningless itself if this is the case. Indeed, even in linear space where a statement such as:

$$\phi(x) = \{x_1, \text{NaN}\}$$

is somewhat more 'well-defined' in terms of the lack of feature products, it still cannot satisfy the condition of $\phi(x) \cdot \phi(y)$ being well-defined in any case where a y_2 is present.

Alignment This result is more significant than it initially seems, since it precludes the use of standard methods for aligning data such as kernel PCA and kernel whitening (§2.3.4), as a consequence of their reliance on an inner-product

based formulation or equivalent. It also makes the common practice of centring a kernel in feature space (§2.3.3) difficult, since the theory behind this method assumes a priori that means can be taken over all present data in a dimension. Intuitively, it also makes sense that a method such as (kernel) PCA, when used naïvely, cannot be meaningful when incomplete data is studied. Take the example of a linear PCA method applied to a dataset of two dimensions: we may visualise incomplete data as being infinite lines parallel to one of the axes, and use a form such as $x = (x_1, \text{NaN})$ to describe entries in a dataset. The function of a linear PCA is essentially to find a rotation of the axes such that the principal axis lines up with the direction of greatest variance in the data; even if this solution did exist, the images of the incomplete data after mapping would describe skewed lines not parallel to the axes, with no simple way of describing their orientations in the usual $|X| = (N \times d), x_{ij} \in \{\mathbb{R}^d, \text{NaN}\}$ dataset matrix form. This problem is exacerbated in the kernel PCA case, where the images of these lines may describe curves parametrised by complex functions.

5.3.2 Non-separability of distance metric

In §5.2 we see that in linear space, the concept of introducing geometric margins for missing data to the SVDD system is a reasonable one, and have shown the simple quantity that needs to be maximised in this case. However, this system depends greatly on being able to explicitly address data in feature space, which is too strong an assumption for successful deployment of a kernelised system. In linear space, missing data can be simply visualised as lines or planes parallel to the axes, and can be described using a simple matrix Π to designate the subspace over which features for any given datapoint are to be considered. The ease of

formulation of a method in the linear case relies on the separable form of the distance metric. For example, for $\mathbf{x} \in \mathbb{R}^2$:

$$\|\mathbf{x} - \mathbf{a}\|^2 = \pi_1(x_1 - a_1)^2 + \pi_2(x_2 - a_2)^2 = \sum_j \pi_j f_j(x_j). \tag{5.20}$$

This equation shows that the usual distance metric has a natural analogue in linear space by introducing the linear binary weights π_j to include or exclude datapoints as necessary; the form remains separable in the input variables. We soon see, however, that naïvely mapping the lines and planes created by this method into even a simple kernel space makes the system much more complex.

Example Consider again the kernel mentioned in §2.3.3 in the context of kernel centring; it has the following form:

$$\phi(x_1, x_2) = (1, x_1, x_2, x_1 x_2). \tag{5.21}$$

Suppose we are to consider the mapping of a point $x = (x_1, x_2)$ into this kernel space:

$$\|\phi(\mathbf{x}) - \mathbf{a}\|^2 = (1 - a_1)^2 + (x_1 - a_2)^2 + (x_2 - a_3)^2 + (x_1 x_2 - a_4)^2 \tag{5.22}$$

Unlike the form of Equation 5.20, this form is not linearly separable in the input variables (x_1, x_2); the fourth term maps lines to quadratic curves, thus precluding the simple solution of a presence matrix Π.

5.3.3 Distance matrices

In order to discuss the problem of finding distances between incomplete data within a kernel-induced feature space, we first consider the concept of a *distance*

matrix. Let the dataset X be of dimension $N \times d$, with entries $x_{ij} \in \{\mathbb{R}, \text{NaN}\}$. Define the $(N \times N)$ kernel matrix K in the usual way (as in Equation 2.32):

$$K = \{K_{ij}\} = \{\phi(x_i) \cdot \phi(x_j)\} = \{k(x_i, x_j)\}$$

In the full data case, there is a function we can apply to a matrix K to recover a matrix of distances between the datapoints. Let:

$$D = \{D_{ij}\}, \text{ where } D_{ij} = K_{ii} + K_{jj} - 2K_{ij} \tag{5.23}$$

It now holds that the entries of the $(N \times N)$ matrix D denote the squares of the inter-point distances in feature space:

$$\begin{aligned} D_{ij} &= k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j) && (5.24) \\ &= \phi(x_i) \cdot \phi(x_i) + \phi(x_j) \cdot \phi(x_j) - 2\phi(x_i) \cdot \phi(x_j) \\ &= \|\phi(x_i) - \phi(x_j)\|^2. && (5.25) \end{aligned}$$

This relation is no longer true outside the full data case; however, there are linear formulations allowing a similar object to be constructed, as shown below.

5.3.4 Well-behaved linear kernel

Despite the limitations presented in §5.1, it *is* possible to create a matrix L which has the property of being "well-behaved" in linear space. Using the same definition of the presence matrix Π with entries π_{ij} as in §5.2, consider the following distance matrix:

$$\Delta^G = \{\delta^G_{ij}\} = \sqrt{\sum_{k=1}^{d} \pi_{ik}\pi_{jk}(x_{ik} - x_{jk})^2} \tag{5.26}$$

This is an intuitive way to express distances in linear space, with distances between points taken only along the subspace of input space for which both points share dimensions. Compare this equation with the definition of δ_{ij} in the full data case above, the inter-point distances of which take equal or greater value than those achieved using geometric radii. With this in mind, the question arises as to whether we can construct a matrix K^G such that, when Equation 5.23 is applied to it, a geometric distance matrix results. Consider the following general formulation of a *linear* inner product:

$$K^L = \{K^L_{ij}\} = \sum_{k=1}^{d} \mu(x_{ik}, x_{jk}) \tag{5.27}$$

Clearly, with full data, the usual inner product is produced by $\mu(x, y) = xy$. Furthermore, this is a dimension-separable formulation since each expression depends only on one dimension at a time. Given Π as defined above, if we instead let μ be defined as:

$$\mu(x_{ik}, x_{jk}) = \frac{1}{2}\left(x^2_{ik}x^2_{jk} - \pi_{ik}\pi_{jk}(x_{ik} - x_{jk})^2\right), \tag{5.28}$$

then we find that:

- When $\pi_{ij} \equiv 1$, $\mu(x_{ik}, x_{jk}) = x_{ik}x_{jk}$ as with the usual inner product;
- If we compute $D_{ij} = K_{ii} + K_{jj} - 2K_{ij}$ where K is constructed with this definition of μ, we find that the distance matrix produces the required inter-point distances in linear space as described above.

Although it may be tempting to think it is possible to perform a method such as SVDD with this 'geometric' kernel, we soon find that the results it produces outside the full data case can be spurious. The kernel ceases to be positive definite

when missing data is introduced, despite the well-defined nature of the distance matrix D on which it is based. In fact, the kernel above was somewhat arbitrarily chosen: since there are less degrees of freedom in a distance matrix than a kernel matrix, in principle there are many K we could choose to satisfy the relation given in Equation 5.23. Although it may be computationally possible to use this kernel in an SVDD computation, the results will be meaningless; if a kernel is not positive definite it does not have an equivalent mapping function ϕ. The same problem arises with the radial basis function (RBF) kernel matrix, used widely in literature:

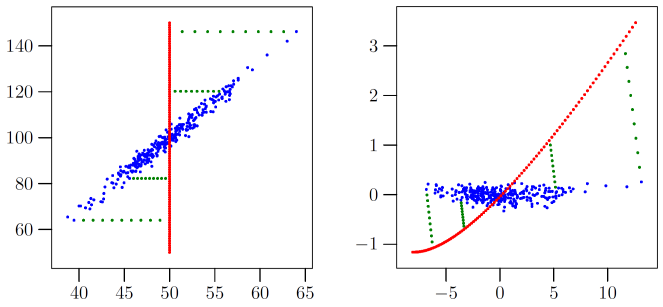
$$k(x, y) = \exp\left(\frac{-1}{2\sigma^2}\|x - y\|^2\right) \tag{5.29}$$

This kernel, uniquely among standard mappings, is not directly based on the dot product but instead on a distance measure. If we replace the term $\|x - y\|^2$ with a geometric measure as per Equation 5.26, we find the same problem: as soon as missing data is introduced, the kernel ceases to be positive definite. As explained above, this result is actually quite expected, since we have shown that it is not reasonable to expect a mapping function ϕ to exist which can emulate a geometric margin approach exactly without assuming some form of imputation. These examples show that the task of creating a classification method to work in a projected space is non-trivial and cannot rely on using standard methods.

5.3.5 Well-defined kernel distance in non-FDC

As a means of capturing exactly what is meant by a geometric distance, let us return to the linear case, as given in §5.2 and Equation 5.2. The approach of comparing features only in the subspaces for which both vectors have each feature present is motivated by a desire to avoid imputation; this, in turn, meaning we can

avoid spurious inferences, which may cause points with missing data to act too strongly upon a classifier. As we have seen, in the linear case this is achieved by taking differences of feature values only where both vectors have features present. In two dimensions, this may be visualised as the minimum distance from a point to a line. In Figure 5.2(a), this is illustrated: the green lines show the contours



(a) Before (b) After

Figure 5.2: Data completions: (a) linear space; (b) quadratic feature space

along which distance has been measured from the red line – representing a data-point with missing feature f_2 – to various points in the cluster of blue data of full features. Clearly, if the data were transformed orthogonally, this would remain the case; however, if the data underwent some non-orthogonal or even non-linear transformation, as demonstrated in Figure 5.2(b), the points in linear space representing the 'optimal completions' along the red line are no longer guaranteed to be optimal in the transformed case. However, it does demonstrate that another point *does* exist with such an optimal property, and therefore that it is reasonable to ask the location of this point.

Optimisation-based distance metric Consider the following implicitly defined distance between two points in linear space. For $\mathbf{x}$ with features $Q_x \subseteq \{1 \dots d\}$

and $\mathbf{y}$ with features $Q_y \subseteq \{1 \dots d\}$ we define the linear *geometric distance* (in this context) as:

$$\|\mathbf{x} - \mathbf{y}\|_G^2 \equiv \min \left\{ \sum_{k=1}^{d} (x_k - y_k)^2 : \begin{array}{l} x_k \in \{1 \dots d\} \setminus Q_x, \\ y_k \in \{1 \dots d\} \setminus Q_y \end{array} \right\}. \tag{5.30}$$

For example, consider the following pair of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^4$:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1 \\ . \\ y_3 \\ . \end{pmatrix}. \tag{5.31}$$

Here, we see $Q_x = \{1, 2\}, Q_y = \{1, 3\}$, thus the definition here of the geometric distance between the two vectors will be:

$$\|\mathbf{x} - \mathbf{y}\|_G^2 \equiv \min_{x_3, x_4, y_2, y_4} \sum_{k=1}^{d} (x_k - y_k)^2 \tag{5.32}$$

In this linear case, as expected, we have the optimal solution of cross-imputation of values (here y_2 would be optimally imputed as $y_2 = x_2$, etc), which recovers Equation 5.2 as required, since the solution $x_k = y_k$ makes the summation zero for this term if either feature x_k or y_k is missing. However, we now have a form which is possible to kernelise. We note that, for a kernel mapping function ϕ, by the definition of its implicit kernel function k, we have:

$$\|\phi(x) - \phi(y)\|^2 \equiv k(x, x) + k(y, y) - 2k(x, y). \tag{5.33}$$

It is natural, therefore, to provide a kernelised analogue of Equation 5.30:

$$\|\phi(x) - \phi(y)\|_G^2 \equiv \min \left\{ \sum_{l=1}^{D} (\{\phi(x_1 \dots x_k)\}_l - \{\phi(y_1 \dots y_k)\}_l)^2 : \begin{array}{l} x_k \notin Q_x, \\ y_k \notin Q_y \end{array} \right\}. \tag{5.34}$$

The sum within the braces simplifies as a result of relation 5.33, to give an analogous, generalised definition of Equation 5.30. With respect to a suitable mapping function ϕ, we thus define the feature space *geometric distance* as:

$$\|\phi(x) - \phi(y)\|_G^2 \equiv \min \left\{ k(x,x) + k(y,y) - 2k(x,y) : \begin{array}{c} x_k \notin Q_x, \\ y_k \notin Q_y \end{array} \right\}. \quad (5.35)$$

As an example, we propose a similar problem to that visited in §2.3.3, with two vectors defined as:

$$\mathbf{x} = \begin{pmatrix} 2 \\ 3 \\ \cdot \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 5 \\ \cdot \\ 7 \end{pmatrix}.$$

The kernel map we shall consider is that of the *quadratic* kernel, $k(x,y) = (1 + (x \cdot y))^2$. Note first that in the linear case, the geometric distance has an obvious solution based on cross-imputation:

$$\mathbf{x}_1 = \begin{pmatrix} 2 \\ 3 \\ 7 \end{pmatrix}, \mathbf{y}_1 = \begin{pmatrix} 5 \\ 3 \\ 7 \end{pmatrix},$$

with $\|\mathbf{x}_1 - \mathbf{y}_1\|^2 = (2-5)^2 = 9$. To demonstrate that this trivial answer to the optimisation problem 5.30 is not applicable in the kernel case, we compute the feature-space distance between the two vectors using Equation 5.33:

$$\|\phi(x_1) - \phi(y_1)\|^2 \equiv 3969 + 7056 - (2 \cdot 4761) = 1503.$$

However, if we had used the following imputation instead:

$$\mathbf{x}_2 = \begin{pmatrix} 2 \\ 3 \\ 7.27804 \end{pmatrix}, \mathbf{y}_2 = \begin{pmatrix} 5 \\ 2.56119 \\ 7 \end{pmatrix},$$

then a more optimal distance is produced:

$$\|\phi(x_2) - \phi(y_2)\|^2 \equiv 4484.95 + 6651.98 - (2 \cdot 4848.31) = 1440.31.$$

The difficulty arises from the fact that we may not address the structure of ϕ explicitly: the mapping function describes sometimes complex (and not explicitly known) relations between input variables and their behaviour in feature space, but does not allow completely free movement; we may only vary the parametrisations of the feature-space curves with respect to the variables in *input space* for which x and y have missing features. The best we can hope for is that an efficient optimisation procedure succeeds in finding the minimum kernelised geometric distance for every pair of vectors given. To arrive at an optimal centre point using this process is quite possible, as we shall see in the next section; it does, however, re-cast the problem as one of a different order of computational complexity due to this extra step of optimisation.

5.4 Exact centre-based method

In this section, we show that it is possible to create a system somewhat separate to those methods used in classical SVDD, which nevertheless produces a meaningful centre of a dataset of arbitrary dimension, such that this centre point, given a good optimisation algorithm, will conform to similar constraints with respect to a dataset and a kernel function.

5.4.1 Formulation

We consider only hard margins first, and only kernel functions based directly on the dot product, $k(x, y) = f(x \cdot y)$; we comment on other cases below. We also assume, prior to performing the classification process, that the data is in a linear-missing form (X, Π) - that is, the pattern of missing data can be simply described by a binary presence matrix Π of the same size as the dataset X. Using the definition of the geometric margin in kernel feature space given in Equation 5.35, we begin with the following problem formulation:

$$\mathbf{a} = \arg\min_{\mathbf{a}} R(\mathbf{a}),$$
$$\text{where } R(\mathbf{a}) = \max_i \|\phi(x_i) - \phi(\mathbf{a})\|_G^2. \tag{5.36}$$

Note the immediate differences between this formulation and that given in Equation 2.50: instead of optimising over a list of weighting coefficients $\{\alpha_i\}$, we seek an input-space solution directly. Also, it is not yet obvious how to extend this method to soft margins, as neither a pay-off parameter C nor corresponding error margins ξ_i are present here. This is also an *unconstrained* problem: there are no limitations on the values our parameter $\mathbf{a}$ may take, unlike the previous formulations which had very specific restrictions on the values α_i could take, derived via the dual of the initial problem.

5.4.2 Value-based algorithm

A straightforward way of minimising $R(\mathbf{a})$ as given in Equation 5.36 can be found via a simple algorithm, detailed in Algorithm 2. Two loops of optimisation are required, since this describes only a single run given a candidate vector for $\mathbf{a}$. Despite this computational expense, however, this algorithm has been shown to

work reasonably well on simple kernel mapping functions. In §5.5 and §5.6 we show the situations for which even a relatively naïve approach such as this can produce pleasing results. As we have two unconstrained optimisation algorithms, a program such as MATLAB's `fminunc` may be used. When this program is run, a warning message will usually show to say that the 'large-scale' method could not be used, as we have not provided any gradients explicitly anywhere in the above algorithm. To remedy this, the following subsection will show a way of refining the above algorithm slightly to ease the computational burden on the optimisation process.

5.4.3 Gradient-based algorithm

In the above algorithm, we see that the following micro-set of data is produced for each i:

$$S = \begin{pmatrix} x_{i1} & x_{i2} & \dots & x_{id} \\ a_1 & a_2 & \dots & a_d \end{pmatrix} \tag{5.37}$$

where we assume that some of the x_{ij} will be missing; this matrix is recomputed, therefore, for every completion vector $\mathbf{c} \in \mathbb{R}^{h_i}$ we choose to allow this matrix to contain full data, and thus making the notion of inter-point distance make sense. It is reasonable, then, to ask how this inter-point distance, denoted above as d, changes with our choice of $\mathbf{c}$, the intention being to find an expression for the gradient vector $\nabla_{\mathbf{c}}(d)$. As mentioned above, we proceed using kernels based on the dot product, $k(x, y) = f(x \cdot y)$. The distance between the datapoint x_i and centre point $\mathbf{a}$, in the full data case, can then be expressed as:

$$d_i = \|x_i - \mathbf{a}\|^2 = f(x_i \cdot x_i) + f(\mathbf{a} \cdot \mathbf{a}) - 2f(x_i \cdot \mathbf{a}). \tag{5.38}$$

Algorithm 2 Value-based algorithm

Require: candidate centre vector $\mathbf{a} \in \mathbb{R}^d$

Initialise $\mathbf{v}_a \in \mathbb{R}^N$

Let $\mathbf{h} \in \mathbb{R}^N = \{h_i\} \leftarrow$ (number of present features in x_i)

for $i \in \{1 \dots N\}$ **do**

Initialise $S_{a,i} \in \mathbb{R}^{2 \times d}$ with: $S_{1j} \leftarrow x_{ij}$, $S_{2j} \leftarrow a_j$

if $h_i = 0$ **then**

$d \leftarrow k(x_i, x_i) + k(\mathbf{a}, \mathbf{a}) - 2k(x_i, \mathbf{a})$

else

Initialise $\mathbf{c} \in \mathbb{R}^{h_i}$

repeat

Complete S with values from $\mathbf{c}$

$K(\mathbf{c}) \in \mathbb{R}^{2 \times 2} \leftarrow K_{i_1,i_2} = k(S_{i_1}, S_{i_2})$

$d(\mathbf{c}) \leftarrow K_{11} + K_{22} - 2K_{12}$. Optimise this quantity with respect to $\mathbf{c}$

until optimiser convergence in $\mathbf{c}$

end if

$v_i \leftarrow d$

end for

With respect to the given candidate $\mathbf{a}$, the dataset's optimised maximal distance is now $\max_i(v_i)$

Taking derivatives with respect to each spatial dimension l, and using the chain rule, we find:

$$\frac{\partial d_i}{\partial x_{il}} = 2x_{il}f'(x_i \cdot x_i) - 2a_l f'(x_i \cdot \mathbf{a}). \tag{5.39}$$

Thus, provided we have a kernel based on the dot product in a way whose derivative is readily calculable, we can incorporate useful information about the direction of descent into Step 7 of the algorithm above, when optimising for $\mathbf{c}$. Furthermore, a gradient for the outer optimisation loop can also be computed via techniques of automatic differentiation. In practice, we have seen that the approaches with and without gradient information always converge to the same result, but the approach *without* this information is many times faster; however, it is envisaged that for more complex kernels, this gradient information may well make the system more robust to global minima, as it allows a large-scale approach.

5.4.4 Comments

This method provides a simple process towards an input-space kernel-based centre of a dataset; however, it has a few obvious drawbacks. The first is its high computational complexity when given a dataset of high dimension: this sort of dataset is a major factor governing the use of kernel methods, since the kernel matrix need, in general, only be computed once, whereafter the dimension of the dataset X need not matter. Secondly, and related: as mentioned in §5.3.1, important 'data-herding' techniques such as centring and whitening have no obvious analogue when used on datasets with missing features, forcing us to work in the highly restrictive case where a presence matrix can be directly appended to the data, instead of being able to use pre-alignment methods. Thirdly, the parameter C, allowing for soft margins, has disappeared completely: the above algorithm works on hard

margins only. An automatic differentiation package could be used to optimise, for example, the second- or third- highest deviation; however, it is obvious even in a one-dimensional setting that these functions may easily end up jagged and non-differentiable in multiple places as a result. Fourthly, no provision has been made for kernels not based around the dot product: a similar analysis was conducted for the RBF kernel, but it was found that this kernel's easily-derived property of mapping every datapoint onto a hypersphere of radius 1 in feature space caused serious problems to the application of this method.

5.5 Preliminary tests

In order to test the viability of the system we have described above, various preliminary tests were conducted on first the popular 'iris' dataset due to Fisher, and then random "clusters" of data (that is, random i.i.d. normally distributed with a given mean and equal dimension variance) to ensure the efficacy in these cases. The following outlines the results of these tests and compares them to imputation followed by 'usual' classification methods. These results are included primarily for completeness; in §5.6 we will describe a more organised structure to allow full comparisons of our method to imputation methods. The following results are less rigidly organised, as they were conducted before the standard process was decided; in certain cases, as we shall see, they do nevertheless produce useful results, and are interesting in that when the MCAR type of randomness is applied in this way, our methods are not expected to perform that much better than imputation methods as they have a structurally-missing paradigm behind their formulation.

Imputation schemes In both the following preliminary tests and the more complete structure of the full investigations, as well as later in Chapter 6 where we test our full method on synthetic data, the following three imputation schemes will be compared as standard:

- **Zeros:** The dataset will be simply imputed with zeros (denoted in the below table as 'Imp 0')
- **Means:** The dataset will be filled in with the feature-specific means of each dimension (denoted below as 'Imp μ').
- **Nearest Neighbours:** A nearest neighbour scheme as used in the paper by Chechik et al [14] is applied, with the parameter K denoting number of nearest neighbours set to a value of $K = 3$ (denoted 'Imp NN').

5.5.1 Iris data, variable missingness ratio

In Tables 5.1 and 5.2, we show the preliminary results derived using the exact-centre method of §5.4 and employing the linear kernel: that is, operating in input space. Tests were performed with the Iris dataset, which has size 150×4 and has three constituent classes, each being 50 datapoints in size. The *hard margin* formulation was used: that is, no points were allowed outwith the boundary. The aim of this preliminary test was to vary a missingness ratio: the iris dataset was 'made incomplete' with a missing-completely-at-random (MCAR) algorithm, in five different cases where the proportion of omitted features was $\{10\%, 20\%, 30\%, 40\%, 50\%\}$ of the full set. These tables compare our method to the three imputation methods as outlined above when applied to these artificially incomplete datasets, and show that in the linear case our method finds a smaller

Table 5.1: Exact centre, linear kernel

Class	Missing Ratio	XC method	Imp. 0	Imp. μ	Imp. NN
1	10	**1.5586**	2.0639	1.9785	1.9786
2	10	**1.5574**	1.7586	1.7586	1.7586
3	10	**1.9622**	2.2924	2.2923	2.2924
1	20	**1.6529**	1.6855	1.8144	1.8144
2	20	**1.6029**	1.626	1.6442	1.6504
3	20	**1.3174**	1.8369	1.8184	1.842
1	30	**1.2913**	1.7745	1.7459	1.746
2	30	**1.3617**	1.75	1.7053	1.7514
3	30	**1.6755**	2.0601	1.967	1.9173
1	40	**1.3092**	1.6953	1.3855	1.3671
2	40	**1.2952**	1.4529	1.6598	1.8608
3	40	**1.5052**	1.9937	1.9417	2.0201
1	50	**1.4691**	1.7894	1.5603	1.6853
2	50	**1.1536**	1.4873	1.2492	1.5665
3	50	**1.7153**	2.0911	1.759	1.9208

Table 5.2: Exact centre, quadratic kernel

Class	Missing Ratio	XC method	Imp. 0	Imp. μ	Imp. NN
1	10	8.52	7.8535	7.4214	**7.3579**
2	10	**3.7225**	3.7883	3.7883	3.7884
3	10	**7.2694**	7.4911	7.4133	7.4131
1	20	6.0789	6.9867	**6.0122**	6.5323
2	20	3.5618	**3.374**	3.5127	3.6858
3	20	7.7284	5.553	**5.2482**	5.5505
1	30	**5.2783**	7.1164	6.8612	6.8612
2	30	**3.1434**	3.7927	3.9657	3.836
3	30	**5.4948**	5.9759	5.8383	6.284
1	40	**4.8038**	5.6856	5.6793	5.2002
2	40	**2.7277**	3.8197	3.6507	4.5012
3	40	**4.9921**	5.4823	5.4478	5.7156
1	50	**5.1435**	5.7873	6.2021	6.6363
2	50	**2.8192**	3.0824	2.9077	3.1869
3	50	**4.7889**	5.8125	5.6753	6.0724

descriptive radius for this dataset in all cases. Table 5.2 shows analogous results using the quadratic kernel $k(x, y) = (1 + x \cdot y)^2$ instead, and we see some cases do not out-perform imputation: we return to this choice of kernel in the next section.

5.5.2 Differing centre points

In the linear kernel case, we also performed another preliminary test to ascertain the differences between the actual centres achieved with our method and imputation methods. In the first test, we left the exact-centre method to run and find its own optimal centre point (we label this method 1); in the second test, we simply used the instance-specific feature averages over the present data in each dimension and calculated the minimal radius with respect to this centre point with our geometrical margin method (labelled method 2); and finally, we compared both of these results to the use of a classical SVDD method with mean-imputation, and calculated the relevant sphere size with our method (3a) and with respect to the mean-imputed dataset (3b). The dataset on which is operates is of size 300×6 and is a Gaussian cluster centred around a single point with equal dimensional variance. These results are summarised in Table 5.3.

Table 5.3: Differing centre points

Meth.	$\mathbf{a}$	R^2
1	$(-0.1923, -0.1189, -0.1312, -0.0401, 0.2173, 0.0524)$	13.0476
2	$(0.0277, 0.0576, -0.0931, 0.0831, 0.0751, -0.0260)$	14.6442
3a	$(-0.0499, -0.0013, -0.0568, 0.1548, 0.0945, -0.0644)$	14.2785
3b	n/a	14.2899

5.5.3 Random data, variable margin softness

We also conducted preliminary tests with a random 100×15 cluster data, distributed similarly to the cluster as given in the previous sub-section, after 40% of the datapoints were removed through an MCAR procedure. Figure 5.3 shows results derived for the two kernel functions above; Figure 5.4 applies the same tests to square-root based and logarithm-based kernels. The variable in each of the graphs as shown depicts the softness of the margin required. Note that the radii as shown in the graphs in Figures 5.3 and 5.4 are *not* referential of full, optimisation-based analyses with the exact-centre method, but rather refer to the querying of the maximal radius from *random centre points*, given the softness of the margin, in comparison to the margins between the centre point and the imputed data. This test was run purely to show the extent to which the margins could differ from imputed to geometric instance-specific, in the cases where different kernel functions were used. Note at this point that we have used two 'extra' kernels whose Mercer validity is questionable: the square-root based kernel having the following form:

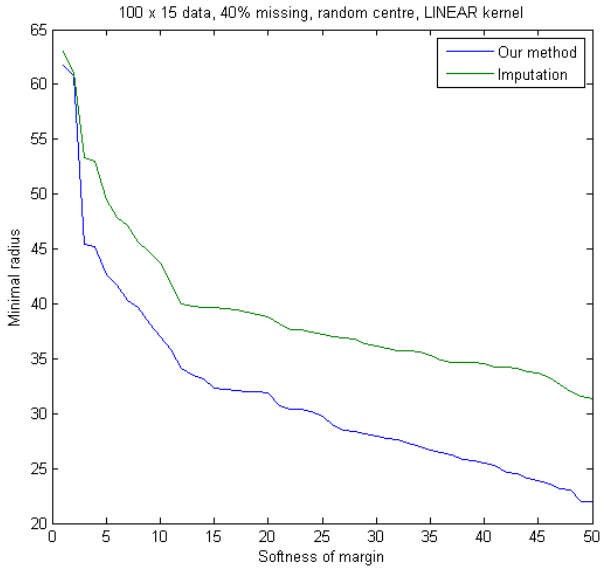
$$k(x, y) = \sqrt{|1 + x \cdot y|} \tag{5.40}$$

and the logarithmic kernel being:

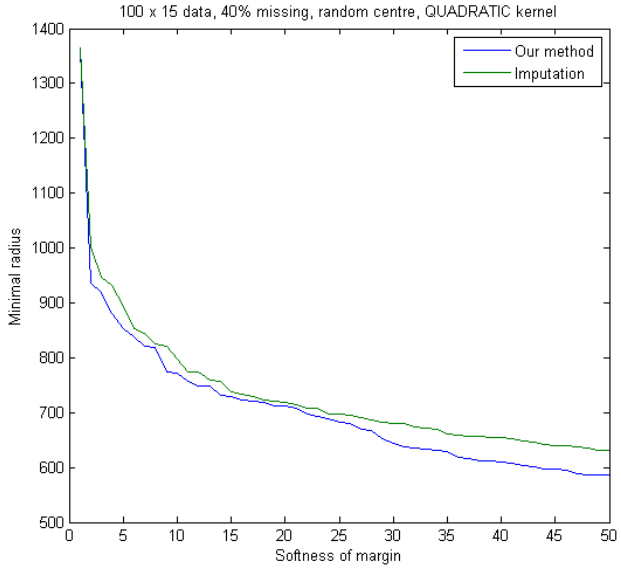
$$k(x, y) = \log |1 + x \cdot y|. \tag{5.41}$$

5.5.4 Discussion of preliminary tests

We note at this point that the MCAR nature of the missingness means that we do not expect our developed methods, in general, to work any better than imputation methods, since there is nothing 'special' about the way in which data has been

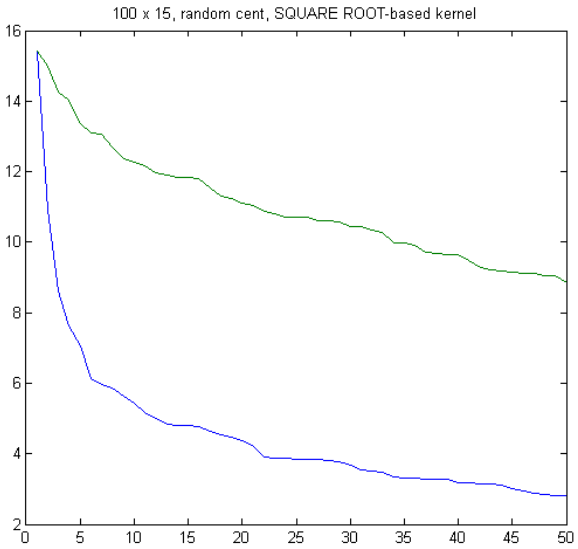


(a) Linear kernel

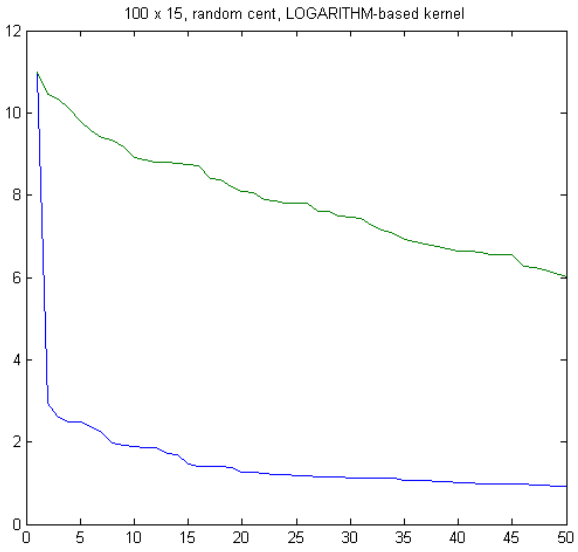


(b) Quadratic kernel

Figure 5.3: Exact centre, random data with variable margin softness



(a) Square-root based



(b) Logarithm-based

Figure 5.4: Square-root and logarithmic kernel results

rendered absent. In §5.5.2, and the first portion of §5.5.1, we have dealt with this issue by considering the linear kernel: that is, working in input space, thus guaranteeing a pre-image of the derived centre-point which is a valid member of the same vector space as the datapoints themselves. Thus, here we may reasonably expect it to perform *at least as well*, but perhaps not by much. Note at this point that the use of different kernels *does not* guarantee as good an image in input space as that in kernel space, and this is one of the major drawbacks of the exact-centre method, as described in the previous chapter.

Observations As shown in Table 5.1, in the case where a linear kernel was used, we have shown that the exact-centre method finds centre-points and radii which are smaller than those given when imputation methods were used, for all classes and all missingness ratios. This is a good result, as it shows that a compact description of the dataset can be made in all cases whilst still utilising all the present data. As a result of the linear nature of the kernel space, this result is also quite expected. The instance-specific margins are designed to deliver smaller inter-particle distances than the use of *any* imputation method in the case of missing data, and these results confirm that the optimisation procedures which led to these results were sufficiently convergent to produce these improved results. In the quadratic case (Table 5.2), however, the inability to project an optimal SVDD solution to a pre-image in input space is shown with the non-optimal radii achieved in some cases by our method. Nevertheless, for higher degrees of missingness it is shown that a smaller radius can still be calculated: this result is also expected, as higher degrees of missingness would require a higher proportion of the data to be (perhaps badly) imputed using classical methods. We provide further insight, and

corroborating results, in Table 5.3; here, we show that even in the linear case the computed centre points may be wildly different when our optimisation procedure is used, as a result of the different weightings which will result from data with missing features. The radii, in this case, show a clear preference for the (slower) method which allows the exact-centre algorithm to compute its own specific centre point; it shows that merely taking the feature averages cannot produce an acceptably tight result, even if the instance-specific margins are used on this point; and finally, they show that when applied to a classical SVDD, the radius reported in this case is better than the mere use of feature averages, but can still be improved upon by the subsequent calculation of the radius with respect to this point using our method. Thus, the deployment of the exact-centre method may very well be heuristically chosen with application: an imputation and SVDD could be used to *find* the centre point, with the subsequent particle-specific radii being computed by *one* run-through of our method; or *multiple* runs could be completed should the dataset be sufficiently simple that optimisation for a centre point is reasonable in terms of computational time. Finally, Figures 5.3 and 5.4 shows that when the softness of the margin is allowed to vary, and different kernel functions are used, the use of higher-degree polynomials results in a smaller improvement over imputation methods; the use of the log- and the square-root-based 'pseudo-kernels' shows here the converse. That is, if we were to do the opposite and 'reduce' the polynomial degree, a better fit can be achieved. These graphs also show that the improvement *given a known centre point* is made higher by a softer margin.

5.6 Results on synthetic data

In this section, we will describe a more regimented approach to comparing our method with the three different imputation schemes we mentioned above. In addition, we test under combinations of further aspects of the experiment that we can vary, which we outline here.

5.6.1 Other experimental variables

Kernels Throughout these results, we consider only kernels based on the inner product between datapoints, of the general form $k(x, y) = f(x \cdot y)$, since this is the framework through which the above methods were both derived. In particular, we will, in all cases, consider the performance of our methods with respect to two mapping functions: firstly, they will be considered in the 'usual' input space, by applying a linear kernel $f(s) = s$; secondly, their performances with respect to a feature space induced by a *quadratic* kernel $f(s) = (1 + s)^2$ will be compared. It is worth noting at this point that the purpose of this thesis is not to explore exhaustively the performances these methods would produce given multiple different kernels, as we set out primarily to show that such a method is reasonable and can produce meaningful results in the case of the inner product based kernel.

Soft margins It is also important that a method be well-suited to data classification in the presence of soft-margins, which our exact centre method, as described, is capable of producing. It performs this by allowing a parameter which describes how many *datapoints to omit* in the input data; that is, once the deviations of the parameters are sorted by size, if there are O_k omissions in each class k, the re-

ported radius of the dataset will simply be that of the $(O_k + 1)^{th}$ datapoint instead of the first (i.e. the maximal margin). This contrasts starkly, however, with the behaviour of the payoff parameter C as used in the SVDD case: the relation between C and the proportion of the data excused from membership of the circle is not clear. Therefore, in comparing our devised methods with those derived from pre-imputation followed by an SVDD analysis, we have performed the SVDD test multiple times, until the parameter C is discovered that has the effect of rendering *half the data* outside the sphere. Accordingly, a number to omit of $O_k = N_k/2$ is passed to the exact centre method for comparison. Thus, in the comparison table below:

- 'Hard' denotes where a hard margin, that is $C = 1$ or the omission rate $O_k = 0$, is used.
- 'Soft' denotes the case where, in the imputation-SVDD case, C is set such that the number of the coefficients α_i for each class such that $\alpha_i = C$ is equal to $N_k/2$; or, in the exact centre case, the omission rate O_k is set at $O_k = N_k/2$.

We thus arrive at the following structure for comparing our methods with these 'standard' approaches:

Table 5.4: Comparison structure

	Linear kernel	Quadratic kernel
Hard, Imp 0	Method **11**	Method **21**
Hard, Imp μ	Method **12**	Method **22**
Hard, Imp NN	Method **13**	Method **23**
Hard, XC	Method **14**	Method **24**
Hard, DO*	Method **15**	Method **25**
Soft, Imp 0	Method **16**	Method **26**
Soft, Imp μ	Method **17**	Method **27**
Soft, Imp NN	Method **18**	Method **28**
Soft, XC	Method **19**	Method **29**

* We include our dual-optimisation (DO) method here for completeness.

Thus, for each dataset we consider, the following will be performed to result in the complete suite of *results for comparison* as described above:

1. Consider a dataset $X \in \mathbb{R}^{N\times d}$, composed of a set of datapoints $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$, and its class vector $\mathbf{c} \in \mathbb{R}^N$, with $c_i \in \{1 \dots K\}$. Ensure that the data is aligned, centred, etc. properly according to the application required.

2. Break the dataset up into K constituent parts $\{X^{(k)}\}_{k=1}^K$, where the contents of each $\{X^{(k)}\}$ denote those datapoints x_i such that $c_i = k$. Let the size of these classes be denoted N_k for each k.

3. Perform three separate imputations of each class of the data corresponding to those detailed above for Imp 0, Imp μ and Imp NN, denoting these $X^{(k)1}$,

$X^{(k)2}$, $X^{(k)3}$ respectively, thus resulting in a suite of $3K$ separate datasets.

4. Perform a hard-margin SVDD analysis with a linear kernel on each of these $3K$ imputed datasets, thus fulfilling Methods 11, 12 and 13 as above. Retain the optimal coefficient vector α and the corresponding radius R for each set.

5. For each dataset, perform a series of soft-margin SVDD analyses, iterating the value of the payoff parameter C until it can be shown that $N_k/2$ (rounded down if necessary) of the coefficients α_i are such that $\alpha_i = C$; i.e. half the data is outside the margin. This fulfils Methods 16, 17 and 18.

6. Repeat the previous two steps with respect to the quadratic kernel $(1+x\cdot y)^2$; this fulfils Methods 21, 22, 23, 26, 27 and 28.

We now have the information required to apply both the method we have described in this chapter and the dual-optimisation system we will cover in Chapter 6 to the results of imputation of data followed by a process of SVDD analysis. We complete this chapter's analysis by applying our exact-centre method to three datasets we have devised to exhibit structural missingness, the construction of which we now describe.

5.6.2 Synthetic datasets

Table 5.5 denotes radii achieved with a 100×2 dataset, normally distributed about the origin with unit variance in each dimension, with a filter applied to render all values x_{ij} with $|x_{ij}| > 1$ as missing.

Table 5.6 denotes radii achieved with a 50×4 dataset composed of two clusters. The first cluster resides in dimensions (f_3, f_4) alone, and consists of a nor-

mally distributed cluster of 25 points with mean $(1, 1)$ and unit variance. The second cluster resides in (f_1, f_2) and is a similarly distributed cluster about $(-1, -1)$, again with unit variance.

Table 5.7 denotes radii achieved with a 90×3 dataset designed so that all datapoints have a common feature f_1, but some (25) have the feature f_2 present, some (30) the feature f_3, and some (35) the feature f_4. Each of the 90 datapoints thus has exactly two features, either of pattern f_{12}, f_{13} or f_{14}. Features are again distributed normally with means of $(1, 2, 3, 4)$ as per each feature, with unit variance.

Table 5.5: XCDD results: 2-D cluster with value filter

	Linear kernel	Quadratic kernel
Hard, Imp 0	1.25289	1.88347
Hard, Imp μ	1.25289	1.88347
Hard, Imp NN	1.25289	1.88885
Hard, XC	1.25289	2.36436
Soft, Imp 0	0.64761	1.00597
Soft, Imp μ	0.62838	0.96624
Soft, Imp NN	0.74140	1.16572
Soft, XC	0.42162	0.78634

5.6.3 Observations

In the study on the 2-D cluster with a value filter (Table 5.5), it is shown in the hard-margin case that no improvements are made in either the linear or quadratic case with our exact-centre method; however, it does show that in this case, our

Table 5.6: XCDD results: Two disjoint 2-D clusters

	Linear kernel	Quadratic kernel
Hard, Imp 0	2.67234	11.11552
Hard, Imp μ	2.23574	11.07670
Hard, Imp NN	2.23574	11.07670
Hard, XC	2.23057	10.57009
Soft, Imp 0	1.48615	3.87435
Soft, Imp μ	1.04090	3.96493
Soft, Imp NN	1.04053	3.96490
Soft, XC	0.98055	3.08169

Table 5.7: XCDD results: One common dimension

	Linear kernel	Quadratic kernel
Hard, Imp 0	4.29481	24.42394
Hard, Imp μ	2.32852	21.03813
Hard, Imp NN	2.52866	23.25400
Hard, XC	2.31592	20.26059
Soft, Imp 0	2.68217	11.57337
Soft, Imp μ	1.05515	9.23119
Soft, Imp NN	1.44247	12.50144
Soft, XC	0.99583	7.73160

method does not perform *worse*, which is the main observation here. The imputation methods, indeed, all produce the same radial result in this case. However, when a soft margin is allowed, we see that our method is effective in finding a smaller descriptive radius for this dataset in both linear and quadratic kernel cases. This result is satisfying and concurs with the observations made in the previous section that softer margins can produce a greater improvement in the use of our methods over imputation. Table 5.6, showing the results of the dataset with two disjoint clusters, tells a slightly different story: here, the radii show a small improvement with the use of our method even in the hard margin case. Furthermore, this is carried over to the quadratic kernel, which itself is a good result since, as we stated above, this forerunner method is not always expected to perform better with this mapping. As expected, the soft margins also show an improvement. These results give us important clues as to the precise nature of the types of dataset for which both this method and the one we will describe in Chapter 6 would find favour in application. Finally, we consult Table 5.7, referring to the dataset with one common dimension. With a hard margin, improvements here are, so far, to be found in all cases and with both kernels, with some tests performing impressively and producing significantly smaller radii. We shall continue the analysis of these synthetic datasets when we analyse our DO method in §6.7.

Chapter 6

Dual optimisation domain description

6.1 Algorithm

6.1.1 Motivation

Although the method described in §5.4 produces results which are favourably comparable to imputation methods in some cases (see §5.6), extending the system into kernel space does not guarantee better performance. This is because the main motivation for extending a classifier into kernel space, as previously mentioned in §2.3.2, is so more features can be considered other than those dimensions present in input space. Since our centre point $\mathbf{a}$ can only be allowed to vary over the original input space according to this method, a different approach is motivated: one that takes into account the full dimensionality of the mapped space ϕ.

Extending input space Recall that, in §2.4, we express the objective vector (the margin $\mathbf{w}$ in SVM, and the centre point $\mathbf{a}$ in SVDD, hereafter referred to as $\mathbf{a}$) as a weighted combination of the datapoints *after* mapping into kernel space:

$$\mathbf{a} = \sum_{i=1}^{N} \alpha_i \phi(x_i) \tag{6.1}$$

Thus, in cases where we do not wish to compute ϕ explicitly, the exact representation of $\mathbf{a}$ is intractable, and we use the N coefficients α_i to provide a projection into an N-dimensional subspace (where hopefully $N \gg d$). We appeal to properties of *linear dependency* using the following definition:

> **Definition** *(Linear dependence)* A set of vectors, $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^d$, are said to be *linearly dependent* if there exists a set of coefficients $\lambda_1 \ldots \lambda_n$, with not all $\lambda_i = 0$, such that:
>
> $$\sum_{i=1}^{n} \lambda_i \mathbf{a}_i = \mathbf{0}, \tag{6.2}$$
>
> where $\mathbf{0} \in \mathbb{R}^d$ is the zero vector. If it is not the case that Equation 6.2 holds, then the set $\mathbf{a}_1, \ldots, \mathbf{a}_n$ is said to be *linearly independent*.

In kernel space, the following two factors influencing the choice of method can be identified:

- Linearly-dependent vectors in input space will not necessarily be linearly dependent in kernel space;
- The preimage of $\mathbf{a} \in \mathbb{R}^D$ will not, in general, exist in input space.

To see both these points, we revisit the simple kernel mapping used in §2.3.3, $\phi(x_1, x_2) = (1, x_1, x_2, x_1 x_2)$. If we take $x_1 = (1, 2)$, $x_2 = (2, 4)$, we see that

although x_1 and x_2 are linearly dependent and define only a one-dimensional basis (since $x_2 - 2x_1 \equiv (0, 0)$, thus satisfying Equation 6.2) – their mappings $\phi(x_1) = (1, 1, 2, 2)$ and $\phi(x_2) = (1, 2, 4, 8)$ form a linearly independent set. Also, if we take $\alpha = (0.5, 0.5)$ we derive their centre of mass in feature space:

$$\mathbf{a} = 0.5 \begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \end{pmatrix} + 0.5 \begin{pmatrix} 1 \\ 2 \\ 4 \\ 8 \end{pmatrix} = \begin{pmatrix} 1 \\ 1.5 \\ 3 \\ 5 \end{pmatrix},$$

which is not expressible as a preimage in input space for any combination $\mathbf{a} = \phi(x_1, x_2)$. Thus, in the general case, there is a clear motivation to use the full kernel-induced feature space, as it gives further choices for selection of the optimal vector for these classification methods.

Further comments Note also that, if specification of a vector α is the main method of determining a centre point according to Equation 6.1, this equation requires that to avoid complete collapse of the definition of $\mathbf{a}$ yielding only trivial results, the mapped data basis $\{\phi(x_i)\}$ must all be of full features. Again, to see this with the simple kernel mentioned above, consider the following: $x_1 = (1, \text{NaN})$, $x_2 = (\text{NaN}, 4)$, $x_3 = (3, 6)$. Even though here x_3 is of full data, any nontrivial combination of these datapoints will yield $\mathbf{a} = (1, \text{NaN}, \text{NaN}, \text{NaN})$, a nonsensical vector giving zero dimensions of information. Thus an important fact arises:

> In order to perform meaningful completions of datapoints with respect to a centre point $\mathbf{a}$ defined as in Equation 6.1, we must have a

dataset which has been *imputed* in some fashion to avoid collapse of the subspace in which $\mathbf{a}$ resides.

Four corollaries of this soon follow:

- $\mathbf{a}$ must itself have full features.
- For every derivation involving a given dataset X, there must be another set, which we will call X^*, of readily-imputed data such that $\mathbf{a}$ can be derived according to Equation 6.1.
- As a result, the exact definition of X^* is almost arbitrary, as long as it is sufficiently large ($N^* \gg d$) to provide a good basis in the domain of ϕ. However, although it is tempting to conclude that taking $X^* = I_d$ recovers the original method in §5.4, this is not the case: we discuss this below.
- Unlike in the full data case, our choice of values which can be taken by α_i are no longer restricted, since they are no longer tied to our original dataset.

The above points show that knowing the kernel matrix between the data itself is almost useless; as shown in Figure 6.1, we cannot expect to be able to express, in general, a relation between the completions with respect to feature-space points and those with respect to linear combinations of those points. Thus, for every α denoting a definition of a feature-space $\mathbf{a}$ we consider, we must again optimise a completion with respect to it, knowing that even if the interpoint completions were known, it would not be of much use. With this in mind, we now derive a general method for full access to the feature space defined by the span of the images of the datapoints in X^*.

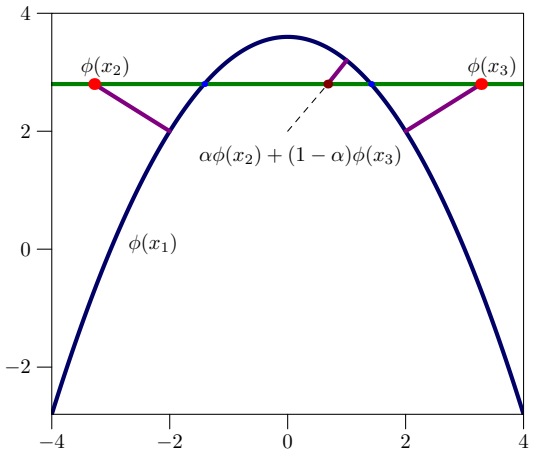


Figure 6.1: Kernel completions in feature space

6.1.2 Derivation

With these points in mind, we derive from first principles a method to calculate the FDC "centre point" of a dataset in kernel space and show that optimal completions of missing data with respect to this point are possible. Let $X = \{x_1, \cdots, x_N\}$ be our dataset, assumed to have some features missing, and let $X^* = \{x_1^*, \cdots, x_M^*\}$ be our FDC dataset (not necessarily with $M = N$, as explained above). Let our centrepoint be described as follows:

$$\mathbf{a} = \sum_{i=1}^{N} \alpha_i \phi(x_i^*) \tag{6.3}$$

To compute distances from points in X to this centrepoint for a given set of α_i, we must calculate the following set of distances:

$$\mathbf{d} = \{\delta_j\}_{j=1}^{N} = \left\| \phi(x_j^a) - \sum_{i=1}^{M} \alpha_i \phi(x_i^*) \right\|^2 \tag{6.4}$$

The first question is how we complete the x_j optimally (in input space) with respect to the kernel that implicitly defines ϕ *and* the specific problem conditions defined by the vector α and the imputed dataset X^*. Expanding out Equation 6.4 gives:

$$\delta_j = k(x_j, x_j) - 2\sum_{i=1}^{M} \alpha_i k(x_i^*, x_j) + \sum_{(i_1,i_2)=(1,1)}^{(M,M)} \alpha_{i_1}\alpha_{i_2} k(x_{i_1}^*, x_{i_2}^*) \tag{6.5}$$

where here we have dropped the a superscripts for clarity. We henceforth operate in the case where $k(x, y) = f(x \cdot y)$ for some function f; this equation thus becomes:

$$\delta_j = f(x_j \cdot x_j) - 2\sum_{i=1}^{M} \alpha_i f(x_i^* \cdot x_j) + \sum_{(i_1,i_2)=(1,1)}^{(M,M)} \alpha_{i_1}\alpha_{i_2} f(x_{i_1}^* \cdot x_{i_2}^*) \tag{6.6}$$

In input space, x_j will have d features, either present or missing. For a feature l that is missing we can calculate the derivative of δ_j with respect to its value as follows:

$$\frac{1}{2}\frac{\partial \delta_j}{\partial x_{jl}} = x_{jl} f'(x_j \cdot x_j) - \sum_{i=1}^{M} \alpha_i x_{il}^* f'(x_i^* \cdot x_j) \tag{6.7}$$

Using this gradient, we can compute (via an optimisation procedure) the *optimal completion* of each datapoint x_j in turn, with respect to a given, known vector α describing the kernel-space centre point we wish to consider. Furthermore, the following quantity can be just as readily computed, showing the reliance of each of the δ_j on the vector α itself:

$$\frac{1}{2}\frac{\partial \delta_j}{\partial \alpha_m} = -f(x_m^* \cdot x_j) + \sum_{i=1}^{M} \alpha_i f(x_i^* \cdot x_m^*) \tag{6.8}$$

This quantity should be used with caution: in order to compute the vector δ of distances $\{\delta_1 \dots \delta_N\}$ from $\mathbf{a} = \sum_{i=1}^{M} \alpha_i \phi(x_i^*)$, we need to first optimise the completion of each x_j with respect to this quantity. Therefore, a gradient-descent must

be applied *for every point* based on the quantity given in Equation 6.7. Once this optimisation loop has been completed, only then is it valid to consider altering the vector α via Equation 6.8, since 'optimising' α with respect to each x_j on-the-fly would otherwise be meaningless.

Sanity check We mentioned the case above that it is possible to take $X^* = I_d$ to recover the exact centre method mentioned in Chapter 5. We note that in this case, Equation 6.7 cancels to:

$$\frac{1}{2}\frac{\partial \delta_j}{\partial x_{jl}} = x_{jl} f'(x_j \cdot x_j) - \alpha_l f'(x_{jl}), \tag{6.9}$$

which cancels down in the linear case to the familiar result of having a minimum at $x_{jl} \equiv \alpha_l$.

Coefficient optimisation In order to optimise a multi-variable objective function against a multi-variable argument, it is often useful to compute the matrix of Jacobians $J = \{J_{mj}\}$, where the J_{mj} are defined as in Equation 6.8, for passing to an optimisation solver. The following is a simple way of achieving this: Let X^* be the full-data $(M \times d)$ computational dataset as described above, and let $X^{(a)}$ be the original $(N \times d)$ dataset with missing data completed optimally with respect to a given weighting vector α. Furthermore, let $\mathbf{e}_N$ be the vector of all ones, of length N. Then the formula for the $(M \times N)$ transformation matrix is given by:

$$J = -2\left[f(X^*(X^{(a)})^T) - (f(X^*(X^*)^T)\alpha)\mathbf{e}_N^T\right]. \tag{6.10}$$

Thus J_{mj}, for $m \in \{1 \dots M\}$ and $j \in \{1 \dots N\}$, can be seen as the following: "If $J_{mj} < 0$, deviation δ_j would benefit – i.e. grow smaller – if the coefficient α_m were increased". How this process then proceeds is largely heuristic, and depends

on the structure of the problem. If a hard margin classifier is required, it may be seen as appropriate to concentrate on the datapoint with largest deviation; a softer margin would require changing α values for x_j with smaller deviations, ignoring the higher ones as being outside the sphere. Either way, the values of J can be used to draw a potentially better α in an iteration procedure.

Minimax optimisation Suppose we wished to enclose a sphere with a hard margin. In MATLAB's Optimization Toolbox, the function `fminimax` can be used to perform this computation automatically. Given the transformation matrix J as described above, the following function will be optimised:

$$\alpha = \arg\min_{\alpha} f(\alpha), \text{ where } f(\alpha) = \max_{i} \delta_i^a. \tag{6.11}$$

The extension of this formula to soft margins is not obvious; the `fminimax` function does rely heavily on the continuity and convexity of the case where only the maximal deviation need be minimised. Softer margins could not be dealt with in this way as a result of the algorithm used for the computation of the 'minimax' solution; the gradient discontinuity proves a problem where the identity of the datapoint with maximal deviation 'switches'. However, it is still heuristically possible to inform our choice of α manually. We summarise this process in Algorithm 3. As with Algorithm 2, we require a run of the algorithm for each candidate vector α; once the results have been found for each iteration, we may use Equation 6.10, along with the deviations δ_j, to inform another unconstrained optimiser as to the best way to alter the vector α according to some set rule (e.g. for hard margins, the α_i with the greatest deviation might be adjusted). Thus we derive a new candidate α and repeat the algorithm until the result is satisfactory.

Algorithm 3 Dual-optimisation method

Require: $X^* \in \mathbb{R}^{M \times d}$ as auxiliary data

Require: candidate weighting vector $\alpha \in \mathbb{R}^M$

Initialise $\delta^a \in \mathbb{R}^N$ and cell array C of size N, where each C_i is at largest a vector of size d

Let $\mathbf{h} \in \mathbb{R}^N = \{h_j\} \leftarrow$ (number of present features in x_j)

for $j \in \{1 \dots N\}$ **do**

if $h_j = 0$ **then**

$d_j^a \leftarrow$(d_j as per equation 6.6)

else

Initialise $\mathbf{c} \in \mathbb{R}^{h_i}$

repeat

$x_j^c \leftarrow \{x_j^{obs}; x_j^{mis} \leftarrow \mathbf{c}\}$

$d(\mathbf{c}; x_j^{obs}) \leftarrow f(x_j^c \cdot x_j^c) - 2\sum_{i=1}^{M} \alpha_i f(x_i^* \cdot x_j^c)$

$+ \sum_{(i_1,i_2)=(1,1)}^{(M,M)} \alpha_{i_1}\alpha_{i_2} f(x_{i_1}^* \cdot x_{i_2}^*)$

until optimiser convergence in $\mathbf{c}$ to $\hat{\mathbf{c}}$

end if

$C_j \leftarrow d(\hat{\mathbf{c}})$

end for

With respect to the given candidate α, the dataset's optimised maximal distance is now $\max_j(\delta_j)$.

Reconstruct X using the elements of C to form a FDC dataset $X^{(\alpha)}$ optimally completed with respect to the candidate α

6.2 Discussion

Rôle of auxiliary data By virtue of no longer being directly tied to the dataset, it is worth examinining the significance of the auxiliary data matrix X^* in more detail. It is required through the problem of not being able to adequately map datapoints with missing data to a centre point $\mathbf{a}$ via dual coefficients α as described above. Although it may be hoped, through the analogues between this method and the SVDD system of providing coefficients α_i in the Wolfe dual problem, that this method always provides a superior range over feature space than the exact centre method described in §5.4, this is not necessarily the case. We demonstrate this via a method similar to proof by contradiction, by considering the question we alluded to in §6.1.1 above of whether the exact centre method can be arrived at as a special case of the dual-optimisation method. That is, if we take $X^* = I_d$, and our exact-centre method produces an optimal solution $\mathbf{a} = \{a_i\}$ with respect to the subspace of feature space it is obliged to operate within, does there always exist some vector $\{\alpha_i\}$ such that:

$$\sum_{i=1}^{d} \alpha_i \phi(\mathbf{e}_i) = \phi\left(\sum_{i=1}^{d} a_i \mathbf{e}_i\right)? \tag{6.12}$$

Here, by similar convention to classical mechanics, we have the vectors $\mathbf{e}_i \in \mathbb{R}^d$ defined as the i^{th} row of I_d. If there were a solution to this, and we maintain our consideration of dot-product based kernel functions, it would imply:

$$\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^d, \exists \alpha \in \mathbb{R}^d$$
$$\text{such that } \sum_{i=1}^{d} \alpha_i f(b_i) \equiv f\left(\sum_{i=1}^{d} a_i b_i\right), \tag{6.13}$$

i.e. that the function f is separable in the variables b_i; this condition is not satisfied by the quadratic kernel, but is trivially satisfied in the linear case, where

the two situations are equivalent. However, in the general case it does raise the question of whether a 'near' point can be derived as a reasonable initial guess for an optimisation procedure: this will be covered later in §6.6.

Further comments We have described a system that, given a kernel based on the dot product, a dataset with missing data and some sort of iteration rule governing how to act when given a list of deviations δ_j, can provide a description of a dataset in kernel space whilst completely avoiding the need for imputation of extra data, all the while providing the data with their optimal completions with regards to the curves and surfaces their projections would describe in kernel-induced feature space. The main drawback of this method is that of computational complexity, but we will see in §6.7 that the results it can produce on simple enough toy examples show that the method is a potentially promising one. The dual optimisation step does add a large amount of computational time, but – unlike the previous, exact centre-based method – the results have the advantage of being able to search the whole of the feature space instead of the subspace defined by the span of the input space alone.

6.3 Method preparation

It is important to have a framework in place where our developed methods are given reasonable points at which to begin optimisation procedures: this applies to both the XC method described in Chapter 5 and the DO method above. In §6.2 we observed that it was not always possible to find, for every point $\mathbf{a}$ in input space, an equivalent representation α in dual space: a fact which is true even in the full data case. Thus, it is not always reasonable to expect the coefficient vectors

α as given by the normal imputation-SVDD methods described above to be the optimal solutions for our methods, which avoid imputation altogether. Indeed, we have seen that when using anything other than linear kernels, the preimage of $\mathbf{a}$ in input space may not exist; this causes problems with our exact-centre method, which requires an input-space solution to be found. In the linear case, the starting point is trivial, as we use the centre point implied directly by the coefficients α_i, and thus take:

$$\mathbf{a}_0 = \sum_{i=1}^{N} \alpha_i x_i^I,$$

where the points x_i^I describe one definition of the imputed dataset and the exact-centre optimisation process can be repeated for all three separate centres in this way to find the best result. In other cases, however, the definition of $\mathbf{a}$ itself will be intractable; as a general optimal starting point for our exact-centre method, we therefore consider the following point:

$$\mathbf{a}_0 = \arg\min_{\mathbf{a}} \left\{ \|\phi(\mathbf{a}) - \mathbf{a}_I\|^2 : \mathbf{a} \in \mathbb{R}^d \right\} \tag{6.14}$$

where $\mathbf{a}_I = \sum_{i=1}^{N} \alpha_i \phi(x_i^I)$, from the results of analysis from a given imputation method as stated above. It follows that:

$$\mathbf{a}_0 = \arg\min_{\mathbf{a}} \left[f(\mathbf{a} \cdot \mathbf{a}) - 2 \sum_{i=1}^{N} \alpha_i f(\mathbf{a} \cdot x_i^I) + \sum_{i,j}^{(n,n)} \alpha_i \alpha_j f(x_i^I \cdot x_j^I) \right]. \tag{6.15}$$

The third term in this sum does not depend on $\mathbf{a}$, and thus we have a nonlinear (but fairly 'simple') optimisation problem based on the first two terms here to determine the optimal starting point $\mathbf{a}_0$. This simple process can, again, be repeated with the three definitions of the imputed dataset X^I, with the exact-centre optimisation process started from each point in turn.

6.4 Joint optimisation approaches

In this section, we detail another approach to the problem of an optimal imputation with respect to a kernel function, and show two methods in which it may be applied. We shall relax the requirement of Equation 5.35: that is, defining the distance from an incomplete datum from the centre of an enclosing sphere as being the instance-specific mean over all possible imputations of *that point alone*, and instead we see the entire dataset as a whole and perform a *joint optimisation* with respect to the missing data over the entire set and the weighting vector α. Thus we seek:

$$(\hat{\alpha}, \hat{X}_{mis}) = \arg \min_{\alpha, X_{mis}} \left(\max_i \|\phi(x_i) - \sum_{j=1}^{N} \alpha_j \phi(x_j)\|^2 \right). \tag{6.16}$$

There are both advantages and drawbacks of this formulation. One major advantage is immediately obvious, in that we have eliminated the problems inherent in §6.1 regarding the requirement of the provision of an auxiliary dataset and the decoupling of the connection between the parameter α and the datapoints. This is a very significant improvement, since it means that not only can computational time be greatly reduced by the lack of need to perform several tests on an heuristic basis and choose the best result, but it also brings rigour back into the system as the α_i can, once again, be seen as relating directly to the x_i. Thus the Lagrangian constraints, derived in §2.4.2 for SVDD in the case of full data, can be re-instated:

$$\sum_{i=1}^{N} \alpha_i = 1, \mathbf{a} = \sum_{i=1}^{N} \alpha_i \phi(x_i), \alpha_i \in [0, C]. \tag{6.17}$$

Furthermore, this system has an advantage in that the result as produced, should the optimisation process function correctly, will *explicitly* give both the best imputation $\hat{X}_{mis}$ and the required Lagrangian coefficients $\hat{\alpha}$ for direct entry into the

quadratic expression for the minimal radius as required. This was also not the case above, since the spaces in which the data and the α coefficients resided were different, albeit in general isomorphic to one another (should a sufficient number of different points be used in X^* to permit basis coverage, and under regularity assumptions about the kernel function).

It is also clear from this new formulation, however, that this is computationally a far more difficult problem than simply taking each datapoint in turn and minimising its missing data component with respect to a given set of α_i and a kernel function, as in §6.1. We now have the entire $(N + \Sigma_{mis})$ dimensional space through which to search for a minimal imputation, and although the relations given in Equation 6.17 will help immensely with the search space for α, it still remains a daunting task, especially if we are dealing with datasets with a high proportion of data absence. We detail below the results of naïve optimisation, and also introduce the method of *particle swarm optimisation* which assists in this type of problem.

6.4.1 Naïve constrained optimisation

For some datasets, particularly those that are smaller or simpler, proceeding directly to a constrained optimisation procedure, using one of the functions written for e.g. the MATLAB Optimization Toolbox, may be a realistic option. Let d_i be the distance from the centre point of point i; to compute the gradient for applica-

tion of a gradient-descent method to find the minimal value, we note that:

$$\frac{\partial d_i}{\partial x_{ml}} = \begin{cases} 2(1-\alpha_i)\left[x_{il}f'(x_i \cdot x_i) - \sum_j \alpha_j x_{jl} f'(x_i \cdot x_j)\right] & m = i; \\ -2\alpha_m \left[x_{il}f'(x_i \cdot x_m) - \sum_j \alpha_j x_{jl} f'(x_m \cdot x_j)\right] & m \neq i. \end{cases} \tag{6.18}$$

$$= 2(\alpha_m - \delta_{im}) \left[\sum_{j=1}^{N} (\alpha_j - \delta_{ij}) x_{jl} \dot{K}_{jm}\right], \tag{6.19}$$

where δ_{ij} is the usual Kronecker delta index, and $\dot{K}$ is a kernel matrix composed of the values $\dot{K}_{ij} = f'(x_i \cdot x_j)$. To construct the necessary family of matrices $\{D^{(l)}\}_{im} \equiv \frac{\partial d_i}{\partial x_{ml}}$ for use as derivative information in an optimisation procedure, we note that if we create a matrix $B_{ij}(\alpha) = \alpha_j - \delta_{ij}$ and a matrix $Q^{(l)}_{ij} = x_{il}\dot{K}_{ij}$, we have:

$$D^{(l)}_{im} = 2B_{im}\left(\sum_j B_{ij} Q^{(l)}_{jm}\right) = 2B_{im}\left\{BQ^{(l)}\right\}_{im} \tag{6.20}$$

$$\Longrightarrow D^{(l)} = 2B \otimes (BQ^{(l)}), \tag{6.21}$$

where $\otimes$ is the matrix Hadamard product $\{A \otimes B\}_{ij} \equiv A_{ij}B_{ij}$. Alternatively, another approach is to use an automatic differentiation package such as TOMLAB to compute derivatives with respect to datapoints 'on-the-fly'. In any case, it is thus straightforward to pick out the relevant indices and submit a vector containing the current iteration's values of both the imputation X_{mis} and the vector α, and apply a standard optimisation procedure such as `fminimax` to solve Equation 6.16.

6.4.2 Particle swarm optimisation

Algorithm 4 describes a well-known method of optimisation known as *particle swarm optimisation* (PSO). Proposed originally in 1995 by Kennedy and Eberhart [39], it is often used where gradient information for an objective function

that needs to be minimised is either difficult to obtain or is known to be highly irregular or discontinuous. It precludes the need to compute and descend along gradients at the expense of several extra parameters that need to be tuned; although these can themselves be optimised, as shown in e.g. Pedersen [61]. The method is based around a random search of feature space via a 'swarm' of particles driven by stochastic choices of extra velocity parameters appended to each feature. When a member of the swarm finds a new (perhaps local) minimum of the objective function, it may then pass the positional information of this point to other swarm members, who then have their new velocity proposals attracted towards this point whilst still randomly searching the local structure. At no point is gradient information either used or necessary for the operation: the optimisation is done purely on a 'high-score' basis. As information is passed between swarm members, it holds that the greater the swarm, the more effective the algorithm, but clearly also the greater the computational time.

The end effects of PSO, in the ideal case, are that most swarm members converge on the minimal point of the objective function; a stopping criterion can thus be adopted that requires a proportion of the members to be within a certain spatial tolerance of the current minimum, or the method can just be run for a certain number of iterations. It is not used as a sampling method as it does not use probabilistic acceptance criteria such as the Metropolis-Hastings ratio; rather, when properly tuned it is theoretically a good way to address the problem of local minima or a high-dimensional feature space with sometimes discontinuous gradients. The tunable hyper-parameters are shown in the pseudocode for Algorithm 4, and we will not pay them too much further attention except to say that in the results gleaned from running PSO tests on the applied data, good results were achieved

Algorithm 4 Particle swarm optimisation

Require: function $f : \mathbb{R}^d \to \mathbb{R}$ to minimise

Initialise x_0, iterations N, swarm size M, inertia I, correction χ, stepsize σ

Let $x_i \leftarrow x_0$, $v_i \leftarrow 0$, $y_i \leftarrow \infty \; \forall i = 1 \dots M$

for $k = 1 \dots N$ **do**

  for $i = 1 \dots M$ **do**

    $x_i \leftarrow x_i + \sigma v_i$

    if $f(x_i) < y_i$ **then**

      $x_i^* \leftarrow x_i$

      $y_i \leftarrow f(x_i)$

    end if

  end for

  $\hat{\imath} \leftarrow \arg\min_i y_i$

  for $i = 1 \dots M$ **do**

    $\theta_j^{(p)} \sim U[0,1]$ for $p = 1, 2, 3$; $j = 1 \dots d$

    $v_i \leftarrow I\theta^{(1)} v_i + \chi \left[\theta^{(2)}(x_i^* - x_i) + \theta^{(3)}(x_{\hat{\imath}}^* - x_i)\right]$

  end for

end for

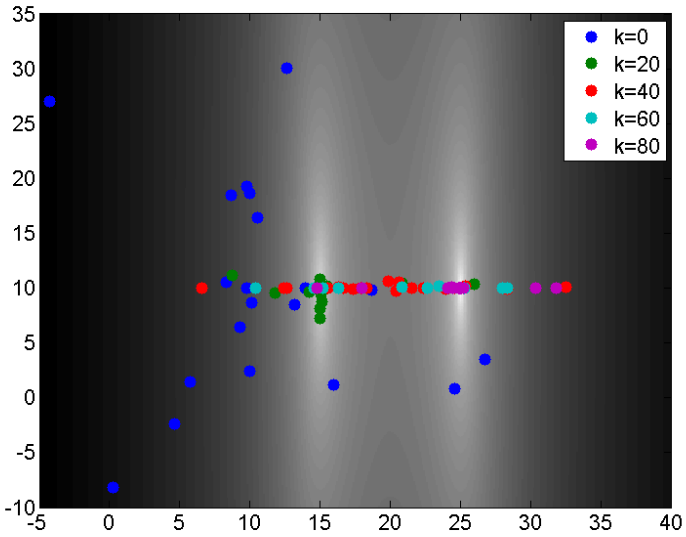


Figure 6.2: PSO example on $f(\mathbf{x}) = ((x_1 - 20)^2 - 25)^2 - 0.5x_1 + (x_2 - 10)^2$.

by fixing these parameters as the following:

- x_0: Gaussian cluster about origin with standard deviation 5. Note that this standard deviation does matter in terms of the efficiency of the algorithm. Too small and the algorithm converges too quickly on an unrealistic point; too large and the risk of non-convergence is encountered.
- Swarm size $M = 20$
- Iterations N: shown in the applied results chapter but stopped after a convergence criterion of 90% of the swarm (18 members) having an objective value of within 0.001 of the member with the minimal objective
- Step size $\sigma = (1.3)^{-1}$, taken from original code by Elshamy[1]. This parameter also contributes significantly towards the efficiency and probability of convergence.
- Inertia $I = 1$
- Correction factor $\chi = 2$

Finally, we note here that clearly the efficiency of any PSO algorithm will be improved by hyper-optimisation for these parameters themselves, and thus is a branch of further work in the study we present here.

Toy example Figure 6.2 shows a typical run of a PSO procedure. The objective function is quartic in the horizontal (first) dimension, and has two significant minima: near $(15, 10)$ and near $(25, 10)$. The latter is the global minimum, and

[1]PSO code is available online at 'MATLAB Central' by Wesam Elshamy (`wesamelshamy@yahoo.com`).

between the points there is an area where the objective function is much higher. The blue dots show the randomised starting points; after 20 iterations, we see the green dots have begun to cluster about the local, non-global minimal point on the left of the figure, as no better point has yet been reported in feature space. We also see that there is still some variance in the iterations for the simpler, quadratic feature x_2. However, by 40 iterations, this is gone: the method has converged on $x_2 = 10$ as a minimum in the second feature; more significantly, however, a better point has been found in the x_1 direction, and the swarm is now attracted to this improved point nearer $(25, 10)$. By 60 iterations, the teal dots show more of the swarm members clustering, and by 80 iterations, the purple dots show that most of the swarm is now near to this point with far less spatial variance.

Application to GMSVDD In §2.4.2, we showed that in the full-data case, a support vector domain description system can be built to derive an optimal α for a given dataset, and thus provide some form of explicit depiction of the position of the most optimal centre point of a dataset. Furthermore, we noted that this is in the form of a *quadratic programming* procedure, thus in general allowing efficient optimisation methods to be employed in its solution due to the guaranteed positive semi-definiteness of the kernel matrix K. This is certainly always the case with a full dataset; however, in the case of missing data, we may consider all possible imputations and conclude that the procedure that describes a full-data SVDD can be viewed as the function:

$$\hat{\alpha}(X_{mis}) \leftarrow Q(X_{mis}|X_{obs}) \quad : \quad \mathbb{R}^{N\times d} \rightarrow \mathbb{R}^N \tag{6.22}$$

$$\simeq \quad \mathbb{R}^{\sum_i h_i} \rightarrow \mathbb{R}^N, \tag{6.23}$$

where $Q(X_{mis})$, conditioned on the known portion of the dataset X_{obs}, provides an *optimal* QP solution $\hat{\alpha}$ for *any* completion X_{mis} we provide, by filling in the dataset X as required and running a quadratic program to derive the *problem-specific* optimal α. This can be viewed as a function as with any other; in particular, however, we note that due to the structure of the problem, a small difference in the imputation may lead to a large jump in the optimal α, so this form is not given well to gradient-descent methods if a search for a minimal radius over the feature space X_{mis} is required. Figure 6.3 shows how $\hat{\alpha}$ changes as we pick different random imputations for a set of incomplete data. The predominantly light-coloured area in this figure denotes those data for which $\alpha = 0$. This clearly illustrates the phenomenon, alluded to in §2.4.4, that in most support-vector tests with dot-product kernels, most of the training points x_i end up with coefficients of $\alpha_i = 0$ for sufficiently large C, denoting their membership of the region inside the sphere. Only a handful of coefficients end up nonzero, to denote their presence either on or (if $C < 1$) outside the sphere. Crucially, the output $\hat{\alpha}$ may be turned into a unique[2] minimal radius R using relation 2.77 from §2.4.4, giving us a function with a single output argument R to minimise. We can motivate the use of a non-gradient method for this problem via careful consideration of the effect of the missing data parameter space on R. Given a current realisation of the missing features in a dataset, and thus a sphere with a centre given by α and known radius R, we consider the gradient of R with respect to the missing features in all incomplete data with $\alpha = 0$. The gradient of α, and hence R, will only have a nonzero derivative should the alteration *change the support vectors*: this itself involves an alteration in the missing features of an incomplete datum sufficiently large to take it near

[2]Unique, as ideally all support radii are equal; in general, take the minimum over all SVs.

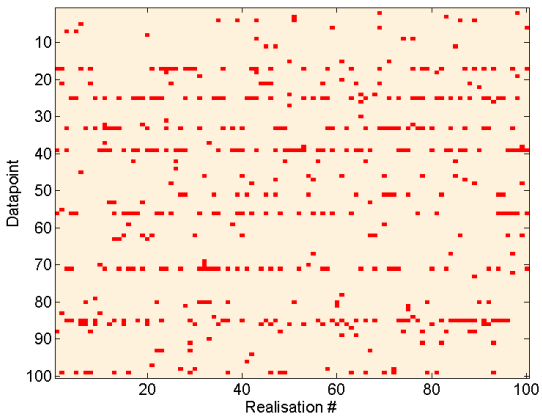


Figure 6.3: 100 realisations of support vectors on a 100×5 dataset

the enclosing sphere. When it does approach the sphere, however, it may well be that another optimal solution can be formed and the identities of the support vectors in α may completely change, thus suggesting that the feature space over all missing entries in a dataset resembles a landscape with large, flat regions where no gradient information will be of use, interspersed with critical points at which the values of α jump in a discontinuous manner, and accordingly the landscape in R, although in general continuous, will not be differentiable. Thus, in this context, even if gradient information $\nabla_X R$ could be rigorously calculated, it would not be of much use to an optimisation procedure due to the non-smooth nature of the objective function, and we therefore see the suitability to this problem of a non-gradient search method such as PSO.

6.4.3 Comments on results

While both of these methods were seen to work reasonably well in a few straightforward test cases, the main methods we tested in all cases were more consistent with their performance, both in terms of producing reasonable results and in predictability of computational time. Accordingly, we have not included a suite of results for the naïve contrained optimisation method as it did not always converge on a meaningful result; furthermore, we have provided results (marked **'PSJO'**) from the particle-swarm joint optimisation method only in the case of the applied data, as a better performance was recorded in this case versus the simulated structurally incomplete datasets. Only hard margin ($C = 1$) results were included, although by design it would also be perfectly valid to include results gleaned through either method for smaller values of C, thus producing analogous soft-margin results. These methods, however, are not intended to contribute towards the main breadth of this thesis; rather, they have been included here and a limited number of results included from the PSO method to illustrate that such an optimisation is possible. This gives us the important result that given a kernel function, an explicit optimal imputation is usually possible without conditioning upon the provision of a readily-imputed dataset *a priori*. Furthermore, the results produced are in the same form as the full-data SVDD, with the same constraints on α and with the same inherent link between these Lagrangian coefficients and the datapoints themselves, thus giving them greater structural significance than those results produced with the main two methods. Ultimately, however, all the approaches as described in this chapter do compute the same notion of inter-particle distance in a dataset with missing features, and it is thus a valid action to compare their results side-by-side.

6.5 Cross validation

It is important to note that when dealing with any form of classification problem, whether it be in the context of missing data or otherwise, whether we use support vector methods, naïve Bayesian methods or otherwise, ultimately the only statistics that provide a benchmark to the efficacy of a method are to be derived via a *cross validation* procedure. With this in mind, despite the greater part of our results section being devoted to proving that smaller sphere sizes can be achieved via the use of our novel methods, it is also necessary for us to provide CV-based statistics as a sanity check, to show that our methods can compare favourably with uninformed imputations of datasets. Therefore, we shall include CV results in the study of Chapter 8; in particular, the study governing the classification by fissile element, as deriving meaningful CV results from this study is particularly straightforward.

6.5.1 Recap of cross validation

In §2.1 we touched upon this method; here, we shall describe how we have adapted Lee and Lee's domain described support vector classification system (DSVC, see §2.2.8) to provide a small-scale indicative evaluation of our methods via a cross-validation on testing data. Recall the following two definitions of the statistics commonly used to assess a classifier's performance:

$$\text{Sensitivity: } \frac{TP}{TP+FN}\text{; Specificity: } \frac{TN}{FP+TN}$$

Essentially, regardless of context a cross validation procedure is performed as follows: A section of the data (the *testing* set) is taken out and stored to one side, as is its class information, whereupon a classifier is then trained on the remaining

points (the *training* set). The results gleaned from the trained classifier are then used to inform decisions about the nature of the testing points, the idea being that if the classifier is good, then the validation step will produce *predicted* class information that largely agrees with the actual class information that was originally hidden from view.

Missing data The situation is a little more complex within the context of missing data, which motivates this study as a whole. As the training and testing sets must necessarily be separated before any computations are derived from the full dataset, the issue arises of how a testing set with missing data can be correctly classified. Thus, we shall adopt the following paradigm:

Testing data will be imputed *as a whole*, using the methods described (zeros, feature specific means, nearest-neighbours) as if it were a complete dataset in its own right, since we are not permitted to use the known class information;

Training data will be split up into its constituent classes before imputation, then the subset of the full dataset corresponding to each class identity will be imputed as a separate dataset.

PSJO method The use of this method already implies an imputation of the full dataset, and thus we simply apply set splitting on its own.

6.5.2 Shape of windowing function

In order to perform domain described support vector classification, we create a sphere around each class and use *only the support vectors* to create a Gaussian

support function about the dataset based solely on the distance of points in feature space from the centre of our spherical domain:

$$\mathbf{p}(\mathbf{z}|\mathbf{a}, R) \propto \exp(-q\|\mathbf{z}-\mathbf{a}\|^2), \tag{6.24}$$

where, as before, in the full data case we have:

$$\|\mathbf{z}-\mathbf{a}\|^2 \equiv f(\mathbf{z}\cdot\mathbf{z}) - 2\sum_{i=1}^{N}\alpha_i f(\mathbf{x}_i\cdot\mathbf{z}) + \sum_{(i_1,i_2)=1}^{N}\alpha_{i_1}\alpha_{i_2}f(\mathbf{x}_{i_1}\mathbf{x}_{i_2}), \tag{6.25}$$

where $\mathbf{a} \equiv \sum_{i=1}^{N}\alpha_i\phi(\mathbf{x}_i)$ and R is the kernel distance to a support vector. It quickly becomes obvious that it is incorrect to take one universal scaling parameter q, since we would prefer all our support windows to be conceptually "the same size". Lee and Lee [44] advocated the use of the *pseudo-density* function that we adapted in §2.4.3 by subtracting the value of the density function at the support vector points to give them null density, but this does not ensure the windows themselves are of a similar shape. Radii with larger length scales will end up flat and heavily curtailed using this method; to ensure homogeneity of window shape we shall investigate the scaling of $q \equiv \frac{1}{2\sigma^2}$ required to ensure inter-class equality of the *integral* below the probabilistic curve. This shall also have the advantage of not unnecessarily penalising smaller radii.

Spherical integrals We wish to achieve homogeneity of the integral within the sphere of the area under the required Gaussian probability density function. Considering a normal distribution with original mean and unit diagonal variance in the required dimension d, we note:

$$I_d \equiv \frac{1}{(2\pi)^{d/2}}\int\int\limits_{\mathbf{x}\in\mathbf{B}_1^d}\cdots\int\int \exp\left(\frac{-1}{2}\|\mathbf{x}\|^2\right)dx_1dx_2\cdots dx_d, \tag{6.26}$$

where we define $\mathbf{B}_s^d := \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2^2 < s^2\}$, thus the domain here is the *unit* hypersphere in d dimensions. Using a change of co-ordinates to a spherical system we can rewrite this as follows:

$$I_d \equiv \frac{1}{(2\pi)^{d/2}} \int_{r=0}^{1} \int_{\phi_1=0}^{\pi} \cdots \int_{\phi_{d-2}=0}^{\pi} \int_{\phi_{d-1}=0}^{2\pi} \exp\left(\frac{-r^2}{2}\right) |J| d\phi_{d-1} \cdots d\phi_1 dr, \tag{6.27}$$

$$\text{where } |J| = r^{d-1} \sin^{d-2}(\phi_1) \sin^{d-3}(\phi_2) \cdots \sin(\phi_{d-2}).$$

The inner integral is trivial and gives a factor of 2π; for the remaining integrals, we can use the separability of the expression under integration to split this into a product. Furthermore, using the fact that:

$$\int_0^{\pi} \sin^m(x) dx \equiv \frac{\sqrt{\pi}\Gamma\left(\frac{m+1}{2}\right)}{\Gamma\left(\frac{m+2}{2}\right)}, \tag{6.28}$$

we see this is a telescopic product whose terms nearly all cancel, giving a factor of $\pi^{d/2-1}/\Gamma(d/2)$. After making the substitution $s = \frac{1}{2}r^2$, the integral in r is also simplified, leaving:

$$I_d \equiv \frac{1}{\Gamma(d/2)} \int_{s=0}^{\frac{1}{2}} s^{\frac{d}{2}-1} \exp(-s) ds = \hat{\gamma}\left(\frac{1}{2}, \frac{d}{2}\right), \tag{6.29}$$

where $\hat{\gamma}$ is the regularised *lower incomplete gamma function* defined as $\hat{\gamma}(x, a) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} \exp(-t) dt$, which is also the definition used in MATLAB for the function `gammainc(x,a)`.

The chi-squared distribution To scale this result in σ or R, we consider the χ^2 probability distribution, which uses the lower incomplete gamma function as its cumulative density function. Specifically, for a random variable Q^2 distributed according to a χ^2 distribution with d degrees of freedom, we have:

$$\mathbf{p}(Q^2 < r^2) = \mathbf{p}(\|\mathcal{N}(\mathbf{0}, I_d)\|^2 < r^2) = \hat{\gamma}\left(\frac{r^2}{2}, \frac{d}{2}\right), \tag{6.30}$$

where here we use the definition of the χ^2 distribution as the sum of squares of Gaussians, or equivalently the 2-norm of a d-dimensional Gaussian distribution. We may now use the frequentist definition of probability to conclude:

$$\int_{\|\mathbf{x}\|<R} f_d^Z(\mathbf{x}) d\mathbf{x} = \hat{\gamma}\left(\frac{R^2}{2}, \frac{d}{2}\right), \tag{6.31}$$

where $f_d^Z(\mathbf{x})$ is the pdf of a d-dimensional standard normal distribution and we have used the fact that its integral over $\mathbf{x} \in \mathbb{R}^d$ is equal to unity. Replacing each x_i by $X_i = \sigma x_i$ throughout, so that $\prod_{j=1}^d dx_j = \frac{1}{\sigma^d} \prod_{j=1}^d dX_j$, we can show with straightforward algebra that R scales with σ irrespective of dimension, leading us to the main result of this section, which shall guide our choice of σ to choose for the support windows:

$$\frac{1}{(2\pi\sigma^2)^{d/2}} \int\int_{\mathbf{x} \in \mathbf{B}_R^d} \cdots \int\int \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{x}\|^2\right) d\mathbf{x} = \hat{\gamma}\left(\frac{R^2}{2\sigma^2}, \frac{d}{2}\right). \tag{6.32}$$

To apply this to choosing an adaptive σ based on the known value of R, we can invert this function so that we may pick a σ such that $100(1-\alpha)\%$ of the probability mass will be found within the enclosing sphere, for a value of α we shall choose (in the experimental results we choose $\alpha = 0.05$). Thus:

$$\frac{R^2}{2\sigma^2} = \hat{\gamma}^{-1}\left(1-\alpha, \frac{d}{2}\right) \Longrightarrow \sigma^2(\alpha; R, d) = \frac{\frac{1}{2}R^2}{\hat{\gamma}^{-1}\left(1-\alpha, \frac{d}{2}\right)}. \tag{6.33}$$

6.5.3 Other parameters

The following is a list of other parameters which are tunable in the cross-validation procedure, and the values we use here for them:

- `iterations`: the number of times we stochastically choose a selection of the dataset for a training/testing data split (**200**).

- `rejectlevel`: we incorporate a reject class to ensure that the classifiers are not forced to pick one of the existing classes, which can improve specificity for those classes which would otherwise be erroneously chosen ($\mathbf{0.01}^d$).

- `alpha`: as above, we set this at **0.05**, meaning 95% of the probability mass will be contained in the hypersphere we derive.

- `C`: We set the cost parameter for SVDD at $\mathbf{C} = \mathbf{1}$ throughout, but this is not crucial and soft margins are consistent with this method.

- The *proportion of data* in the testing set (**20%**).

Furthermore, we define the dimension d as the number of non-infinitesimal eigenvalues of the kernel matrix of a typically imputed dataset.

6.6 Method preparation

The process for the dual-optimisation method is slightly more complex. From preliminary trials, the process seems to be very sensitive to local minima, and thus merely starting from the requisite α in each case yields a null optimisation process, with the final point not having moved from the initial guess. Thus, some care must be taken with regards to initial point selection. We use the following process. For each type of imputation and appropriate kernel, retrieve the vector α given as the optimal value in the original comparison test. Then substitute this vector into the deviation function for the DO method as given in Equation 6.6 to find the geometric-margin based *maximal* distance with respect to this 'target guess', using as X^* the corresponding specific imputation of X. One type of

imputation method will work best – that is, induce a minimal max-deviation with respect to its own dataset. This particular imputation of X we now denote as our X^* and perform the following pseudo-code process:

- Initialise α_0 as some normally-distributed $(M \times 1)$ vector, with M the size of X^* (where here clearly $M = N$).
- Initialise $\alpha = \alpha_0$, $u = 0$, $t = 0$, $v = \max_i \mathbf{d}_i$ the maximal distance computed using Equation 6.6, using X^* and α_0.
- While $(t - u) < \tau$: (otherwise break; this ensures a count since last update. $\tau = 10$ is reasonably effective)
- $t \leftarrow (t + 1)$
- $\alpha_p = \alpha + \sigma_p Z_N$, where Z is an N-length normally-distributed noise vector, and $\sigma_p = 0.1$ is effective.
- $d_p = \max_i \mathbf{d}_i$ where this is computed in the same fashion as above but using α_p as the coefficient vector.
- If $d_p < v$: $\alpha \leftarrow \alpha_p$, $u \leftarrow t$, $v \leftarrow d_p$, otherwise no update.
- End 'while' loop and thus repeat until condition is breached.

After this process, we can take our initial α as being the last value obtained by α: that is, one that was unbeatable in τ consecutive Monte-Carlo style iterations. The MATLAB function `fminimax` can now be used with this α as the starting point. To ensure speed of convergence we generally have also set `options.MaxFunEvals=300` or similar, as by 300 evaluations of the function described by $\max_i \mathbf{d}_i$, a reasonable value has usually been reached.

6.7 Results on synthetic data

Table 6.1: DODD results: 2-D cluster with value filter

	Linear kernel	Quadratic kernel
Hard, Imp 0	1.25289	1.88347
Hard, Imp μ	1.25289	1.88347
Hard, Imp NN	1.25289	1.88885
Hard, XC	1.25289	2.36436
Hard, DO	1.25289	1.88347

Table 6.2: DODD results: Two disjoint 2-D clusters

	Linear kernel	Quadratic kernel
Hard, Imp 0	2.67234	11.11552
Hard, Imp μ	2.23574	11.07670
Hard, Imp NN	2.23574	11.07670
Hard, XC	2.23057	10.57009
Hard, DO	2.23039	10.58504

Table 6.3: DODD results: One common dimension

	Linear kernel	Quadratic kernel
Hard, Imp 0	4.29481	24.42394
Hard, Imp μ	2.32852	21.03813
Hard, Imp NN	2.52866	23.25400
Hard, XC	2.31592	20.26059
Hard, DO	2.31557	19.67628

These tables show similar results to those gained in the previous chapter with the exact-centre method, and we have included the results given there by way of side-by-side comparison. In Table 6.1, as with the previous methods, no improvements are to be seen in either the linear or quadratic cases with the dataset exhibiting value filtering; again, this is not a bad result as we have shown our methods to not underperform imputation. Table 6.2, referring to the two disjoint clusters dataset, shows a small improvement in the linear case and a more significant one in the quadratic case, with both of our methods performing equally well; finally, an even more encouraging result is shown in Table 6.3, where the DO method achieves a smaller radius than XC, which itself outperforms all imputation methods.

General comments The second method we have described attempts to rectify this by performing a dual optimisation process: that is, optimising an objective given a parameter value, and then optimising this value itself, so that a greater dimension of a given feature space: indeed, as shown above, *any* given feature space we care to consider, can be explored. Its main drawbacks involve the lack of provision for hard margins as a result of breaking the correspondence between the

weighting coefficients α_i and the data, as is the case with the Wolfe dual problems described above in the full data case.

The remainder of this document will deal with the dataset provided for us by the external sponsor, and the results gleaned will include runs from both our exact-centre and dual-optimisation methods, with a joint optimisation PSO approach being used in some cases to demonstrate that our geometric margin approach can hold up to a full cross-validation test.

Chapter 7

Radiation emissions data

7.1 Introduction

For the purposes of this investigation, the Non-Destructive Assay (NDA) group collected two sets of data from a range of experiments involving spectral analysis of radioactive material, in the hope that algorithms could be provided allowing remote analysis of the composition of the material being processed. Two major properties of the experimental data were seen by the NDA group as being desirable for classification: that of the nature of the *fissile material* and also that of the *shielding method* being used. The data consignment was provided by an external third-party and the author of this thesis was not involved in the data collection process. Its constitution will therefore be described in full in Appendix A. In this chapter, we shall describe the two types of experimental measurement found in the data suite which we intend to use in our analysis, and give a full description of the physical processes involved in each, thus providing crucial background concerning the selection of features we shall make in the next chapter. The two

categories of data structure we have chosen to consider are spectroscopy counts and neutron arrays. Much of the information we give in this chapter is summarised from sections of a comprehensive text on this subject by Glenn Knoll [41].

7.2 Spectroscopy data

7.2.1 Gamma rays

Definition Gamma rays (γ-rays) are electromagnetic (EM) waves that occur at the highest frequencies of the spectrum. They are one of three main forms of emission produced when radioactive materials decay, the others being α- and β-particles. Unlike these other emissions, however, γ-rays do not carry any electromagnetic charge, nor do they have any intrinsic mass; when a material decays in a γ-ray manner, neither its atomic number nor mass changes. Gamma rays occur in the form of photons, and are produced as a by-product of α- or β- particle decay, as a result of excess energy needing to be released from the nucleus after the α/β-particle emission. The energies of γ-ray photons vary between $10^1 - 10^5$ keV; they are the most penetrating form of ionising radiation. Thick ($\sim$ 100mm) lead or concrete is often used to shield users during experiments that produce γ-ray emissions.

Spectroscopy When a radioactive element decays producing γ-ray radiation, certain distinct energies of photons are produced in varying amounts. The resultant spectrum showing the relation between photon energy and activity is unique to one isotope, and thus studying a γ-ray emission spectrum is a method of deducing the material that is decaying: this is known as *gamma ray spectroscopy*.

Radiation is collected over a set time period to produce a time-independent spectrum of the γ-ray signature. A γ-ray spectrometer can be defined as any system where the energy spectrum of incoming γ-ray photons is measured over a period of time: this will generally involve causing the γ-ray to cause another particle to produce activity, such as a photoelectron. The spectrometer can then generate an electrical pulse whose size can be readily measured. Spectrometers fall into two main types of material in their construction: scintillators (organic) and semiconductors (inorganic). The differences between these in terms of readings and other properties are summarised below in §7.2.4, but first we explain here how γ-rays progress from their emission from the fissile material's nucleus to the statistics on the histogram.

7.2.2 Interaction with matter

In order to be measured, γ-rays must first be converted into *photoelectrons*, which in turn are amplified and processed into detector counts by a photomultiplier tube (PMT), producing the energy-activity spectrum. Photoelectrons are produced through interactions between the γ-rays and the electrons of the atoms within a specially manufactured crystal inside the detector. These interactions can happen in three different ways: photoelectric absorption, Compton scattering and pair production.

Photoelectric absorption This occurs when a γ-ray comes into contact with an electron - generally in the K-shell – orbiting an atom of detector material; it transfers all its energy to the electron and thus disappears. The photoelectron is then detected with energy equal to that of the incident photon, minus the binding

energy required to release it from orbit, E_b. This binding energy is then released once the remaining electrons reassemble, usually through a characteristic X-ray, which is then absorbed itself; if nothing escapes from the detector, effectively "all" the γ-ray energy is converted into detectable form. Photoelectric absorption could thus be said to be the most "preferable" interaction: if an energy-count graph were to be plotted where only absorption were taking place, only a *full-energy peak* at exactly the γ-ray energy would be shown (see Figure 7.1).

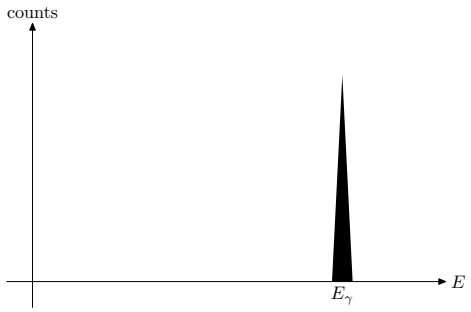


Figure 7.1: Photoelectric absorption

Compton scattering This occurs when the γ-ray instead comes into contact with an unbound (free) electron. Instead of transferring all of its energy for detection, only partial absorption occurs and a lower-energy γ-ray escapes, unlike the absorption case. The proportion of energy transferred depends on the *scattering angle*, θ, i.e. the angle through which the γ-ray is deflected. Two extremes can be identified: where $\theta = 0$ and the γ-ray passes through without loss of energy; and where $\theta = \pi$ (a 'head-on collision') and the γ-ray transfers the maximum possible energy to the recoil electron. The latter is defined as $E_{max} = E_\gamma - E_c$, where E_c is the maximum recoil energy; it is *not* possible for a Compton recoil electron

to absorb all of the energy of the γ-ray. Thus, in an energy histogram exhibiting Compton scattering, a continuum of energies are present between $E = 0$ and $E = E_\gamma - E_c$, depending on the distribution of values of θ. The polar probability distribution of these recoil angles is as follows:

$$\Pr_d(\theta) \propto \frac{d\sigma}{d\Omega} = Zr_0^2 \left(\frac{1 + \cos^2\theta}{2(1 + \alpha(1 - \cos\theta))^2} \right) \left(1 + \frac{\alpha^2(1 - \cos\theta)^2}{(1 + \cos^2\theta)(1 + \alpha(1 - \cos\theta))} \right) \tag{7.1}$$

Here, α is the energy of the photopeak γ-ray divided by 511 keV, the rest mass of an electron (m_0c^2; α is nondimensional), $\theta \in [0, \pi)$ is the scattering angle, Z is atomic number and r_0 is the radius of an electron. Sampling θ from this distribution and combining with the recoil energy equation given by the following:

$$E_{e^-} = h\nu \left(\frac{(h\nu/m_0c^2)(1 - \cos\theta)}{1 + (h\nu/m_0c^2)(1 - \cos\theta)} \right) \tag{7.2}$$

will produce a continuum of energies known as the *Klein-Nishina cross-section* (see Figure 7.2).

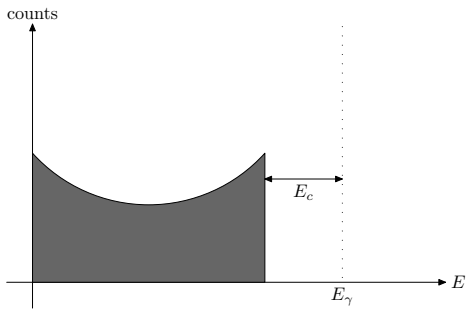


Figure 7.2: Compton scattering

Pair production This occurs if the energy of the incident ray is large enough to reach the intensely-charged region (*Coulomb field*) around the nucleus of the

absorbing material. Its energy may then be used to produce an *electron-positron pair*, which according to relativity theory has combined energy $h\nu - 2m_0c^2$, i.e. that of the incident γ-ray, less the energy required to produce the mass of the pair, where m_0 is the mass of an electron (or positron). As $2m_0c^2 \approx 1.02$MeV, this process therefore can only occur at γ-ray photon energies above this. The electron's kinetic energy is detected as per the previous two cases; however, the positron is antimatter surrounded by "hostile" electrons, and thus quickly annihilates, producing two *annihilation photons* each of energy m_0c^2. These photons then escape from the detector, with only the kinetic energy from the original pair being detected. This results in a *double-escape peak* which is $2m_0c^2$ short of the energy of the γ-ray (see Figure 7.3).

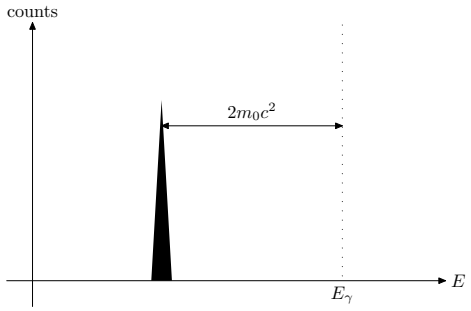


Figure 7.3: Pair production

7.2.3 Real-life spectra

In practice, when measuring emissions spectra of γ-rays, all of the above three reactions may be taking place, with some energy being detected in each case and some escaping. The size of the detector must therefore be taken into account,

since there will be 'by-product' γ-rays which may take part in multiple interactions at lower energies. At one extreme is a hypothetical 'large' detector, which conserves the total input. In this case, a full-energy peak will be the only feature on the resultant histogram. In practice, some photons will escape; in a hypothetical 'small' detector, exactly one reaction is allowed to occur for each detected γ-ray. Assuming only one distinct frequency of radiation is provided, two subcases arise: If $h\nu > 2m_0c^2$, pair-production effects would show, otherwise only the Compton continuum and full-energy peak would be present. In reality, a detector will allow both, producing multiple Compton events and/or a single escape peak where one of two annihilation photons escapes, occurring at $E = E_\gamma - m_0c^2$ (see Figure 7.4).

Useful features It was suggested that the useful features which could be taken from this spectroscopy data would involve the study of the location of the *Compton edge* – where it exists, this is defined as the termination point of the visible Klein-Nishina cross-section as described in the 'Compton scattering' paragraph of §7.2.1 above. We describe the feature extraction process to isolate an estimate of this feature in §8.2.1. Secondly, it was observed that although in general, the position of the Compton edge is more a feature of the fissile material being studied, the *shape* of the cross-section prior to this point is heavily affected by the nature of the shielding method being used; we thus created an estimate of this feature, related to the area under the graph of an energy histogram upto the estimated Compton edge location, and describe the algorithm we used for this in §8.2.2.

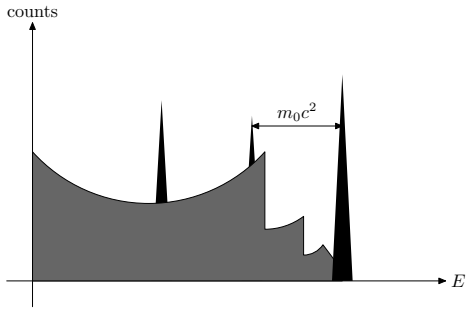


Figure 7.4: Multiple Compton events, single escape peak

7.2.4 Detector differences

The essential requirement of any spectrometry system is that of *linearity*. Since the energy of the γ-ray must first be translated into the kinetic energy of a photo-electron before it can be detected, it is vitally important to ensure that the detector achieves this with a near linear relation between its *channel* number (i.e. the energy bins used for classification of the various frequencies of incoming rays) and the actual energies produced. This requires a step of *calibration* of a detector; this will be described in more detail in Appendix A. Two *desirable* traits are efficiency and resolution of the detector. Other general factors which may affect choice of detector material are that of price - which can vary significantly, and is of particular concern if large-scale tests are to be conducted; the temperature at which the detector material can be operated; the sensitivity towards the ambient temperature; the size of unflawed crystals that may be produced (important for semiconductors) and the sensitivity towards bias voltage (how the power required to operate the detection device affects the output - this is particularly relevant with photomultiplier tubes amplifying results from scintillators). All these factors are

used in the following comparison of the two detector types. For the purpose of the

Table 7.1: Comparison of detectors

	Scintillator	Semiconductor
Example	NaI (sodium iodide)	HPGe (germanium)
Material nature	Organic	Inorganic
Operating Temp.	300K (r.t.p)	77K (liquid N)
Price	Cheaper	(factor of 10)
Manufacture	Easy to mass-produce	Difficult
Temp. Sens.	Sensitive	Insensitive
Bias Voltage	Sensitive	Insensitive
Resolution	Poor (8%)	Excellent (0.15%)
Efficiency	More efficient	Less

experiments we analyse, the main discriminating factors between the two detectors used were that the NaI detectors showed only general shape, with the specific peaks taking many more energy bins to describe and as a result sometimes not being visible; the HRGS detectors, however, picked out these peaks very specifically (see Figure 7.5). This is known as the spectral *resolution* and is a desirable trait in spectrometers; the main drawbacks with the semiconductors are not in their experimental readouts but in considerations of practicality and cost, as outlined above. Statistically, a peak's well-definedness can be derived through the measure of *full width at half maximum* (FWHM). As the name suggests, this measure is used to denote the width of a peak at the benchmark point of half of its maximal amplitude. If the peak is a perfect Gaussian shape, the following equation relates

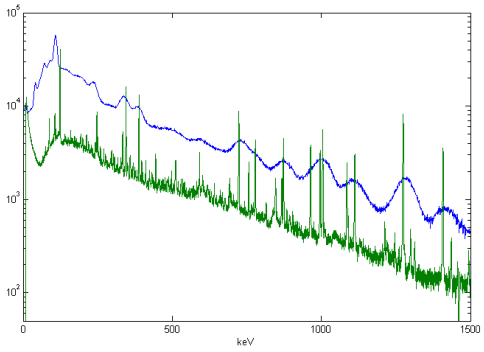


Figure 7.5: Comparison between NaI and HRGS detector

the FWHM to the variance:

$$\text{FWHM} = 2\sqrt{2\log 2}\sigma \approx 2.35482\sigma. \tag{7.3}$$

Clearly, smaller values of the FWHM are more beneficial to spectral signature readings. The manufacturer's rating of a spectrometer's *efficiency* is based on three main factors – it is marked higher if:

- the amount per unit activity of γ-rays removed from the beam (analogous to the detection probability mentioned above) is high;
- a large amount of detector material is available;
- the signal can be readily extracted for processing into a spectrum.

7.2.5 Calibration information

In §7.2.4 above, we mentioned that the *linearity* of a spectroscopy system is important. The *calibration information* is an expression of the linearity of a detector

of this type: that is, how good the representation is in the collection bins relative to the actual spectrum being produced. It takes the form of three coefficients a_0, a_1, a_2. These coefficients are devised via a least-squares fit on known spectral data, to calibrate a detector. They provide the following bestfit relation between the channel number k and the energy E of the incident γ-ray:

$$E = a_0 + a_1 k + a_2 k^2 \tag{7.4}$$

Thus, in an ideal world it will be the case that $a_0, a_2 \approx 0$, with a_1 providing the required linear relation. However, if these coefficients are known, analysis is usually straightforward, especially if the calibration coefficients are the same between the background reading and spectral reading (see below).

7.3 Neutron data

In addition to the spectroscopy readings as detailed above, neutron emissions data was simultaneously taken for many of the experiments. Four detector panels were used to collect neutron readings, which were fed into a Time Correlation Analyser (TCA). We describe their interaction with matter in this section, and explain the form of the data which was received.

7.3.1 Interaction with matter

The detectors operated with Helium-3 gas-filled tubes at 10atm pressure; this type of neutron detector is used to gathering neutrons of far lower energies (0.025 eV) than those produced in fission events (2 MeV), thus the tubes were initially surrounded by high-density polyethelene (HDPE), which has sufficient hydrogen

content to "slow down" these particles to those that a Helium-3 tube may detect. The HDPE scatters the neutron energy many times over to reduce it to the 'thermal' level of 0.025 eV, whereupon it may be captured by the He-3 gas and be detected: this process takes of order $O(10^{-5})$s. Once the slowed neutron has been captured by an atom within the helium gas, a charge is deposited within the tube and accelerated by the bias voltage towards the anode and cathode. An amplifier/discriminator unit then sends a 5V pulse to the TCA.

Thus, an experimental run-through of a fission/detection process will involve a *stream of pulses* being sent to an analysis machine, where each pulse would represent a *point in time* at which (a few tens of microseconds previously) a fission event had occurred of neutron type within the nucleus of the fissile material being studied. Note the immediate differences from the γ-ray spectroscopy described above: with spectroscopy we have full information about the different energy levels at which γ-rays were being produced and a set time period over which to measure, but no information about when these emissions took place within that time period (as it is less important to spectroscopy readings); with neutron experimental data we shall expect ample information about the times at which events happened but we do not attempt to provide information about characteristic energies, since this information will have been lost via the "deceleration" process of the HDPE casing, in order to make the neutrons detectable to begin with.

7.3.2 Multiplicity arrays

Once the pulses are sent to the TCA, they must be processed to provide meaningful information. Although, in principle, it would be possible to record the exact times at which every reaction occurs, this information is largely redundant: more

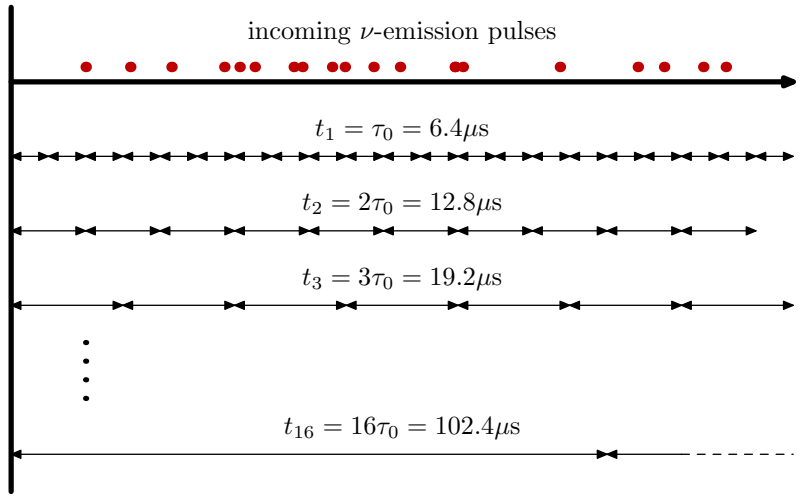


Figure 7.6: Operation details of the TCA

informative (and succinct) is a process of recording *multiplicities*. The function of the TCA is, at the simplest level, to open a time gate, count the number of pulses received during this window, close the time gate and record its result. The TCA used in these experiments is capable of opening 16 simultaneous time gates of different widths (see Figure 7.6). It requires the shortest time gate to be specified: we refer to this as $t = \tau_0$, and the 16 time gates will then be the scalar multiples of this fundamental timescale: $t \in \{\tau_0, 2\tau_0, \ldots, 16\tau_0\}$. In terms of this suite of data, it was decided to let $\tau_0 \equiv 6.4\mu$s.

Periodic triggers Readings from three separate *trigger* methods were used: the *periodic*, *signal* and *delayed* trigger methods. Hereafter, we refer only to those readings taken from the periodic trigger experiments, as this suite was judged to be sufficient for discrimination between properties of the different experiments. Figure 7.6 illustrates how the periodic trigger functions. The raw, time-series data

Table 7.2: Typical neutron multiplicity array

Multiplicity	Gate τ_0	Gate $2\tau_0$	Gate $3\tau_0$	...	Gate $16\tau_0$
0 ('zeros')	223.4m	110.9m	73.4m	...	12.5m
1 ('ones')	1.54m	1.52m	1.51m	...	1.37m
2 ('twos')	43775	52177	58646	...	118305
3	260	622	963	...	7457
4	4	16	28	...	478
5	0	0	0	...	19

is shown at the top of the diagram, with the time gate windows depicted below. For each time gate, a *multiplicity histogram* is compiled to show the frequency of the reception of 'zero' neutrons within that particular window; 'one' neutron event; and so on. Consider, for example, the event that nothing happens initially, then a pulse is received at $2.2\tau_0$: if we let the detector run for a time period of $4\tau_0$, we will see the first time gate record $t_1 = (0, 0, 1, 0)$ and thus produce a histogram of $h_1 = (3, 1)$. Similarly, the $2\tau_0$ time gate will record $t_2 = (0, 1)$ and thus produce $h_2 = (1, 1)$. Over a sufficient amount of time, these histograms will thus be expected to approximate a *Poisson process*, with means linearly scaling according to time gate. The following table shows an example of a typical periodic trigger multiplicity array: This experiment was taken from a Californium-252 source with aluminium shielding, over a time period of 24 minutes = 1440 s. The sum of the events corresponding to the first time gate show that the gate was opened 225,000,000 times: at a time gate of $\tau_0 = 6.4\mu$s, this figure is indeed equal to 1440 s. The same verification can be performed for each of the time gates. If we take the weighted sum with respect to the multiplicities on any

column (which is valid since the time gates were opening simultaneously over the same experimental period), we find that this source, over the total time allowed, produced 1,630,000 fission events in total. Thus, if this were a purely Poisson process, we expect to have mean multiplicities over the time gates roughly equal to $\lambda_k = k\frac{1.63m}{225m} = 0.007244k$, which we find, for this experiment, is in fact very accurate. We shall later see, however, that the process does not produce an exact replica of the expected Poisson distribution for these means, and the deviation from the expected distribution is a useful feature to be able to measure: we have included a description of this later in the section on meaningful feature extraction, in §8.2.4.

7.3.3 Data files

During the course of any given experiment, as with the spectroscopy datasets, there will be assigned one of 8 *master time scales*, ranging from $T = 300$s to $T = 14400$s. The neutron data was collected in parts over the course of this timescale – again, this is in contrast to the spectroscopy data, which was simply left to run over the entire time scale and energy histogram information collected at the end. With the neutron readings, T is broken up into ten constituent parts t_i of length $T/10$. A multiplicity array is taken for each of the t_i from $i \in \{1 \dots 10\}$ and every multiplicity matrix recorded as an independent experiment; this is to incorporate any time-specific feature information that may be of use. Thus, for example, the longest time scale is $T = 14400\text{s} = 240\text{m}$ so each of the $t_i = 1440\text{s} = 24\text{m} \forall i$.

Useful features It was suggested that the best features to consider within these multiplicity arrays would include the mean multiplicity rate (as described above)

and the 'tail' properties: that is to say, how far the tail of the distribution typically extends beyond the mean multiplicity. We have taken this to be virtually equivalent to a study of the deviation from the expected Poisson distribution with the given multiplicity mean and describe the extracting of these features in §8.2.3 and §8.2.4.

Chapter 8

Tests on emissions data

8.1 Feature extraction

In its raw form, we cannot hope to be able to make any immediate inferences from the data in terms of a potential design of a classification machine. This is where a thoughtful and appropriate process of *feature extraction* is important. In general, feature extraction can be defined as any process which reduces high-dimensional data – which, in general, is suspected to be highly correlated – to a 'small' set of manageable features, usually themselves with some physical significance. This reduces the redundancy inherent in most 'real-life' datasets; indeed, feature extraction is most often used as an initial pre-processing step in the field of image processing. In general, the exact methods of feature extraction will be heuristic, subjective and will rely heavily on the context of the data set and the prior information which is known: in our case, we know our data comes from detectors, and we have given information previously on the meaning of these readings. This allows us to make intelligent decisions as to the best ways of extracting features. A tech-

nical report due to the Lawrence Livermore National Laboratory [29] performs a similar task based on classification of gamma-ray spectra; Hilario et al. [34] give a good example of this prior information being used in a similar situation: that of protein mass spectra. For the general case, this may sometimes be more difficult; to address this, Guyon et al. [31] give a useful flowchart-style set of instructions, which we partly reproduce here:

- If there is knowledge of the domain (the case of our data is an example of this), then construct a list of 'ad hoc' features.
- Normalise the range and numerical position of the features – it is usual to *standardise*, that is ensure that $\mu_j = 0, \sigma_j = 1 \forall j$ where j is the dimension number of input space.
- If *interdependence* of features is suspected, it is likely there will be some correlation between the feature values. The above alignment methods, such as PCA, can assist with finding a more independently-aligned set of products of features.
- If necessary, either *prune* the numbers of features or construct *extra* features made of weighted sums.
- Reduce noise if this is suspected within the dataset, according to some heuristically chosen method.

In addition to the above flowchart, well-used statistical methods of feature selection based purely on numerical properties of the dataset are also common, a good example of this being due to Kira et al. [40]. The dimensionality reduction methods we covered in the previous sections, §2.2.4, §2.2.5, and their kernel

equivalents, briefly covered in §2.3.4, are very widely used in all forms of feature extraction, especially when dealing with highly correlated image data. A few examples of this can be found in Liang [45] (in the context of kernel discriminant analysis, the kernel form of LDA), Xu [85] (where the kernel form of PCA is used to aid in the feature extraction process) and Honkela [35]. As an alternative method for extracting information, *wavelet analysis* [51] is a relatively modern process used for this purpose: Sullivan [75] uses this Fourier-transform inspired method to deal with similar spectra to the ones we study. A comprehensive review of statistical pattern recognition methods – a category under which all these methods fall – is due to Jain et al. [36]. The following two sections will describe the features we have chosen to select in the context of the data from the provided experiments, and the reasons behind these choices; we shall then explain how we have pre-processed this data to achieve an easily-analysable dataset. Finally, we perform our classification methods upon this processed dataset, and note the comparisons with imputation methods in a similar way to those results given in §5.6 and §6.7.

8.2 Selected features

8.2.1 Compton edge position

In Figure 8.1 we consider three separate spectra taken from sources of californium, uranium and plutonium (in blue, green and red respectively). By inspection, we see that for the uranium and plutonium spectra, the point at which the cross-sectional portion of the reading ceases – the *Compton edge* – is clearly visible; in fact, the position is different according to the fissile element being studied. In

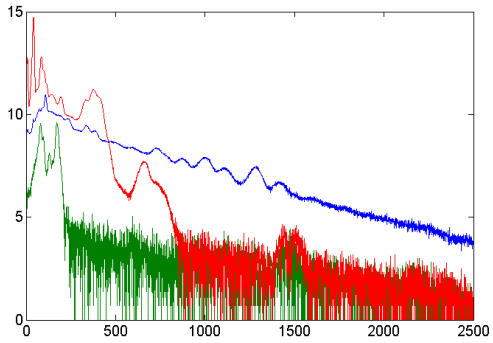


Figure 8.1: Location of Compton edge by fissile material

the californium spectrum, however, no such edge is apparent. We describe an algorithm to approximate the location of the Compton edge, and later we shall use it in our feature extraction process.

Code The following list summarises the code that is relevant to extracting this first feature:

1. Consider a spectrum vector $\mathbf{s} \in \mathbb{R}^N$, where $N = 4096$ for an NaI spectrum, or $N = 16384$ for an HRGS spectrum. Instead of the usual notation, label the entries of $\mathbf{s}$ as $\mathbf{s} = \{s_i\}_{i=0}^{(N-1)}$ (use this notation for all subsequent vectors of length N).

2. For this spectrum, retrieve the following information: its timescale $T_s \in \mathbb{R}$ in minutes, its relevant background $\mathbf{b} \in \mathbb{R}^N$, the timescale of the background T_b, and the calibration coefficients (which must be commensurate between spectrum and background) a_0, a_1, a_2.

3. Create a calibration vector $\mathbf{c} \in \mathbb{R}^N = \{c_i\}_{i=0}^{(N-1)}$ via $c_i = a_0 + a_1(i) + a_2(i^2)$.

4. Perform background cancellation and time normalisation via $s'_i = \frac{s_i}{T_s} - \frac{b_i}{T_b}$.

5. Apply the map $f(s) = \text{sign}(s)\log(1 + |s|)$ to each member of the vector $\mathbf{s}'$. Thus $\mathbf{n} = f(\mathbf{s}')$.

6. Standardise this vector so that it has zero mean and unit variance.

7. Let the series $n_j^0 = \sum_{i=1}^{j} \delta^0(n_i)$, where $\delta^0(n_i)$ takes the value 1 if $n_i < 0$, otherwise it is zero. That is, this vector is a cumulative count of the number of values $\mathbf{n}$ takes which are below zero.

8. Decide on a series of threshold levels $\{t_k : k \in \{1 \dots N_T\}$. For each threshold level t_k, let the feature value corresponding to this threshold be the value that $\mathbf{c}$ takes at the point where the vector $\mathbf{n}^0$ reaches the proportion Nt_k. For example, if $t_k = 0.1$, the feature value will correspond to the point (in keV) at which the first $0.1N$ of the values of $\mathbf{n}$ were below zero.

8.2.2 Area under graph

In Figure 8.2 we now consider various different types of uranium spectrum. The bare spectrum, denoted here in blue, is an equivalent, 'zoomed-in' image of the green uranium spectrum shown in Figure 8.1; the remaining spectra show the effects of different shielding methods (of the form described above) applied to the source. We see that although the Compton edge stays the same throughout (occurring at $\approx 210 - 220$ keV), the *shape* of the spectral readings in the portion before the Compton edge seem to differ according to the shielding method used. We describe an algorithm to integrate below each curve, with a right-hand limit denoted by the feature found in §8.2.1. As shielding methods are also important to

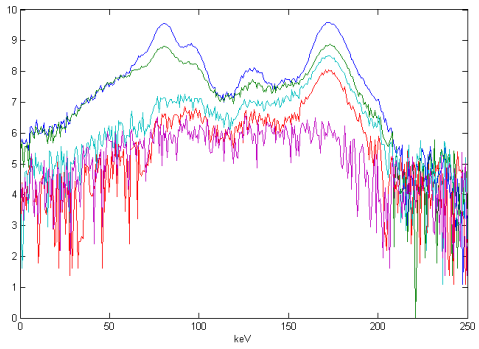


Figure 8.2: Difference in areas under graph by shielding method

our study, we shall include this feature next in our dataset. Extracting this feature, once the steps for extracting feature 1 have been followed, is straightforward: simply sum the points under the curve between zero and designated points along the energy axis. For the purposes of this experiment, three cut-points were applied, at 250, 500 and 750 keV, and area under the graph summed for each.

8.2.3 Neutron features

Neutron data is an integral, and important, part of our data consignment, and as Figure 8.3 shows, it can provide a very useful reading as to the nature of a fissile element involved in an experiment. This graph displays four distinct sections, with the entries along the x-axis denoting different experiments. The graph displays differences between material with respect to the neutron data-based *mean multiplicity* in such a clear way that the distinction between the different sections is visually obvious. In the short initial section, background readings have been taken; in the second section, californium (our element 1) was allowed to decay; in

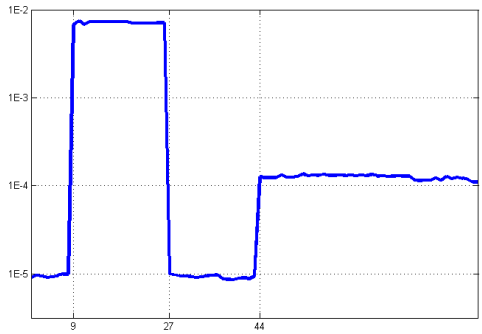


Figure 8.3: Neutron mean multiplicity data, by fissile material

the third section, we consider uranium (our element 2), and in the fourth section plutonium (our element 3). The following describes how the neutron readings involved in Figure 8.3 were obtained.

Neutron data structure In §7.3, we described the interaction of a time-correlation analyser (TCA) detection system with radioactive emissions of the type involving expulsion of neutron particles. Table 7.2 gives an example of how this data is presented to the end-user for analysis. In §7.3.3, we explained how, during any given experiment, the master time scale T was broken up into 10 equal segments of length $T/10$ and neutron arrays were presented for each subsection. We thus arrive, in general, with the following data structure:

$$N \in \mathbb{R}^{k \times 16 \times 10}$$

where k is the maximum multiplicity observed in the entire experiment (over the master time scale T), 16 denotes the 16 separate time gates as being multiple of $\tau = 6.4\mu$s, and 10 denotes each $(T/10)$-second long segment of the experi-

ment. Occasionally the latter will not be the case, as for some experiments (as outlined in the 'Exceptions' paragraph of Appendix A.1) not all ten timeframes were completed, but this is not of great significance, as we shall see in the following descriptions.

Computing mean multiplicity Computing the mean multiplicity feature for neutron data is straightforward. It is calculated thus:

$$\lambda_{jm} = \frac{\sum_{i=1}^{k}(i-1)N_{ijm}}{\sum_{i=1}^{k}N_{ijm}} = \frac{1}{\eta_{jm}}\sum_{i=1}^{k}(i-1)N_{ijm}. \tag{8.1}$$

8.2.4 Neutron tail properties

From this 3-D array N, we compute the following two matrices:

$$\lambda \in \mathbb{R}^{16\times 10}$$

$$\eta \in \mathbb{R}^{16\times 10}$$

Here, η_{jm} designates the total number of *gate openings* involved in the experiment, and λ_{jm} designates the average number of *neutrons per opening* in the experiment, for each different width of time gate and sub-period of the experiment, calculated by dividing the neutron count by the number of openings. They are calculated thus:

$$\eta_{jm} = \sum_{i=1}^{k} N_{ijm}, \tag{8.2}$$

$$\lambda_{jm} = \frac{\sum_{i=1}^{k}(i-1)N_{ijm}}{\sum_{i=1}^{k}N_{ijm}} = \frac{1}{\eta_{jm}}\sum_{i=1}^{k}(i-1)N_{ijm}. \tag{8.3}$$

Note that λ takes this form since the i^{th} member of the first dimension of N describes the data for multiplicity $(i-1)$: row 1, as described above, denotes

'zeros', row 2 'ones', etc. We now compute another 3-D array L from these values, designed to emulate the expected behaviour of a Poisson process should the parameters λ and η be those governing the model, by defining:

$$L \in \mathbb{R}^{k\times 16\times 10}$$

$$L_{ijm} = l_{jm} \in \mathbb{R}^k \tag{8.4}$$

$$l_{jm} \sim \eta_{jm} \cdot \text{Poisson}(\lambda_{jm}) \tag{8.5}$$

$$\text{on the set } \{0, 1, \ldots, (k-1)\}.$$

Thus, we have L as an 'expected model', and N as the 'real data' for our neutron experiment. It is mentioned above that the neutron tail often extends beyond the expected range for a perfect Poisson process, and in order to analyse to what extent this happens, the following array is created:

$$D = \frac{N}{L}.$$

In general, we find that for any given time period $m \in \{1 \ldots 10\}$, the relationship between i and j, given sufficient 'good' data, often describes a curve similar to that of $x \to \frac{1}{x}$. The deviation fraction described by the entries of D decays with the time gate width j in this manner, with a lower limit of 1. Thus, it makes sense to try and fit a rational function to this relation. We use the MATLAB function `lsqcurvefit.m`, from the Optimisation Toolbox, to fit a rational polynomial least-squares fit to the data. The following polynomial is selected:

$$\rho(x) = \frac{p_0 x^m + p_1 x^{m-1} + p_2 x^{m-2} + \ldots + p_{m-1}x + p_m}{q_0 x^n + q_1 x^{n-1} + q_2 x^{n-2} + \ldots + q_{n-1}x + q_n} \tag{8.6}$$

Setting $n_p = n_q = 1$ we have:

$$\rho(x) = \frac{p_0 x + p_1}{q_0 x + q_1} \tag{8.7}$$

Dividing top and bottom through by q_0 we obtain:

$$\rho(x) = \frac{r_1 x + r_2}{x + r_3} \tag{8.8}$$

where $r_1 = \frac{p_0}{q_0}$, $r_2 = \frac{p_1}{q_0}$ and $r_3 = \frac{q_1}{q_0}$. The above method of describing the curve is included here for completeness, as there exist rational polynomial solvers more adapted to handle this general form; however, MATLAB's `lsqcurvefit` provides a general nonlinear solver. We thus use it in the following manner in order to fit this curve:

```
lsqcurvefit(@(s,X)(s(1).*X+s(2))./(X+s(3)),
                [1,0,1],(1:v)',Dm);
```

with the starting point at $(1, 0, 1)$ denoting the curve $\frac{x}{1+x}$. Here, the variable `Dm` denotes the means over all time periods of the entries in the matrix $D^{(i)}_{jm}$, where i is a given multiplicity, whence:

$$\hat{D}^{(i)}_j = \frac{1}{N_m} \sum_{m=1}^{N_m} D_{ijm}, \tag{8.9}$$

where the size of the third dimension of D, denoted by N_m, is usually 10: this shows that under this system, neutron arrays are treated invariantly with respect to the number of time periods contained in the experiment. Of the above parameters r_1, r_2, r_3, observe that r_1 and r_3 are the most significant: this equation describes rational functions of a similar shape to $x \to \frac{1}{x}$ which are singular at $x = -r_3$ and whose limit is r_1 as $x \to \infty$. These r_i are used as features 14-16 (for 'ones'), 17-19 ('twos') and 20-22 ('threes') where these are available; the data for 'zeros' were discarded. Features 23-25 denote the reconstruction error committed by fitting these models for each category respectively.

8.3 Pre-processing

Table 8.1: Summary of features used

Feature number	Detector	Description
1	NaI	Compton edge, threshold 0.05
2	NaI	Compton edge, threshold 0.10
3	NaI	Compton edge, threshold 0.20
$4-6$	HRGS	Compton edge, threshold $\{0.05, 0.10, 0.20\}$
7	NaI	Area under graph, cut-point 250 keV
8	NaI	Area under graph, cut-point 500 keV
9	NaI	Area under graph, cut-point 750 keV
$10-12$	HRGS	Area under graph, cut-point $\{250, 500, 750\}$ keV
13	Neut.	Mean multiplicity λ
14	Neut.	Poisson fit r_1 for 'ones'
15	Neut.	Poisson fit r_2 for 'ones'
16	Neut.	Poisson fit r_3 for 'ones'
$17-19$	Neut.	Poisson fit $r_{1,2,3}$ for 'twos'
$20-22$	Neut.	Poisson fit $r_{1,2,3}$ for 'threes'
23	Neut.	χ^2 error, 'ones'
24	Neut.	χ^2 error, 'twos'
25	Neut.	χ^2 error, 'threes'

An important aspect of data recovery is the ability to process exactly what is necessary, and a set as complex as this cannot be dealt with naïvely. In a raw form, 372 experimental events took place, each containing one or more of the

Table 8.2: Characterisation of feature groups

Feature group	Feature numbers	Identifiable with
F1	1,2,3,7,8,9	f_1, f_2 for NaI
F2	4,5,6,10,11,12	f_1, f_2 for HRGS
F3	13,14,15,16,23	ν multiplicity and 'ones'
F4	17,18,19,24	ν 'twos'
F5	20,21,22,25	ν 'threes'

Table 8.3: Data and feature groups

Datagroup	Member points	F1	F2	F3	F4	F5
D1	1			√	√	√
D2	14			√		
D3	102	√	√	√		
D4	142	√	√			
D5	17	√	√	√	√	√
D6	9	√	√	√	√	
D7	4			√	√	
D8	7	√				
D9	1	√		√		

three categories of readings taken: NaI spectroscopy, HRGS spectroscopy and neutron readings. This was collected into a structural datatype, with different fields denoting the different types of data, headers, etc. Firstly, on inspection of the composition of the whole consignment, an index array was created to denote which background should be used with each reading, by way of cancellation of the radiation which would have been present in the ether at the time the experiment was taken. Special care needed to be taken over this, as the spectroscopy experiments had variable coefficients of calibration, and in order to make background cancellation meaningful, the same calibrations needed to be used when performing this. Information about relative timeframes was also incorporated into this array so that, if necessary, the data could be scaled up to that of the relevant background, so direct spectrum-on-spectrum subtraction cancellation could be performed. An index vector $\{1 \ldots 372\}$ was created, and the following steps were then taken:

- Remove the indices corresponding to background spectra (39 of these). These include the backgrounds with strange calibration coefficients, mentioned earlier.
- Remove the mixed-element readings, since these have no direct comparison opportunities (6 of these).
- Remove those spectra with a hemi-shell interface shine: this was a problem which occurred during the Set 1 readings (8 of these).
- Remove those with fissile element 7, the 'active' Barium source (22 of these).

The breakdown of the distribution of data may then be described as follows:

- By set: 98 in Set 1a, 18 in Set 1b[1], 181 in Set 3.
- By timeframe: (37, 38, 38, 35, 39, 38, 37, 35) occurring with timescales of (5, 10, 15, 20, 30, 60, 120, 240) minutes.
- By fissile element: (45, 69, 59, 39, 39, 46) occurring with element (1, 2, 3, 4, 5, 6).
- By shielding method: (49, 48, 54, 8, 8, 10, 48, 72) with shielding method (0, 1, 2, 3, 4, 5, 6, 7).

Table 8.4: Breakdown of data by class and subclass

	0	1	2	3	4	5	6	7	Total
1	9	8	8				8	12	**45**
2	8	8	8	8	8		8	21	**69**
3	9	8	8			10	8	16	**59**
4	8	8	8				8	7	**39**
5	7	8	8				8	8	**39**
6	8	8	14				8	8	**46**
Total	**49**	**48**	**54**	**8**	**8**	**10**	**48**	**72**	

From these remaining sets of raw data, we can now engage in the task of the extraction of these features, using a system following the recipes as stated above during the feature description sections. Table 8.1 outlines the features that were used to make up the 'raw' dataset. Where a feature was not present because the experiment was not performed on that day, it is left missing, as a challenge for our

[1]Those in Set 1 with different calibration coefficients.

classification machine. The other cause of missingness is when the indicators of a good fit to a $x \rightarrow 1/x$ fit in the neutron data, which we use here as our features 23-25, were not sufficiently low: we have deleted the relevant features where either the χ^2-based fit for 'ones', 'twos' or 'threes' exceeded 1 (sometimes these fits were so bad that their χ^2 error referenced $O(10^6)$ or higher). The only other piece of pre-processing at this stage is to apply the following mapping function to feature number 13, the mean multiplicity of the neutron data:

$$x \longrightarrow \Re[\log(x)] = \log|x| \tag{8.10}$$

This is to cater for the fact that mean multiplicities show a far more useful variance when taken as logarithmic values, while avoiding the production of nonsense results should the background levels be so similar that the feature argument ends up slightly below zero. In this case, it does not matter as it will be classified as if it were a very small amount above zero instead.

Alignment As a result of our particular choice of features, a certain amount of within-group alignment, via the PCA method (see §2.2.4) may be used at this stage. It turns out, therefore, that our dataset X, of size (297×25), has 9 distinct presence patterns existent in its rows, and 5 in its columns. This means that the data can be characterised into a 9×5 array of sub-groups. We consider first the 5 separate patterns of features; Table 8.2 shows how groups of features can be considered as being 'similar' in that they describe the same properties of a dataset. Note that the neutron multiplicity is put in the same group as the 'ones', since the latter flags were present for all data. Secondly, we consider the partition of the data itself: Table 8.3 shows how the different groups of data are distributed among these features. In order to pre-process the data, we *performed a linear*

PCA on each feature group in turn, using only those datapoints which had those features present as part of the analysis and kept the *first two* principal components in each case. As PCA was used, and not a whitening procedure, the orthogonality of the transformation was maintained and thus the variances differed according to the actual variance in those directions; this was thought to be appropriate in order to give more importance to those principal components with the highest variance. Thus, our final dataset $\hat{X}$, once all pre-processing had taken place, was of size (297×10), as a result of taking these two components for each of the five groups present.

8.4 Results tables

8.4.1 Categorised by Fissile Element

The following tables, labelled Table 8.5 to 8.10, show the sets of results we have conducted through classification of the processed NDA data, initially by fissile element type. Taking the same form as Table 5.4, we compare our devised methods directly with processes involving the pre-filling of data via three imputation methods, and in all cases compare performance in linear space with that of quadratic kernel space. We have arrived at optimal centres via the 'warm start' methods as described in §6.3 and §6.6 and compared the size of the radii achieved with the solution arrived at through standard methods.

Table 8.5: NDA results: Fissile element 1: Cf-252

	Linear kernel	Quadratic kernel
Hard, Imp 0	6.1135	31.2656
Hard, Imp μ	6.1135	31.2436
Hard, Imp NN	6.1135	31.2357
Hard, XC	6.1100	33.3050
Hard, DO	6.1095	31.2356
Hard, PSJO	6.1095	31.2251
Soft, Imp 0	3.7723	18.3113
Soft, Imp μ	3.7190	18.0375
Soft, Imp NN	4.2173	19.4690
Soft, XC	2.7454	16.9038
PSJO iterations	38	90

Table 8.6: NDA results: Fissile element 2: Uranium

	Linear kernel	Quadratic kernel
Hard, Imp 0	5.4887	50.9168
Hard, Imp μ	5.3995	50.9168
Hard, Imp NN	5.4115	50.9168
Hard, XC	5.3995	52.4335
Hard, DO	5.3995	50.9168
Hard, PSJO	5.3995	50.9168
Soft, Imp 0	1.6412	9.1676
Soft, Imp μ	0.8887	7.5507
Soft, Imp NN	1.2879	10.3296
Soft, XC	0.8319	5.8288
PSJO iterations	82	36

Table 8.7: NDA results: Fissile element 3: Plutonium

	Linear kernel	Quadratic kernel
Hard, Imp 0	4.0994	21.5392
Hard, Imp μ	4.0994	21.5566
Hard, Imp NN	4.0994	21.5601
Hard, XC	4.0994	23.6893
Hard, DO	4.0994	21.5392
Hard, PSJO	4.0994	21.5310
Soft, Imp 0	2.9210	12.5147
Soft, Imp μ	2.8996	12.5282
Soft, Imp NN	2.9017	12.5607
Soft, XC	1.0130	9.4269
PSJO iterations	22	29

Table 8.8: NDA results: Fissile element 4: Ba-133

	Linear kernel	Quadratic kernel
Hard, Imp 0	2.9482	14.5697
Hard, Imp μ	2.6633	14.3735
Hard, Imp NN	2.6633	14.3735
Hard, XC	2.6633	15.3242
Hard, DO	2.5587	14.1981
Hard, PSJO	2.5923	14.1419
Soft, Imp 0	1.7145	7.7913
Soft, Imp μ	1.6621	8.0464
Soft, Imp NN	1.6980	8.3258
Soft, XC	0.5634	1.3318
PSJO iterations	211	111

Table 8.9: NDA results: Fissile element 5: Co-60

	Linear kernel	Quadratic kernel
Hard, Imp 0	3.4087	11.2224
Hard, Imp μ	3.3495	11.4088
Hard, Imp NN	3.3534	11.4072
Hard, XC	3.3490	11.3428
Hard, DO	3.3490	11.1860
Hard, PSJO	3.3651	11.1886
Soft, Imp 0	1.6682	8.1721
Soft, Imp μ	1.6254	8.4695
Soft, Imp NN	1.6123	8.3533
Soft, XC	0.4718	3.0885
PSJO iterations	122	138

Table 8.10: NDA results: Fissile element 6: Cs-137

	Linear kernel	Quadratic kernel
Hard, Imp 0	3.5531	16.8503
Hard, Imp μ	3.5221	16.8180
Hard, Imp NN	3.5350	16.9787
Hard, XC	3.5221	17.4725
Hard, DO	3.4936	16.3896
Hard, PSJO	3.4936	16.2072
Soft, Imp 0	1.5152	4.1718
Soft, Imp μ	1.3890	4.5136
Soft, Imp NN	1.8601	5.6570
Soft, XC	0.7218	1.5844
PSJO iterations	64	94

Discussion These tables show that in the hard margin case, the exact-centre method and the dual-optimisation method can be shown to have comparable and sometimes superior results to using imputation alone, if we take as our measure of goodness the tightness of the bound given for the cluster of data, and therefore the smaller values of the radius R. Care should be taken, however, when interpreting the soft margin results, as these often show an unrealistically high improvement in the enclosing radius of the dataset when the exact-centre method is used over any imputation method. There is a simple reason for this: the imputation methods are forced to use all dimensions of the dataset and fill in as required, whereas the exact-centre method, on knowing that half of the datapoints in the soft-margin case may be discarded, simply ignores those with full data. This does not mean the above results are useless, however; it is merely an indication that in this case, the missingness of the NDA data is not exactly structural. It is true that where there is missing data, the experiments themselves were not carried out, as opposed to there being some underlying value which the experimental team failed to measure properly; however, this is not the same as the case of pure structural missingness, as it is reasonable here to say that *had the missing experiment taken place*, there would be a commensurate reading to provide features in this case. In the case of true structural missingness, the very fact that data was not there would itself be a very significant part of the makeup of the dataset, and thus in this case, these results would be more meaningful. Arguably, however, it is still important to note that the above results do show that the exact-centre classification machine works particularly well given the requirement of a soft margin, and the results still hold true, in that the geometric distances from the full-data centre of the data which was used by the classifier in these seemingly unrealistic cases, were all within a

bound far smaller than that to be found through the imputation methods discussed.

8.4.2 Cross validation on Fissile Element

In this section, we briefly outline the results from our cross validation study to show that under the benchmark of overall classification performance, our method can also compare favourably with imputation. To this end, we use the results gleaned from the particle swarm optimisation tests as they refer directly to an optimal completion of the dataset, without further analysis being required to recover this as with the case of the exact-centre and dual-optimisation methods. As before, each class of data (i.e. points with a given nature of fissile element) was taken as a separate entity and training and testing sets were taken stochastically from the entire dataset, whereupon the optimal completion was used that resulted from the *same SVDD* that provided the results for §8.4.1. In the case of Class 1, as it is 10-dimensional compared to the 6-dimensional other classes, the final four dimensions were discarded, thus making the following results an underestimation of the performance. As noted in §6.5, we consistently take 20% of the points out for testing and repeat the process over 200 iterations. As the full dataset contains 297 points, this makes for a training set of 238 points (whose class constitution was allowed to naturally differ from test to test depending on the points selected), and a testing set of 59 points for each test. Thus, results show over $59 \times 200 = 11800$ points the total numbers classified correctly and wrongly from each class. In each of the following tables, the rows denote the predicted class values and the columns the actual nature of the data. We include the reject class set at a probability density of $(0.01)^d$, where in the linear case $d = 6$ and in the quadratic case $d = 28$.

Table 8.11: CV study: Zero imputation, Linear kernel

1	**1343**	424	886	1045	1334	637
2	161	**1595**	42	283	5	172
3	0	0	**1422**	0	0	0
4	52	568	5	**102**	8	107
5	130	32	13	0	**183**	836
6	21	121	13	152	11	**78**
Rej.	0	19	0	0	0	0

Table 8.12: CV study: Zero imputation, Quadratic kernel

1	**1035**	1181	0	271	0	97
2	74	**361**	79	61	38	68
3	43	0	**1488**	0	0	23
4	165	687	58	**126**	27	241
5	84	0	0	0	**1242**	528
6	384	521	707	1103	205	**862**
Rej.	0	41	0	0	0	0

Table 8.13: CV study: Mean imputation, Linear kernel

1	**1443**	487	846	896	1447	893
2	180	**1715**	31	322	3	190
3	0	1	**1474**	15	0	5
4	29	306	8	**130**	8	70
5	71	38	3	65	**62**	556
6	31	170	40	131	11	**80**
Rej.	0	43	0	0	0	0

Table 8.14: CV study: Mean imputation, Quadratic kernel

1	**1099**	1412	11	290	27	169
2	91	**301**	94	48	74	74
3	38	3	**1548**	6	1	61
4	76	426	29	**183**	15	127
5	72	1	0	1	**1288**	605
6	384	557	656	1064	165	**775**
Rej.	0	29	0	0	0	0

Table 8.15: CV study: NN imputation, Linear kernel

1	**1424**	107	739	788	1184	593
2	195	**1867**	25	282	23	184
3	2	4	**1442**	6	2	7
4	52	574	19	**181**	10	175
5	132	33	33	118	**290**	752
6	33	111	28	192	24	**91**
Rej.	2	61	0	0	0	15

Table 8.16: CV study: NN imputation, Quadratic kernel

1	**1159**	1471	6	248	37	156
2	106	**433**	124	95	124	118
3	77	8	**1683**	4	12	185
4	125	586	61	**315**	30	215
5	138	22	0	1	**1281**	772
6	161	243	420	851	95	**396**
Rej.	0	36	0	0	0	6

Table 8.17: CV study: PSJO method, Linear kernel

1	**1579**	371	861	760	1463	696
2	154	**1977**	32	332	0	180
3	0	0	**1304**	3	0	0
4	1	385	31	**232**	10	119
5	42	0	48	0	**49**	690
6	36	69	75	198	11	**89**
Rej.	0	3	0	0	0	0

Table 8.18: CV study: PSJO method, Quadratic kernel

1	**1242**	1562	101	318	295	130
2	78	**387**	107	56	49	273
3	89	13	**1654**	10	0	79
4	80	563	77	**690**	38	336
5	97	0	0	0	**1088**	655
6	186	164	382	440	103	**414**
Rej.	0	44	0	0	0	0

Table 8.19: CV study: Method comparison, total correctly predicted

	Imp.0	Imp.μ	Imp.NN	PSJO
Lin.	4723	4904	**5295**	5230
Quad.	5114	5194	5267	**5475**

8.4.3 Categorised by Shielding Method

In addition to the above study, a shielding method domain description machine was also built, results of which are in Tables 8.20, 8.21, 8.22, 8.23, 8.24 and 8.25 in this section, with each table pertaining to the analysis of one fissile element. In these tables, for each element in turn, we perform a domain description classification task **with a hard margin**, for each shielding method contained within. This is repeated for the shielding methods numbered 0, 1, 2, 6 and 7 from above: these correspond to Bare, Aluminium, Steel, Lead and CHON shieldings respectively. Having the entire suite of hard-margin results present for all methods and element/shielding combinations means that conclusions may be drawn on the patterns which emerge. Although soft-margin results analogous to those performed in the 'usual' tests above were collected, it was not thought necessary to include them in analysis as their meaning would not be very pertinent, such is the small sample size of each particular combination: sometimes there will be just 8 datapoints in each set, as described in Table 8.4 above.

Table 8.20: Shielding study: Radii for Element 1

Shield	Kernel	Imp 0	Imp μ	Imp NN	XC	DO
0	Lin	3.4686	2.7956	2.8045	2.7956	2.7955
1	Lin	2.9181	2.9181	2.9181	2.9181	2.9181
2	Lin	1.5248	1.2515	1.2515	1.2515	1.2516
6	Lin	2.9637	2.9090	2.9090	2.9090	2.9090
7	Lin	2.2000	1.9449	1.9777	1.9422	1.9402
0	Quad	17.0310	19.1062	19.2031	19.2217	16.9367
1	Quad	19.6761	19.6761	19.6761	21.2543	19.6761
2	Quad	9.6191	8.2597	8.2597	8.3557	8.2597
6	Quad	20.3526	20.1868	20.1868	20.2669	20.1868
7	Quad	6.7867	7.0448	7.4152	7.1021	6.7077

Table 8.21: Shielding study: Radii for Element 2

Shield	Kernel	Imp 0	Imp μ	Imp NN	XC	DO
0	Lin	1.0987	1.0946	1.0946	1.0946	1.0946
1	Lin	5.1671	5.1671	5.1671	5.1671	5.1671
2	Lin	0.525	0.525	0.525	0.525	0.525
6	Lin	2.7501	2.7501	2.7501	2.7501	2.7501
7	Lin	1.7574	1.0958	1.0969	1.0958	1.0943
0	Quad	4.1239	4.1111	4.1111	4.2338	4.1111
1	Quad	47.8537	47.8537	47.8537	48.0793	47.8537
2	Quad	2.8614	2.8614	2.8614	2.8634	2.8614
6	Quad	17.4315	17.4315	17.4315	18.1869	17.4315
7	Quad	6.6775	4.9068	4.9846	4.9551	4.9182

Table 8.22: Shielding study: Radii for Element 3

Shield	Kernel	Imp 0	Imp μ	Imp NN	XC	DO
0	Lin	0.9312	0.4595	0.4595	0.4595	0.4595
1	Lin	2.1647	2.1647	2.1647	2.1647	2.1648
2	Lin	0.8323	0.4899	0.4899	0.4899	0.4899
6	Lin	1.3343	1.1979	1.1979	1.1979	1.1979
7	Lin	2.0321	2.0321	2.0321	2.0321	2.0321
0	Quad	2.277	1.4559	1.4559	1.4601	1.4559
1	Quad	8.7729	8.7729	8.7729	8.9248	8.7729
2	Quad	1.9331	1.3595	1.3595	1.3672	1.3595
6	Quad	2.7433	3.1537	3.1537	3.1821	2.7424
7	Quad	4.6487	4.6169	4.6169	4.9933	4.6169

Table 8.23: Shielding study: Radii for Element 4

Shield	Kernel	Imp 0	Imp μ	Imp NN	XC	DO
0	Lin	0.7305	0.1469	0.1469	0.1469	0.1473
1	Lin	2.2083	2.1566	2.1566	2.1566	2.1566
2	Lin	1.2942	1.1351	1.1351	1.1351	1.1351
6	Lin	0.7483	0.3872	0.3872	0.3872	0.3872
7	Lin	0.5637	0.2893	0.2893	0.2893	0.2896
0	Quad	1.7300	0.4122	0.4122	0.4106	0.4170
1	Quad	10.4382	10.3936	10.3936	10.6071	10.3936
2	Quad	4.3591	4.0918	4.0918	4.1445	4.0918
6	Quad	3.4118	2.3247	2.3247	2.2563	2.3230
7	Quad	1.2161	0.7134	0.7134	0.7162	0.7134

Table 8.24: Shielding study: Radii for Element 5

Shield	Kernel	Imp 0	Imp μ	Imp NN	XC	DO
0	Lin	2.2311	2.1570	2.1570	2.1570	2.1570
1	Lin	2.0861	1.9423	1.9423	1.9423	1.9423
2	Lin	2.6584	2.5962	2.5962	2.5962	2.5962
6	Lin	3.2832	3.2358	3.2358	3.2358	3.2358
7	Lin	1.9986	1.8906	1.8906	1.8906	1.8906
0	Quad	11.0464	11.0092	11.0092	11.2344	11.0017
1	Quad	10.9149	10.6021	10.6021	10.7534	10.6018
2	Quad	11.0592	11.3380	11.3380	11.5530	11.0567
6	Quad	10.6106	10.8323	10.8323	11.0461	10.5980
7	Quad	10.4586	10.2295	10.2295	10.3287	10.2585

Table 8.25: Shielding study: Radii for Element 6

Shield	Kernel	Imp 0	Imp μ	Imp NN	XC	DO
0	Lin	1.5768	1.3945	1.3945	1.3945	1.3945
1	Lin	2.2386	2.0929	2.0929	2.0929	2.0929
2	Lin	2.3555	2.3350	2.3350	2.3219	2.3203
6	Lin	1.6997	1.6338	1.6338	1.6338	1.6338
7	Lin	3.1375	3.1375	3.1375	3.1375	3.1375
0	Quad	2.8832	3.5453	3.5453	3.6203	2.8832
1	Quad	7.5913	7.5913	7.5913	7.8601	7.5913
2	Quad	6.8881	7.0075	7.0075	8.5692	6.8662
6	Quad	5.2915	5.6310	5.6310	5.5091	5.2904
7	Quad	15.2593	15.2404	15.2633	15.4751	15.2387

Table 8.26: Shielding study: Minimal radii, linear kernel

	Sh 0	Sh 1	Sh 2	Sh 6	Sh 7
Fis 1	2.7955	2.9181	1.2515	2.9090	1.9402
Fis 2	1.0946	5.1671	0.5250	2.7501	1.0943
Fis 3	0.4595	2.1647	0.4899	1.1979	2.0321
Fis 4	0.1469	2.1566	1.1351	0.3872	0.2893
Fis 5	2.1570	1.9423	2.5962	3.2358	1.8906
Fis 6	1.3945	2.0929	2.3203	1.6338	3.1375

Table 8.27: Shielding study: Minimal radii, quadratic kernel

	Sh 0	Sh 1	Sh 2	Sh 6	Sh 7
Fis 1	16.9367	19.6761	8.2597	20.1868	6.7077
Fis 2	4.1111	47.8537	2.8614	17.4315	4.9068
Fis 3	1.4559	8.7729	1.3595	2.7424	4.6169
Fis 4	0.4106	10.3936	4.0918	2.2563	0.7134
Fis 5	11.0017	10.6018	11.0567	10.5980	10.2295
Fis 6	2.8832	7.5913	6.8662	5.2904	15.2387

Discussion Plainly, with this set of results, it is more difficult to interpret the conclusions which can be drawn. With the study relating to fissile element given in §8.4.1, the main purpose was to see in which cases smaller radii could be achieved when using the methods we developed in comparison to imputation pre-processing, and thus results were simple to arrive at; it is in some cases very clear where these methods can excel, and as long as the caveats given in the 'Discussion' section are noted, it is clear that our results could be of some use to datasets such as these. However, when performing the shielding method analysis, it is noted that the efficacy of our methods has already been shown and we are thus merely performing an empirical study into what can be said of the dataset as given. To make sense of these results, we interpret them as follows: A shielding method is essentially a barrier against which meaningful analysis may be made. In the case that a shielding method is overpoweringly effective, it will be the case that no meaningful information can penetrate, thus making classification of the material's properties almost impossible. We shall interpret **smaller radii** in the case of each fissile/shielding combination to imply that the cluster of data was compact in feature space, and thus that it was possible to process the necessary information for classification, thus resulting in a lower volume of hypersphere, which itself (when combined with a hybrid computational classification method) would lead to more distinctive clusters and therefore better classifier performance. Accordingly, the first observation we make, consulting Tables 8.26 and 8.27, is that as expected, the 'Bare' results in most cases produce smaller class-specific radii than others, meaning greater classification accuracy. This result is encouraging, as it shows that shielding is generally causing the sub-classes to be more widespread, and thus have a greater chance of overlap.

Chapter 9

Conclusions

This project has achieved the following aim:

> We have created a domain-descriptive utility to calculate the minimal enclosing sphere of any given dataset in the presence of missing data, under no assumptions about any models governing its distribution. Throughout, we have avoided the uses of imputation, and shown that our method can be effective in certain types of kernel space, with soft and hard margins, and works best where the nature of missing data is structural. Thus, we have extended existing work relating to three fields of theory: the SVDD classification machine, the instance-specific SVM method of Chechik et al, and that of dealing with missing data. We have also applied our methods to the provided NDA dataset, and shown them to be effective in terms of classifying by fissile material and in a shielding study.

In this section, we will review the methods we have introduced and the results they were produced through a more general view. We shall interpret the results

gleaned from Chapters 5, 6 and 8, and consider the facts that are shown by these. Furthermore, we shall discuss the various ways in which this project could be improved, and the ways it would be possible to take it forward given a wider remit.

9.1 Interpretation of results

9.1.1 Synthetic datasets

In the above technical chapters, we provided a brief discussion as to the areas in which our results on synthetic datasets proved to be satisfying and those where our methods fell short. We note again, before continuing any further that *no method is perfect*, and that the fact that in some cases our methods failed to produce better results than the use of imputation methods does not necessarily make them ineffective. Indeed, in many cases it was shown that a marked improvement can be made on the margins involved with imputing data, and as long as this information is interpreted in the correct manner, these results in particular show that our methods can be made effective in some cases. It is also important to note, as we did in §5.5, that in the case of the preliminary tests in this chapter, the randomness mechanism was entirely arbitrary: we made data absent by applying an MCAR (missing completely at random) algorithm, and thus it is perhaps unreasonable to expect a method honed to the analysis of *structurally* missing data to perform significantly better when the inherent nature of the missingness in the dataset it is presented with is not even structural. There is a conceptual difference here between:

- data that is missing in such a way that we may write (not measured); and
- data that is missing *because* it is not applicable.

An obvious question that may be asked, where the methods do show a significant overall improvement (for example, in parts of the full structural analysis given in §5.6 and §6.7), is the following: just because our methods are apparently achieving smaller descriptive radii for a dataset, does that necessarily mean they are overall better classification methods? The answer is clearly no: consider, for example, a Parzen window classifier with a kernel function of infinitesimal variance, so that the classifier produces an incredible density of probability over the immediate neighbourhood of each training datapoint, and virtually zero elsewhere. Thus, the 'enclosing sphere' radii of each datapoint would be virtually zero, and yet this is clearly not a good method of classification. Arguably, however, this is not what we set out to achieve in the first place at all. A full study of the various hybrid methods which could then be used to perform a cross-validated, sensitivity/specificity analysis on all the data given here is arguably unnecessary. We have shown that a *stable* classification process can be built which uses all the particle information present and does not use any information that is not present, thus achieving the aim of an analysis machine designed to work with structurally missing data as given in §3.1.5. It is not intended to be a catch-all procedure for handling all types of data, but we showed above the various situations in which our methods can give good pointers as to what a reasonable description of a dataset could be without the need for use of the pre-imputation procedure. Ultimately, our remit was a divergent one, in that we sought to improve two fields at once: firstly, we extend Chechik et al's plane-based binary classification process in the presence of structurally missing data to domain description, which is a method commonly used

for multi-class classification; secondly, we refine and improve the way their paper dealt with kernel spaces to emerge with a method that deals with these concepts more rigorously. Evidence that at least in some cases, we can show that a smaller descriptive radius can be provided for a dataset in both linear and quadratic cases, with both of the methods we have developed, shows these methods to be of worth at least in terms of being able to ascertain whether the visualisation of a dataset as having non-parametric structural missingness is a good one. We certainly do not intend the interpretations of our results to be a single-minded refutation of the tried and tested imputation methods that are sometimes incredibly effective in the useful completion of datasets; indeed, this is why in Chapter 3 we were at such pains to give a careful, thorough analysis of common parametric methods, as some or all of these could equally be perfectly fine for use on a dataset where some of the assumptions we listed in this chapter are valid. One of the main advantages of our methods, when they are shown to be effective, is that they make no assumptions whatsoever about the underlying distribution of the data, and they should therefore be seen as an aid to a proper analysis where the exact nature of a dataset is perhaps not clear for whichever reason.

9.1.2 Provided datasets

In §8.4, we provided results from two disjoint experiments performed on the dataset provided by the external company, which we subsequently processed to extract some meaningful features. After showing in the previous chapter, and with the abovementioned tables, that in some situations our methods can prove effective, these results were included as applications of this method to a 'real-life' category of data to assess how well they could perform. In both tasks – that of

classifying the different fissile elements in §8.4.1 and the shielding method study in §8.4.3 – we observe that the total amount of information given to the classifier is actually relatively small. Three major points may be raised of this study:

- Firstly, by nature of how the dataset was ultimately organised, it could easily be argued that the type of missingness dealt with in this set is not entirely structural.

- Secondly, the process by which the features were originally extracted was largely heuristic and fairly crude.

- Thirdly, in order to reduce the enormous problem of a black-box classifier down to one which is manageable via standard methods, a lot of information had to be discarded from the raw consignment of data before any serious analysis could take place.

These points notwithstanding, numerically the results show that our methods in many cases performed well on the NDA dataset, and it should certainly not be assumed therefore that the methods should *only* work where the nature of the missingness of data is exclusively structural. We have therefore proven that it is possible to classify incomplete 'real' experimental data in a way which is not tantamount to 'guessing' what the results of the experiments were that were not carried out. This in itself is an advantage, as the simulation of real experimental data may itself be a complex issue, and contrapositively the creation of features extracted from presumed experimental data would require knowledge of correct statistical distributions: where the feature extraction algorithms and the process of producing the raw data is as complex as it is in this example, this step alone is nontrivial. Thus, it could be seen that the major factor by which our method

can improve understanding of generic incomplete experimental data such as this, is precisely that it avoids the need for simulation. It is still true that this dataset could still be analysed, with perhaps impressive results, using any non-parametric completion procedure, but these methods will always force a degree of subjectivity or assumption, whereas potentially our methods achieve the same aim without needing to take these sometimes arbitrary 'guesses'. Naturally, this is all subject to the same caveats as given in the previous section: namely, that the presence of a smaller radius in comparison with imputation methods, etc. will not always guarantee that mathematically the solution is superior, but in the cases where the end-user is forced to decide between a parametric or non-parametric analysis, these methods may well be seen to have worth.

9.2 Evaluation of project

9.2.1 Appraisal of methods

It is always important, in a conclusive appraisal, to see what any new method developed for a particular purpose can achieve, along with its limitations. Although in itself our method describes an entirely self-contained system for computation of an optimal domain-descriptive margin around a dataset in the presence of (structurally) missing data, it should be seen simultaneously as a first attempt at a more general theory. At present, both methods have heavy reliance on the complex machinery involved in the optimisation algorithms necessary for their execution. This is a subject we only briefly touched upon when describing these methods, as the practical application of commands such as `fminunc` was seen as more important than the design of the algorithms themselves. It is important to

note, however, that the heavy-duty use of these algorithms makes the convergence process very slow when compared to the simple application of a system such as SVDD after an imputation method has been applied. Both the exact-centre and the dual optimisation methods use two loops of an optimisation process to converge on a best-fit solution: the inner process regarding the optimal completion of a dataset *given* a parameter, and the outer process being the optimisation of these parameters themselves. Thus, one major criticism of these methods would simply relate to their inefficiency in terms of computational time. This is less of a problem than may be initially thought, however: this thesis has merely served to show such a thing is possible, rather than to say that this is an appropriate method to use in all cases. It is hoped that further studies into this area may take this into account, and build on the structures we have introduced here to produce a system of greater elegance to handle similar data-driven problems in a rigorous, non-parametric fashion. Moreover, our methods have essentially the same form, and thus computational complexity, as the statistical sampling procedure known as randomised maximum likelihood (RML) [59] – and should therefore be dismissed as too inefficient only if other methods based on a similar looping procedure are also to be thought of as being of little significance.

It should also be noted that these methods depend, from the outset, on kernels based on the inner products of datapoints, and thus we have not covered the case of the popular radial basis function (RBF) kernel. Various logistical problems would arise from the use of this kernel due to its property of mapping every point to an infinite-dimensional hypersphere of unit radius; as a result, these methods fail when applied to this kernel. Clearly, some adaptation would be necessary to bring RBF kernels into the remit of these methods. Furthermore, although we showed

in §5.5 that the effectiveness of our exact-centre method is somewhat inversely proportional to the degree of polynomial kernel function, it should be noted again that the square-root and logarithm-based kernels used in this section are not valid in a Mercer sense, and thus should not be seen as valid for a serious analysis in the full-data case: there is no implied mapping function which they could be said to be equivalent to. On a side-note, however, the application of our kernel-based methods to structurally missing data ties them in with relatively new concepts in the artificial intelligence field, and will hopefully provide a springboard on which further analysis can be made marrying these two fields.

The method we have referred to as the 'dual optimisation' method has some obvious areas in which improvement could be of use. Most pertinently, and arguably ironically, its very performance relies on a method of pre-imputation to create a master dataset we have referred to as X^*, which itself both violates the principles of structurally missing data and has the unfortunate side-effect of loosening the inherent ties between the coefficient weighting vector α and the dataset itself (as discussed in the relevant sections). Further studies may serve to plug this inconsistency, effectively introduced as a temporary measure to provide a halfway house between the exact-centre method on input space, and the intractability of a full working in feature space. If this could be resolved, and the ties between X^* and α reinstated, the optimality constraints on α could therefore also be brought back such that a more analogous analysis relating to the computation of the full-data case SVDD result could be derived. This would have far-reaching implications for this method in terms of its ability to deal with soft margins, which we did not cover above in the dual-optimisation method's analysis, due to the lack of any efficient algorithms to deal with a 'relaxed minimax' problem: that is, min-

imising the k^{th} largest value of a set of objectives. If the definition of α were to be reattached like this, we could demand again something similar to:

$$\forall i, \alpha_i \in [0, C], \sum_{i=1}^{N} \alpha_i \equiv 1, \tag{9.1}$$

thus providing a far better framework with which to analyse soft margins using this more versatile method.

Finally, we include some comments about how our methods can be affected by large proportions of missingness in a dataset. For our applied dataset, generally the proportion of absent features was between 20-30%, and we found that in this case our methods could function properly and produce interesting results. However, care should be exercised when applying these methods to datasets with a very high (50% +) proportion of missing values. In theory, there is no reason our methods would not be able to function under these conditions, but the results they may produce may be of doubtful use, in a similar way to the results we obtained when leaving half of the datapoints out of an enclosing sphere using the exact-centre method: the resulting sphere sizes were abnormally small, and their interpretation in terms of the structure of the datasets on which they were operating is unclear.

9.2.2 Further work

With the above criticisms in mind, we may identify further areas in which a study could be taken forward relating to this project. Most significantly are the points raised above: the heavy reliance on optimisation procedures could be made more rigorous by using further mathematical methods on the theory of structurally missing data: it is a convincing argument that these methods could be made far more efficient should an artificially-intelligent machine be able to come up with, say,

an analytic relation between kernel type and the best completion to use when analysing two datapoints with missing data. This is an interesting area and a natural extension of this project, which would yield both efficiency and versatility in terms of the mathematics behind why it is effective. Secondly, further insight into the exact nature of whether data is structurally missing or not would perhaps bear fruit: this area has not been extensively covered in literature, as this type of data is arguably relatively rare in terms of the examples of datasets available publicly. If this method should be a pointer towards whether a non-parametric method should be assuming anything about the nature of absent data, would it make sense for another process to pre-determine if the structural absence of data, based on the present values and any known models, is even a statistically reasonable assumption to make? Thirdly, the prevalence of the radial basis function cannot be ignored, and thus any extension to this work would need to consider a form where this type of kernel could be effectively studied, as instance-specific geometric kernel distances in RBF feature space are still reasonable quantities to query. Fourthly, as discussed above, the issues relating to the decoupling of the significance of the vector α when relating to a dataset would be another obvious direction in which to perform further work, and may indeed be the most fruitful should it lead to a more elegant handling of the soft-margin problem, possibly based on the same principles as those used for the joint optimisation technique we included for the applied dataset. Finally, a deeper analysis of the relationship between the structure of a dataset and the conditional behaviour of these algorithms would be of use, as at present we have not really touched on issues regarding the shape of the landscape that results from the optimisation-based measures of distance. Further analysis of the structure of this would help analysis into the most

appropriate optimisation methods and assist in making this system more efficient. Related to this is a full study of what would happen to the optimisation landscape if a large amount of the data were missing; clearly, in this case, it is more important to demand that the dataset, despite being heavily incomplete, would still in some sense exhibit structural missingness, as if this were not the case, the resultant outputs of sphere radii would be almost meaningless. Therefore, future work on this area would involve an in-depth analysis of how to construct a large, heavily structurally incomplete dataset in a rigorous and meaningful manner, and investigate how our methods would react to this scenario.

9.2.3 Concluding remarks

The field of classification is almost peerless when considered in terms of generality of application, variety of analytical methods and variance of the complexity of algorithms for imparting the required artificial intelligence: for every application, inevitably some methods will work perfectly, others badly. Neither will it ever be seen as unnecessary; the proliferation of AI methods in all areas of modern life ensures classification will never cease to find diverse uses. As we showed at the start, perhaps it is yet too soon to hope that the artificial intelligence of computers in pattern recognition will challenge the astounding intuitive capabilities of the human mind in many significant areas – but to assume any class of problems will forever be unsuitable for machine intelligence methods is both ignorant and incorrect, as is the assertion that we can ever have completed our studies in classification. In this field, it would seem, the computational scientist's work is never done.

Appendix A

NDA data and supporting notes

The following information has been adapted from supporting material provided by the AWE.

A.1 Introduction

The first set, consisting of experiments conducted between January – March 2006, was delivered for analysis in the autumn of 2006; after further discussion, this dataset was augmented by a second set, consisting of experiments carried out between November 2007 – January 2008. Delivery of this set and its corresponding background measurements was completed in June 2008. There were two main methods of measurement seen as useful for analysis: spectroscopy data and neutron array data. Both are described below; a more in-depth discussion of radiation analysis may be found in the book by Knoll [41].

Data exists describing emissions of various radioactive materials when allowed to decay. For each of these, experiments were conducted with various

shielding materials (aluminium, steel, lead, and an organic compound known as CHON), and over different time scales ranging from 5 minutes to 4 hours. Bare readings were also taken, as were background readings, and in most cases four separate detectors were used for each experiment: a sodium iodide (NaI) scintillator spectrometer, a high-purity germanium (HPGe) semiconductor spectrometer, a neutron array and a gamma camera.

Coding system In Tables A.1, A.2 and A.3 which follow, a coding system is used to designate each experiment by fissile material, shielding method and timescale of the experiment. Blocks of experiments are coded by *set number* to designate the order in which they were received, and the dates are also recorded. The individual sets are covered below; the following universal code is used for the experimental properties. Time codings are arranged thus:

$$t_i = \left\{ \begin{array}{ll} 0: & T = 5 \text{ mins} \\ 1: & T = 10 \text{ mins} \\ 2: & T = 15 \text{ mins} \\ 3: & T = 20 \text{ mins} \\ 4: & T = 30 \text{ mins} \\ 5: & T = 60 \text{ mins} \\ 6: & T = 120 \text{ mins} \\ 7: & T = 240 \text{ mins} \end{array} \right\}.$$

Fissile element codings are arranged:

$$f_i = \left\{ \begin{array}{ll} 0: & \text{background} \\ 1: & \text{Californium-252} \\ 2: & \text{Uranium} \\ 3: & \text{Plutonium} \\ 4: & \text{Barium-133} \\ 5: & \text{Cobalt-60} \\ 6: & \text{Caesium-137} \\ 7: & \text{Large Barium source} \\ 8: & \text{Mixed sources} \end{array} \right\}.$$

Finally, for the shielding methods we have:

$$s_i = \left\{ \begin{array}{ll} 0: & \text{bare reading} \\ 1: & \text{aluminium} \\ 2: & \text{steel ball} \\ 3: & \text{steel planar to } \gamma \text{ detector} \\ 4: & \text{steel planar to N detector} \\ 5: & \text{lead ball} \\ 6: & \text{lead ball with GI} \\ 7: & \text{CHON} \end{array} \right\}.$$

These shielding designations only apply to those experiments where $f_i \neq 0$, i.e. where there existed a fissile source to analyse; for the background readings, the s_i were instead used to designate the instance (equivalent to date) of that particular background reading. During the classification process, some of these designations (e.g. shielding methods 3 and 4, or 5 and 6) ended up being relaxed in places: see

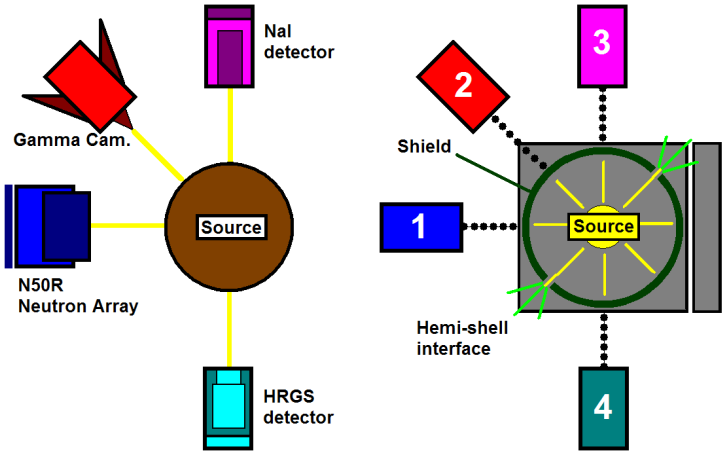
below for a fuller description of this. In the tables below, there is an entry for each date and timescale combination, since usually the fissile element and shielding method were kept constant on a given date. Thus, in each cell, we have denoted an 's' if there was a *S*odium iodide scintillator-based spectroscopy experiment at that timescale on that day; similarly 'g' for *G*ermanium; and 'n' will denote a *N*eutron array experiment, which (as stated above) will generally consist of ten equal arrays of timescale $T/10$.

Exceptions The following facts are stated here as inconsistencies with the completeness of the observed data in some way:

- 11-1-2006: This background reading was unnaturally large and was discarded from the background-cancellation process.
- 6-2-2006: In the 20 minute test, the tenth and final reading from the neutron array was not present.
- 27-2-2006 to 13-3-2006: The spectroscopy experiments performed between these dates were calibrated differently. Background data from 27-2-06 was used for these.
- 7-11-2007 to 8-11-2007: The HRGS experiments were calibrated with unusual attenuation coefficients; the overlap readings from 20-11-2007 were used instead and these were discarded.
- 9-11-2007: In the 10 minute test, only the first seven neutron timeframes were present.

- 28-11-2007: The 4-hour neutron test was run for 4 times as long, making a 16-hour test with 40 'frames'. The first 10 were kept.

A.2 Experimental geometry



(a) Detectors (b) Shield setup

Figure A.1: Experimental geometry

Figures A.1(a) and A.1(b) describe the experimental geometry used during the collection of the data consignment. The nature of these is not too significant in terms of a classification task, but is re-produced here to show that multiple detectors would in general be used in parallel when analysing any particular fissile element and shielding method combination.

date	set	f_i	s_i	0	1	2	3	4	5	6	7
11-1-2006	1	0	1								*sg*
13-1-2006	1	1	0					n			
13-1-2006	1	3	0			n					
16-1-2006	1	3	0	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
17-1-2006	1	3	1	n	n	n	n	n	n	n	n
18-1-2006	1	2	1	sgn	sg	sg	sg	sgn	sgn	sgn	sgn
19-1-2006	1	1	1	sgn	sgn	sgn	sgn	sgn	sgn	sgn	sgn
23-1-2006	1	1	0	sg	sg	sg	sgn	sgn	sg	sg	sg
24-1-2006	1	3	1	sgn	sgn	sgn	sg	sgn	sgn	sgn	sgn
25-1-2006	1	3	2	sgn	sgn	sgn	sg	sgn	sgn	sgn	sgn
26-1-2006	1	1	2	sgn	sgn	sgn	sg	sgn	sgn	sgn	sgn
27-1-2006	1	2	2	sg	sg	sg	sg	sg	sg	sg	sg
1-2-2006	1	2	0	sg	sg	sg	sgn	sgn	sgn	sgn	sgn
2-2-2006	1	2	3	sg	sg	sg	sg	sg	sg	sgn	sg
6-2-2006	1	2	4	sgn	sgn	sgn	*sgn*	sgn	sgn	sg	sg
7-2-2006	1	0	2		n	n	n	n	n	n	sgn
9-2-2006	1	3	5	sgn	sgn	sgn	sgn	sgn	sgn	sgn	sgn
10-2-2006	1	0	3								sg
27-2-2006	1	0	4		s						gn
28-2-2006	1	3	6	sg	sg	sg	sg	sg	sg	sg	sgn
2-3-2006	1	3	5						n	n	
13-3-2006	1	3	7	sg	sg	sg	sg	sg	sg	sgn	sgn

Table A.1: Full description of NDA data, page 1

date	set	f_i	s_i	0	1	2	3	4	5	6	7
7-11-2007	4	0	5	s*g*	s*g*	s*g*	s*g*	s*g*	s*g*	s*g*	s*g*
8-11-2007	4	0	6	s*g*	s*g*	s*g*	s*g*	s*g*	s*g*	s*g*	s*g*n
9-11-2007	3	7	5		s*n*	sn	s	sn	sn	s	sn
20-11-2007	4	0	7	sg	sg	sg	sg	sg	sg	sg	sg
21-11-2007	3	7	5	sgn	sgn	sg	sg	sgn	sgn	sg	sgn
22-11-2007	3	7	7	sg	sg	sg	sg	sgn	sgn		n
23-11-2007	3	4	7		sgn	sg	sg	sgn	sg	sg	sgn
26-11-2007	3	1	7	sg	sg	sg	sg	sgn	sgn	sg	sgn
28-11-2007	4	0	8		n,n				n		*n*
29-11-2007	3	1	7	n	n	n				n	
30-11-2007	3	2	7	s	s	s	s	s	s	s	sn
3-12-2007	3	2	7	n	n	n		n	n		
5-12-2007	3	6	7	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
7-12-2007	3	2	6	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
10-12-2007	3	1	6	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
12-12-2007	3	6	1	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
13-12-2007	3	4	1	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
4-1-2008	3	4	2	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
7-1-2008	3	6	2	n	n	n		n	n		n
8-1-2008	3	6	2	sg	sg	sg	sg	sg	sg	sg	sg
9-1-2008	3	2	7	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn
10-1-2008	3	3	7	sgn	sgn	sgn	sg	sgn	sgn	sg	sgn

Table A.2: Full description of NDA data, page 2

date	set	f_i	s_i	0	1	2	3	4	5	6	7
11-1-2008	3	4	6	sg	sg	sg	sg	sg	sg	sg	sgn
14-1-2008	3	6	6	sg	sg	sg	sg	sg	sg	sg	sgn
15-1-2008	3	5	7	sg	sg	sg	sg	sg	sg	sg	sgn
16-1-2008	3	5	1	sg	sg	sg	sg	sg	sg	sg	sgn
17-1-2008	3	5	2	sg	sg	sg	sg	sg	sg	sg	sgn
18-1-2008	3	5	6	sg	sg	sg	sg	sg	sg	sg	sgn
21-1-2008	3	4	0	sg	sg	sg	sg	sg	sg	sg	sgn
21-1-2008	3	5	0	sg	sg	sg	sg	sg	sgn	sg	
22-1-2008	3	6	0	sg	sg	sg	sg	sg	sg	sg	sgn
22-1-2008	3	8	0	sg	sg	sg	sg	sg	sg		

Table A.3: Full description of NDA data, page 3

A.3 Data structure

As described above, the experimental data we have analysed is due to three detectors that were used in the collection of data from one type of source behind one shielding method at a time, typically describing the experimental diet for one specific day. Generally, all timeframes were covered on any one day. For the neutron data, the data came in the form of typically thirty matrices describing multiplicity data, of which the ten describing the periodic trigger method of data collection were kept, giving rise to a matrix $N \in \mathbb{R}^{k\times 16\times 10}$, where k signifies the highest multiplicity experienced in that test. Background readings were also taken for neutron data, and had the singular purpose of providing background multiplicity information of neutron emissions in the environment (see §8.2.3 for a discussion). For the spectroscopy experiments, however, the data were a little more complex

and had three constituent parts:

- **Spectral readings:** These are the experimental data themselves, with requisite calibration information. They have the form of a vector $\mathbf{s} \in \mathbb{R}^m$, with the entries marked from 0 to $(m-1)$; in the case of an NaI detector we have $m = 4096$ and for HRGS $m = 16384$.

- **Calibration information:** As described above, the 3 coefficients giving the energy information for the bins. The equation given in §7.2.5 translates the bin numbers into energy readings.

- **Background readings:** These were taken at various points during the data gathering process and allow, in the case that their coefficients are the same as the spectral readings, simple background cancellation by subtraction.

The timeframe, as described above, will also be known, giving a simple way to review the counts per second in any given experiment.

A.4 Received consignments

The following detail the various consignments of raw data which were received, in the order in which they were sent.

Set 1 was received August 2006, and provides data for the experiments conducted in the time period January – March 2006. These were supplemented with *gamma camera* imaging data; it was later decided that this did not produce enough useful information for this particular study.

Set 2 was received July 2007, and showed different types of spectroscopy data which could also come up in a study with a wider remit: plastic scintillator data was provided to show that in this case, very little information may be conveyed. A PIDIE set of Plutonium readings was also produced for our comparison.

Set 3 was received March 2008 and was the major completion of the experimental part of this project. It described the experiments for November 2007 – January 2008, and served the purpose of 'completing the set' in terms of the fissile element and shielding methods for neutron and spectroscopy data. This is shown in Table A.4 below. After this large consignment of data was received, only certain background readings were absent; it was, however, possible from here to provide full analyses based on the entire consignment.

Set 4 described the missing backgrounds which were required for full completeness; these were received June 2008.

Table A.4: Summary of received data

	1 Cf_{252}	**2** HEU	**3** Plut	**4** Ba_{133}	**5** Co_{60}	**6** Cs_{137}	**7** BaLge	**8** Mixed
0 Bare	1	1	1	**3**	**3**	**3**		**3**
1 Alum	1	1	1	**3**	**3**	**3**		
2 Steel	1	1	1	**3**	**3**	**3**		
3 Lead	**3**	**3**	1	**3**	**3**	**3**	**3**	
4 CHON	**3**	**3**	1	**3**	**3**	**3**	**3**	

Further information We also include some technical information here about the assigned dataset, gleaned from liaison sessions with the team responsible for the data collection:

- Elements generally combine linearly with regards to spectroscopy data, should there be more than one element present in a data collection set.
- The exact definition between the shielding methods given is not too significant; merely the knowledge that they are different should suffice.
- The nature of the background readings is not significant.
- Backgrounds from different dates will generally be continuously transitional.
- Calibration coefficients come from simple curve-fitting on known peaks.

Bibliography

[1] Paul D. Allison. *Missing data*. Sage university papers. Quantitative applications in the social sciences ; no. 07-136. Sage, Thousand Oaks, Calif., 2002.

[2] Paul A. Armknecht and Fenella Maitland-Smith. *Price imputation and other techniques for dealing with missing observations, seasonality and quality change in price indices*. IMF working paper ; WP/99/78. International Monetary Fund, Statistics Department, Washington, D.C., 1999.

[3] A. Asunción and D.J. Newman. UCI machine learning repository, 2007.

[4] G. A. Babich and O. I. Camps. Weighted Parzen windows for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):567–570, 1996.

[5] John. Banasik and Jonathan N. Crook. *Reject inference, augmentation, and sample selection*. Working paper series ; no. 05/04. Credit Research Centre, University of Edinburgh,, 2005.

[6] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[7] A. Berg. Shape matching and object recognition using low distortion correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Citeseer, 2005.

[8] C. Bhattacharyya. A second order cone programming formulation for classifying missing data. In *Advances in neural information processing systems 17: proceedings of the 2004 conference*, page 153. The MIT Press, 2005.

[9] J. Bi and T. Zhang. Support vector classification with input data uncertainty. *Advances in Neural Information Processing Systems*, 17:161–168, 2004.

[10] Sean Borman. The Expectation Maximization algorithm, a short tutorial, July 2004. Updated 9th January 2009.

[11] Encyclopaedia Britannica. Ockham's razor (philosophy), 2009. [Online; accessed 7th December 2009].

[12] James Carpenter. Missing at Random (MAR), March 2005. Retrieved 19th November 2009. Web page resource.

[13] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[14] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel, and Daphne Koller. Max-margin classification of data with absent features. *J. Mach. Learn. Res.*, 9:1–21, 2008.

[15] B. Chen, H. Liu, and Z. Bao. A kernel optimization method based on the localized kernel Fisher criterion. *Pattern Recognition*, 41(3):1098–1109, March 2008.

[16] John B. Copas and Guobing Lu. Missing at random, likelihood ignorability and model completeness. *The Annals of Statistics*, 32(2):754–765, April 2004.

[17] Corinna Cortes and Vladimir Vapnik. Support vector networks. In *Machine Learning*, volume 20, pages 273–297, 1995.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[19] Uwe Dick, Peter Haider, and Tobias Scheffer. Learning from incomplete data with infinite imputations. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 232–239, New York, NY, USA, 2008. ACM.

[20] Q. Du. A linear constrained distance-based discriminant analysis for hyperspectral image classification. *Pattern Recognition*, 34(2):361–373, February 2001.

[21] Trygve Eftestol, Kjetil Sunde, Ole, John H. Husoy, and Petter A. Steen. Predicting outcome of defibrillation by spectral characterization and nonparametric classification of ventricular fibrillation in patients with out-of-hospital cardiac arrest. *Circulation*, 102(13):1523–1529, September 2000.

[22] Rong E. Fan, Pai H. Chen, and Chih J. Lin. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, 6:1889–1918, 2005.

[23] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[24] Gabriel Frahm and Uwe Jaekel. A generalization of Tyler's M-estimators to the case of incomplete data. *Computational Statistics & Data Analysis*, 54(2):374–393, February 2010.

[25] Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT '90: Proceedings of the third annual workshop on Computational learning theory*, pages 202–216, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[26] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Computer Science and Scientific Computing Series. Academic Press, 2nd edition, September 1990.

[27] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems 6*, volume 6, pages 120–127, 1994.

[28] Daniel Gildea. The Wolfe dual, Spring 2008. University of Rochester. Retrieved 19th November 2009 from `http://www.cs.rochester.edu/~gildea/2008_Spring/wolfe.pdf`.

[29] T.B. Gosnell. Gamma-ray identification of nuclear weapon materials. Technical Report UCRL-ID-127346, Lawrence Livermore National Laboratory, February 1997.

[30] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, December 1995.

[31] Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[32] Jintae Han, Hoeil Chung, Sung-Hwan Han, and Moon-Young Yoon. Score-moment combined linear discrimination analysis (SMC-LDA) as an improved discrimination method. *Analyst*, 132(1):67–74, 2007.

[33] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *J. Mach. Learn. Res.*, 5:1391–1415, 2004.

[34] Melanie Hilario, Alexandros Kalousis, Christian Pellegrini, and Markus Müller. Processing and classification of protein mass spectra. *Mass Spectrometry Reviews*, 25(3):409–449, 2006.

[35] A. Honkela, H. Valpola, A. Ilin, and J. Karhunen. Blind separation of nonlinear mixtures by variational Bayesian learning. *Digital Signal Processing*, 17(5):914–934, September 2007.

[36] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[37] R. Jenssen, J. Principe, D. Erdogmus, and T. Eltoft. The Cauchy-Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, 343(6):614–629, September 2006.

[38] W. Kang and J. Choi. Domain density description for multiclass pattern classification with reduced computational load. *Pattern Recognition*, 41(6):1997–2009, June 2008.

[39] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[40] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[41] Glenn F. Knoll. *Radiation detection and measurement*. Wiley, New York, 3rd edition, 2000.

[42] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent E. Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

[43] Yann Lecun and Corinna Cortes. The mnist database of handwritten digits, 1998 – 2009. Available at `http://yann.lecun.com/exdb/mnist/`.

[44] Daewon Lee and Jaewook Lee. Domain described support vector classifier for multi-classification problems. *Pattern Recognition*, 40(1):41–51, January 2007.

[45] Z. Liang, D. Zhang, and P. Shi. Robust kernel discriminant analysis and its application to feature extraction and recognition. *Neurocomputing*, 69(7-9):928–933, March 2006.

[46] Mark Lillibridge, Martin Abadi, Krishna Bharat, and Andrei Z. Broder. Method for selectively restricting access to computer systems, April 1998. United States Patent 6195698.

[47] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. Wiley series in probability and statistics. Wiley, Hoboken, N.J., 2nd edition, 2002.

[48] M. Lobo. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228, November 1998.

[49] Junshui Ma, J. L. Sancho-Gomez, and S. C. Ahalt. Nonlinear multiclass discriminant analysis. *IEEE Signal Processing Letters*, 10(7):196–199, 2003.

[50] David J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge Univ. Press, 2003.

[51] S. G. Mallat. A theory for multiresolution signal decomposition, the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[52] Benjamin Marlin. *Missing data problems in machine learning*. PhD thesis, University of Toronto, 2008.

[53] Aleix M. Martinez and Avinash C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, February 2001.

[54] Tshilidzi Marwala. *Computational intelligence for missing data imputation, estimation and management : knowledge optimization techniques*. Information Science Reference,, Hershey, Pa., 2009.

[55] The Mathworks. Matlab optimization toolbox user's guide.

[56] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[57] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K. R. Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):623–628, 2003.

[58] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[59] Dean Oliver, Nanqun He, and Albert C. Reynolds. Conditioning permeability fields to pressure data. In *Proceedings of the 5th European conference on the Mathematics of Oil Recovery*, September 1996.

[60] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[61] M.E.H. Pedersen and AJ Chipperfield. Simplifying particle swarm optimization. *Applied Soft Computing*, 10(2):618–628, 2010.

[62] Claudio A. Perez, Claudio M. Held, and Pablo R. Mollinger. Handwritten digit recognition based on prototypes created by euclidean distance. *Information, Intelligence, and Systems, International Conference on*, 0:320, 1999.

[63] Donald B. Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley, Hoboken, N.J., Wiley Classics Library edition, 2004. Previous ed.: New York : John Wiley, 1987.

[64] Donald D. Rubin. Inference and missing data. *Biometrika*, 3(63):581–592, 1976.

[65] Joseph L. Schafer. *Analysis of incomplete multivariate data*. Monographs on statistics and applied probability ; 72. Chapman & Hall,, London :, 1997.

[66] Sandro Scheid. *Selection models for nonignorable missing data*. Anwendungsorientierte Statistik ; Bd. 8. P. Lang,, Frankfurt am Main; New York, 2005.

[67] Bernhard Schölkopf, John C. Platt, John, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[68] Bernhard Schölkopf, Alexander Smola, and Klaus R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[69] Bernhard. Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press,, Cambridge, Mass. :, c2002.

[70] Pannagadatta K. Shivaswamy, Chiranjib Bhattacharyya, and Alexander J. Smola. Second order cone programming approaches for handling missing and uncertain data. *J. Mach. Learn. Res.*, 7:1283–1314, 2006.

[71] Karl Sjöstrand, Michael S. Hansen, Henrik B. Larsson, and Rasmus Larsen. A path algorithm for the support vector domain description and its application to medical imaging. *Medical Image Analysis*, 11(5):417–428, October 2007.

[72] Alex Smola and S. V. N. Vishwanathan. Kernel methods for missing variables. In *In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 325–332, 2005.

[73] Alex J. Smola, Bernhard Schölkopf, and Klaus R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649, 1998.

[74] I. Steinwart, D. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. *IEEE Transactions on Information Theory*, 52(10):4635–4643, 2006.

[75] C. J. Sullivan, M. E. Martinez, and S. E. Garner. Wavelet analysis of sodium iodide spectra. *IEEE Transactions on Nuclear Science*, 53(5):2916–2922, 2006.

[76] M.A. Tanner and W.H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540, 1987.

[77] David M. J. Tax and Robert P. W. Duin. Support vector domain description. *Pattern Recogn. Lett.*, 20(11-13):1191–1199, 1999.

[78] David M. J. Tax and Piotr Juszczak. Kernel whitening for one-class classification. In *SVM '02: Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines*, pages 40–52, London, UK, 2002. Springer-Verlag.

[79] De La Torre and O. Vinyals. Learning kernel expansions for image classification. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7, 2007.

[80] V. Vapnik. Estimation of dependences based on empirical data. *Nauka*, 1979.

[81] Vladimir N. Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer,, New York :, 2nd edition, c2000.

[82] K. Wagstaff. Clustering with missing values: No imputation required. In *Classification, Clustering, and Data Mining Applications (Proceedings of the Meeting of the International Federation of Classification Societies)*, pages 649–658, 2004.

[83] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *ICML '04: Proceedings*

of the twenty-first international conference on Machine learning, New York, NY, USA, 2004. ACM Press.

[84] James William Weir. A review of some missing data methods. Thesis (M.Sc.) - University of Glasgow, 2006.

[85] Y. Xu, D. Zhang, F. Song, J. Yang, Z. Jing, and M. Li. A method for speeding up feature extraction based on KPCA. *Neurocomputing*, October 2006.

[86] Jian Yang, David Zhang, Jing-Yu Yang, and Ben Niu. Globally maximizing, locally minimizing: Unsupervised discriminant projection with applications to face and palm biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(4):650–664, 2007.

[87] Tao Yang and Vojislav Kecman. Adaptive local hyperplane classification. *Neurocomputing*, 71(13-15):3001 – 3004, 2008. Artificial Neural Networks (ICANN 2006) / Engineering of Intelligent Systems (ICEIS 2006).

[88] Yu Zhang and Eric Sung. EDFCES: a new example-driven 3D face construction and editing system. *MG&V*, 17(3):313–346, 2008.

[89] X. Zhuang and D. Dai. Improved discriminate analysis for high-dimensional data. *Pattern Recognition*, 40(5):1570–1578, May 2007.